Title
Targeted Genome Mining for the Discovery and Study of Sterol Pathway Fungal Natural Product Drugs

Author
Liu, Nicholas

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Targeted Genome Mining for the Discovery and Study of Sterol Pathway Fungal Natural Product Drugs

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Chemical and Biomolecular Engineering

by

Nicholas Liu

2020

ABSTRACT OF THE DISSERTATION

Targeted Genome Mining for the Discovery and Study of Sterol Pathway Fungal Natural Product Drugs

by

Nicholas Liu

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2020

Professor Yi Tang, Chair

For millennia, humans have been utilizing plants, fungi, and microbes for their medicinal purposes or for commercial use as dietary supplements, cosmetics, etc that are derived from natural products. Natural products can be broadly defined as any chemical compound that can be found from living organisms in nature. Natural products have a wide variety of bioactivities and uses that have spurred efforts to discover and characterize novel natural products. The genesis of these efforts in the modern era began with the discovery of penicillin by Sir Alexander Fleming, which helped realize the power of utilizing chemical compounds from nature. Natural products are very much like superheroes; they all have a backstory and Nature draws the comic books panels that depict their tales in the form of genetic information found in living organisms. The study of natural product biosynthesis thus aims to decipher the narratives told through the ATCG

nucleotides to explain the origins of these molecules through enzymatic pathways and their superpowers (bioactivities).

The rise of next generation sequencing has opened a new world of genetic information and facilitated a renaissance in drug discovery based on genomics. Analysis of microbial genomes has revealed that we have only accessed <10% of the chemicals these organisms are capable of producing. This dissertation describes efforts to access this genomic space in fungi with targeted genome mining, which aims to search for natural products of desired bioactivity through the presence of self-resistant enzymes. Fungi have evolved to contain these self-resistant enzymes, which are copies of the target enzyme of a natural product that are resistant to the inhibitory effects of the produced molecule.

We aimed to utilize targeted genome mining to discover natural product inhibitors of the sterol pathway, which already contains drug targets that many commercial anticholesteremic and antifungal drugs specifically inhibit. This search strategy allowed us to discover and elucidate the biosynthesis of zaragozic acid A (a cholesterol lowering natural product) and restricticin (an antifungal natural product). Discovery of the zaragozic acid A cluster in *Curvularia lunata* was done through targeted genome mining of squalene synthase. The elucidation of the zaragozic acid A biosynthetic pathway in the engineered *Aspergillus nidulans* heterologous host has described the unique steps that lead to the production of the alkylcitrate benzylic polyketide intermediate of the pathway. Restricticin is another example of successful targeted genome mining using Cyp51 as a query. Cyp51 is an important sterol pathway target that is inhibited by commercial azole drugs. Targeted searches for Cyp51 clusters led to the identification of the restricitin biosynthetic gene cluster in *Aspergillus nomius*. Biosynthetic elucidation of restricticin

reveals the critical steps that lead up to attachment of the glycyl ester group that serve as the chemical warhead for its antifungal properties. Identification of the cluster allowed us to evaluate the *rstn2* self resistance gene which exhibits azole resistance. Targeted genome mining has also helped to identify other possible novel Cyp51 inhibitors.

The dissertation of Nicholas Liu is approved.

Yvonne Y. Chen

Neil Kamal Garg

Junyoung O. Park

Yi Tang, Committee Chair

University of California, Los Angeles

2020

Table of Contents

Table of figures

Acknowledgements

Section 1.2.1 contains material written by Liu, N. from the following publication:

Hang, L.[#], and **Liu, N.**[#], Tang, Y.* "Coordinated and Iterative Enzyme Catalysis in Fungal Polyketide Biosynthesis." *ACS Catalysis* **2016**, 6, 5935–5945. PMCID: PMC5436725

Section 1.2.2 contains material written by Liu, N. from the following publication:

Zhang, J., **Liu, N.**[#], Cacho, R. A.[#], Gong, Z., Liu, Z., Qin, W., Tang, C., Tang, Y.*, Zhou, J.*, "Structural Basis of Nonribosomal Peptide Macrocyclization in Fungi." *Nat. Chem. Biol*. **2016**, 12, 1001-1003. PMCID: PMC5110376

Sections 3.2.2 and Section 4 contains material written by Liu, N. from the following publication:

Liu, N., Hung, Y., Gao, S-S., Hang, L., Zou, Y., Chooi, Y-H.*, Tang, Y.*, "Identification and Heterologous Production of a Benzoyl-Primed Tricarboxylic Acid Polyketide Intermediate from the Zaragozic Acid A Biosynthetic Pathway." *Org. Lett*. **2017**, 19, 3560–3563. PMCID: PMC5673471

This work was supported by the by NIH 1R35GM118056 and 1DP1GM106413 as well as supported by NIH Biotechnology Training in Biomedical Sciences and Engineering (T32GM067555).

There are many people that I would like to thank for their help along this long and arduous journey. First, I would like to thank Professor Yi Tang for his 6 years of guidance. When

I first came in, I was intimidated by the seemingly huge commitment. He told me that these 5 years would past extremely fast and I'll be out before I knew it. Indeed, the last 5 years (+1) have indeed passed by in a blink of an eye and I still find myself wondering how much still needs to be done. PT has been extremely patient with me through the ups and downs throughout my graduate school career and I really do appreciate the patience especially during the low times in my studies. I came into this lab not knowing what cloning was and now I feel like I could build any recombinant plasmid I wanted to. PT has always been a guiding light in my times of darkness in this at times seemingly untraversable science landscape. I greatly value all the insight that he provides during our meetings and it is with his mentorship that I feel I was able to grow as a scientist. Go Lakers.

I would like to thank my committee members Professor Yvonne Chen, Professor Neil Garg, Professor Junyoung Park, and Professor Tatianna Segura who have been nothing but helpful during my prospectus in giving my valuable criticism and recommendations. I truly appreciate the time that they have taken out of their days to listen to me talk about my research.

Next I would like to thank my cohorts and partners in crime, the members of the Fungal Journal Club: John Billingsley, Leibniz Hang, and Yan Yan. These three individuals were my life support during our PhD studies and are everything I could wish for in labmates. They made me laugh and cry (mostly laugh) and taught me so much about life and science that I would not be close to where I am today without them.

I would like to thank my mentors Ralph Cacho and Yi Zou who were the ones who took me under their wing when I first started out in the lab. It was with their help that I was able to get accustomed to the lab and get a jump start on what I needed to know to be successful in the lab. I would like to thank all the members of the Tang lab who have helped me over the years:

Mancheng Tang, Shushan Gao, Carly Bond, Anthony Denicola, Sunny Hung, Danielle Yee, Undramaa Bat-Erdene, Eun Bin Go, Joshua Misa, Ike Okorafor, Wei Xu. These members formed the community that gave me all I needed to learn and progress. I've learned many things from all of them and it is because of this diverse group of members that I believe our lab have such success. I'd also like to give a special shoutout to Masao Ohashi who was my late-night crew buddy who was always available to help even with the massive responsibility he had on his shoulders. He has always been a stabilizing force in the lab and willing to help any person who needed it.

I would like to give a special thank you to the undergraduate who has worked with me for the last two years, Elizabeth Abramyan. She has been nothing but dedicated in her assistance in helping me work through the last couple of years. Without her hard work I would not have been able to accomplish many of the things that I set out to accomplish. She has also been a great friend during all the stressful times and a great person to talk to when things are rough.

I'd like to thank my friends outside of the lab including Ximin Chen, Freddy Chen, Duo Xu, and Xianyang Lee. Those study sessions in Weyburn made the learning experience fun and helped us get through the preliminary exams through communal suffering. I'd like to thank my good hometown friends Anthony Nguyen, Elizabeth Ko, Lihan Woo, Eric Fan, Howard Chang and Ekta Doshi who are always a great sight to see during the short breaks that we have. They have always been a joy to be around and have been a great support network no matter where in the world they are.

Finally, I'd like to thank my wonderful family who has supported me through these years. Thanks to my dad who has always been there for me. Those trips to come visit me at UCLA and bring me supplies and food have always been the highlight of any week. I'd like to thank my

mom whose love shines through her constant nagging, but it is through the nagging that I remember to do the right things to keep myself healthy. The consistency in which my parents have cared for me over these years has truly been a blessing that I sometimes take for granted but will always treasure. Thanks to my brother who is a bundle of joy to be around and I will always love the witty banter and snarky remarks that we make to each other. All I wish is to make you guys proud.

VITA

2014- present	UCLA, Henry Samueli School of Engineering and Applied Science Ph.D. Candidate, Chemical and Biomolecular Engineering Los Angeles, CA
2010 – 2014	Johns Hopkins University, Whiting School of Engineering B.S. in Chemical and Biomolecular Engineering Minor in Entrepreneurship and Management Baltimore, MD
2016 – 2018	NIH Biotech Training in Biomedical Sciences and Engineering Fellowship University of California, Los Angeles Los Angeles, CA

Publications

Liu, N., Hung, Y., Gao, S-S., Hang, L., Zou, Y., Chooi, Y-H.*, Tang, Y.*, "Identification and Heterologous Production of a Benzoyl-Primed Tricarboxylic Acid Polyketide Intermediate from the Zaragozic Acid A Biosynthetic Pathway." *Org. Lett.* **2017**, 19, 3560–3563. PMCID: PMC5673471

Yan, Y., **Liu, N**., Tang, Y.* "Recent developments in self-resistance gene directed natural product discovery." ***Nat. Prod. Rep,* 2020.**

Zhang, J., Liu, N.[#], Cacho, R. A.[#], Gong, Z., Liu, Z., Qin, W., Tang, C., Tang, Y.*, Zhou, J.*, "Structural Basis of Nonribosomal Peptide Macrocyclization in Fungi." *Nat. Chem. Biol*. **2016**, 12, 1001-1003. PMCID: PMC5110376

Hang, L.[#], and Liu, N.[#], Tang, Y.* "Coordinated and Iterative Enzyme Catalysis in Fungal Polyketide Biosynthesis." *ACS Catalysis* **2016**, 6, 5935–5945. PMCID: PMC5436725

Yee, D. A.; Kakule, T.; Cheng, W.; Chen, M.; **Chong, C**.; Hai, Y.; Hang, L.; Hung, Y.; **Liu, N.**; Ohashi, M.; Okorafor, I.; Song, Y.; Tang, M-C.; Zhang, Z.; Tang, Y.* "Genome Mining of Alkaloidal Terpenoids from a Hybrid Terpene and Nonribosomal Peptide Biosynthetic Pathway." ***J. Am. Chem. Soc.* 2020**, 142, 710-714**.** PMCID: PMC7000236

Presentations

2016 American Institute of Chemical Engineers (AIChE) Annual Meeting. November 14, 2016, Hilton San Francisco Union Square. "Structural Basis of a Macrocyclization in a Fungal Nonribosomal Peptide through a Condensation-like Termination Domain in Fungi."

2013 MedImmune Summer Conference. August 15, 2013, Gaithersburg, Maryland. "High Protein Concentration of Monoclonal Antibody Suspensions in Nonaqueous Solvents."

1. Introduction and History of Natural Products

Since Sir Alexander Fleming first discovered penicillin in 1938, humans have realized the potential of extracting compounds from nature as a source of biologically relevant drugs. Since then, over 23,000 natural products have been characterized, with most of them being derived from bacteria.[1] Microorganisms in particular have proven to be prolific producers of natural products of a wide spectrum of bioactivities, ranging from the anti-cholesterol drug, Lovastatin, to the anti-bacterial, erythromycin, etc. (Figure 1).[2] These natural products are also known as secondary metabolites and are often not essential for life, but still serve as chemical weapons, facilitators of symbiosis, sexual effectors, differentiation effectors, or metal-transporting agents that are still important to the functions of the cell.[3] The wide variety of bioactivities can often translate to effective pharmacological drugs as approximately 50% of the new chemical entities approved by the US Food and Drug administration in the last 30 years have been either natural products or natural product derived.[1, 4] In addition to the various applications that natural products pose, the structural complexity and diversity of natural products also have drawn the interests of synthetic chemists to develop total syntheses for producing the molecules. Academic interest in studying the natural product biosynthesis pathways of these complex molecules has also grown in order to understand the regulation, enzymology, and chemical mechanisms that nature employs to synthesize these compounds. Natural products research covers many different disciplines to explore the secrets behind the chemical treasures that Nature offers.

1.1 Methods of Discovery for Natural Products

Natural products discovery has spanned over many decades. With the rise of technological advances over the years, the techniques used to discover and study Natural

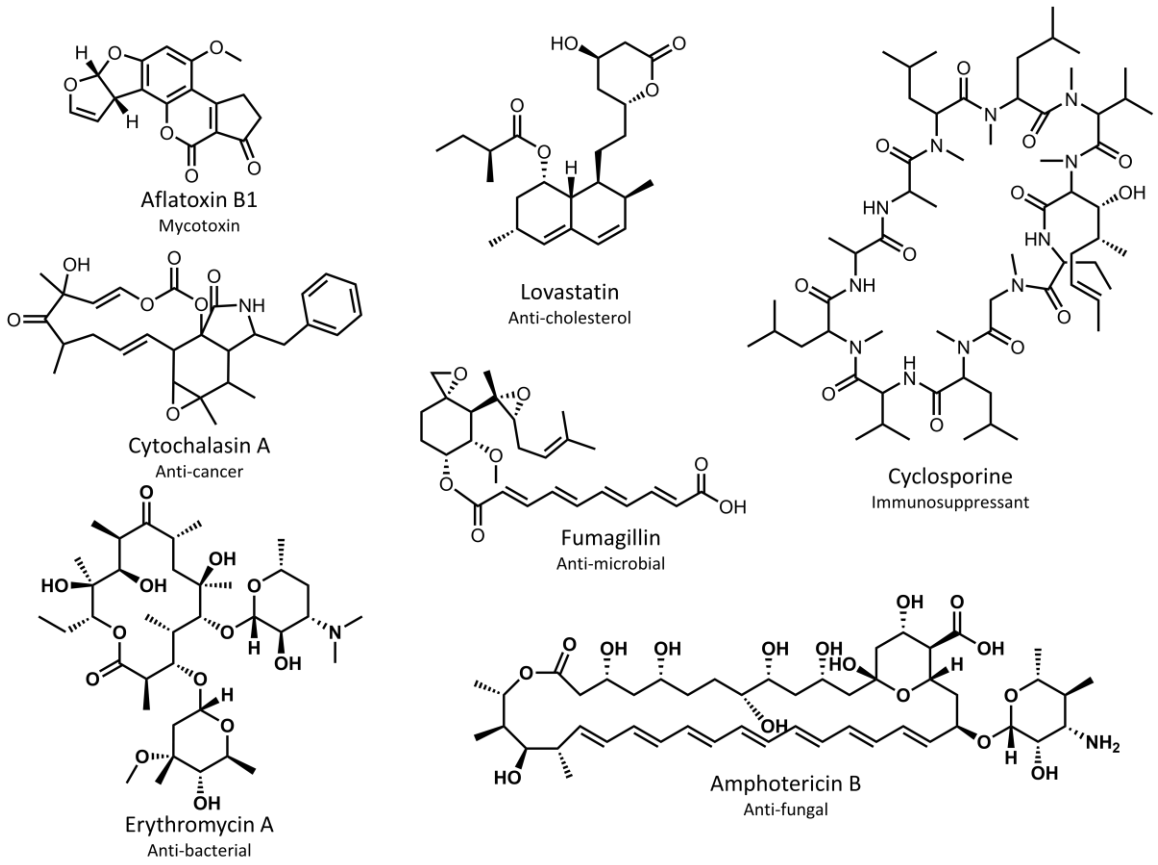


Figure 1. Chemical and Biological Diversity of Natural Products
Natural Products are diverse in both chemical structure and bioactivity. Many of them are highly impactful pharmaceutical drugs and novel products are being discovered all the time.

products have also expanded. After the discovery of penicillin, efforts to begin identifying other valuable compounds produced from microorganisms began. Early efforts involved the search for microorganism strains that would be capable of killing fungi or bacteria through phenotypic screens in the 1940s. Phenotypic screens, however, were limited in their ability to find compounds of specific desired activities. Thus, around the 1970s, target activity guided screens were also developed as ways to find molecules of interest with specific activity against enzyme targets. In addition to screening methods for new natural products, genetics and enzymatic studies on natural product biosynthesis pathways revealed many fundamental components behind

natural product biosynthesis. With the rise of genomics and next generation sequencing in the early 2000s, genes responsible for the biosynthesis of natural products could be readily identified. This has led to a renaissance in natural product discovery as it is now apparent many of natural product genes are cryptic or silent under laboratory conditions, leaving much more to be discovered from natural producers of secondary metabolite drugs.

1.1.1 Phenotypic Screening

The discovery of penicillin first introduced the idea behind fermenting microorganisms to mass produce a chemical of interest. World War II had generated the need for mass production of antibiotics, so efforts to improve scalability of natural product fermentation began then. This started with identifying the best producing strains of penicillin as well as ways to ferment those strains in 10,000-gallon tanks. The interdisciplinary collaboration of scientists began the advent of natural product developement for not only clinical relevance, but also commercial success.[5] The development of penicillin in the 1940s also indicated that bacterial strains were valuable leads to natural products. It was soon found that *Actinomyctes* and *Streptomycetes* were fruitful producers of many important compounds. *Streptomyces* in particular proved to be a rich source of antibiotics including macrolides, tetracyclines, polyenes, and peptides.[1] Thus began the era of phenotypic screening of microbes for the discovery of relevant natural products. Scientists would test different fermentation conditions of various strain isolates from soil. The crude extracts could be tested directly or fractionated to test against bactericidal or fungicidal screens, often simply performed by looking at visual growth inhibition.[6] Identification of desired phenotypes would be subsequently followed up with chemical structure elucidation and studies about the mechanism of action. In addition to antibacterial screens, companies started as early as the 1950s to look toward finding anti-cancer agents through phenotypic screens. These screens were

performed with induced tumors in whole animal tests to search for drugs capable of cytotoxic effects.[7]

These types of studies led to the discovery of important molecules such as the antibiotic, tetracycline, by Pfizer et al. in 1952[8] and the immunosuppressant, cyclosporine A, by Sandoz (Novartis) in 1972.[9] Tetracycline is a semi-synthetic drug that was developed through modification of the natural product Aureomycin. Aureomycin was a promising natural product compound found by Cyanamid during the boom of antibiotic discovery in the 1940s. Cynamid was testing soil sample extracts against gram positive and gram negative bacteria and found one sample in particular was able to cause large inhibition zones on the bacterial agar plates of various different strains and had a distinct yellow color. Later, Pfizer had realized the C7 chlorine of Aureomycin was not necessary for the activity of the drug, and thus the descholor version was developed as tetracycline. Tetracycline had higher potency, better solubility, and better activity.[8] Cyclosporine was first discovered when the Sandoz lab searched for anti-inflammatory and immunosuppressant drugs by observing the effects of natural product libraries for compounds on lymophocte mediated effector cell lysis. Cyclosporine was a promising candidate that inhibited the *in vitro* cell lysis of the allogenic target cells.[9]

Phenotypic screens still remain an important forward pharmacology discovery technique, though the types of strains screened through have expanded toward many different sources, including marine based microbes rather than traditional soil derived samples.

1.1.2 Target Activity Guided Screens

Phenotypic screens have proven to be an efficient way to discover new natural product compounds. However, with phenotypic screens the mode of action is often still unknown, resulting in a surface understanding of the novel natural products discovered. With the

advancement of biochemical techniques, the understanding of natural biochemical pathways and enzymes involved in these pathways has greatly improved. The specific modes of action

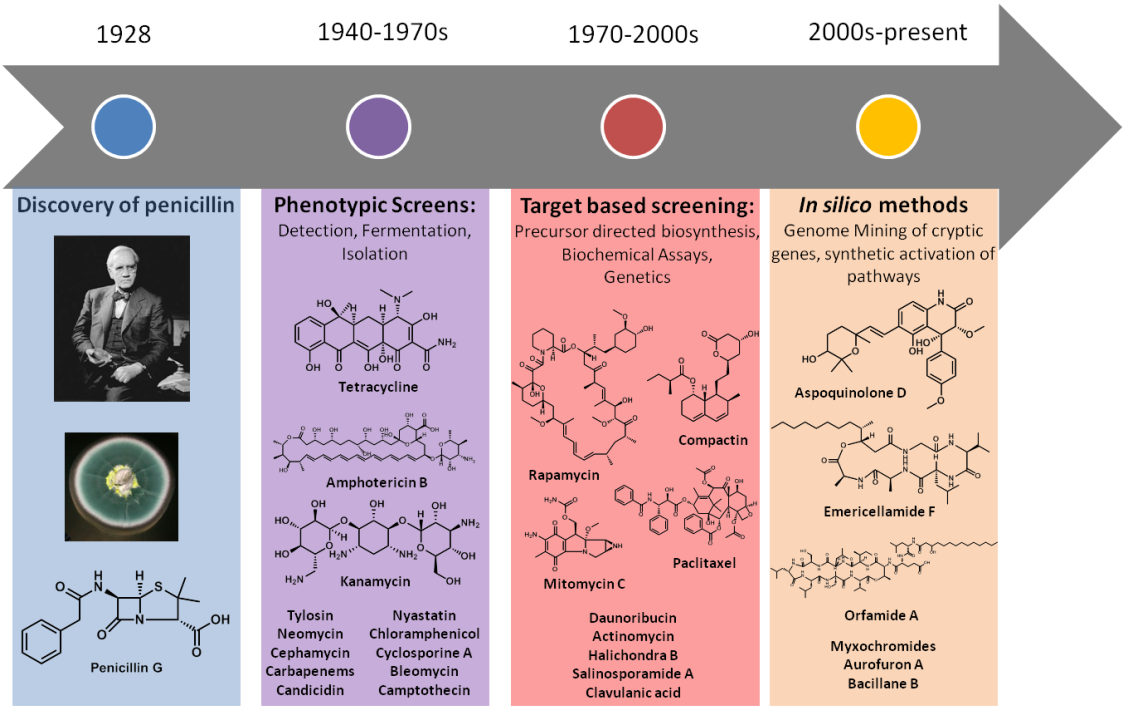


Figure 2. Timeline of Natural Product Discovery
This timeline displays the different natural product discovery techniques that were developed over different time periods and significant natural product molecules found during these time periods

between the molecule and the specific target enzymes could be tested with developed activity assays using a specific custom readout. This allowed for the onset of target activity guided screens, where scientists could use high throughput assays to test for natural products that would specifically inhibit or interact with the target of interest.

One example of an early natural product discovered through the use of target activity guided screens was compactin. The early 1950's came with many efforts to understand the cholesterol pathway after it was found that heart disease was linked with the cholesterol levels of

the patient. The 30 enzymatic reactions involved to synthesize cholesterol were fully elucidated. Akira Endo had first hypothesized that fungi and mold might be capable of producing inhibitors of hydroxymethylglutaryl-CoA reductase (HMGR). He had developed a multi-step assay to screen for compounds that had HMGR inhibitory activity. They screened through fermentation broths to see which could inhibit the incorporation of radio labeled C^{14} acetate into lipid products such as cholesterol. Successful broths were then checked for inhibitory activity of the conversion of labeled H^3 mevalonate to labeled lipid products. Broths that passed this step would be subjected to inhibitory evaluation of C^{14}-HMG-CoA to C^{14}-mevalonate. This led to discovery of compactin from *Penicillium citrinum*.[10] These precursor directed studies also allowed for scientists to identify the building blocks that are incorporated during natural products biosynthesis.

1.1.3 Bioinformatics Guided Discovery

The rise of genetic tools eventually came to fruition, providing researchers with an alternative strategy to better study natural product biosynthesis. It was soon understood that natural product genes are often colocalized as gene clusters in fungi and bacteria. The organization of these gene clusters will be further explored in Section 1.2.6. Consequently, gene clusters were being linked to the natural products that were identified through phenotypic and activity-based screens. Notwithstanding the new knowledge about natural product biosynthesis gained from genetic analysis, natural product discovery soon began to run into issues of dereplication using phenotypic and targeted based screens. Simply going through different types of new strains and fermentation broths combinations would lead to rediscovery of many already known natural products. Thus efforts to find new natural products eventually began to slow down.[11] In the early 2000's, however, the first two *Streptomyces* genome sequences were fully

assembled, launching the revelation that the number of gene clusters in these microbial genomes were far greater in number than previously thought. It is estimated that only that less than 10% of the secondary metabolite gene clusters are expressed during laboratory fermentation conditions, indicating that many of these gene clusters are cryptic.[1] With the sequencing of microbial genomes also came the opportunity to tie together natural product compounds and the genes that produce them, allowing us to readily identify new clusters or activate cryptic ones. The natural product landscape entered into a new era of discovery that brought upon a natural products renaissance that is still being explored today.

Deeper understanding about natural product biosynthesis also gave insight into the building blocks that go into making these secondary metabolites. Thus based on the type of enzymes in a natural product biosynthetic gene cluster, chemical structure could be predicted from the genes and what types of building blocks they would be using.[12] One successful example is the identification of the aspoquinolones A-D, which are prenylated alkaloids from the fungi *Aspergillus nidulans*. Efforts to look at this gene cluster began with the interest in finding alkaloid compounds in which anthranilate is a known precursor. Sequencing of the *Aspergillus nidulans* genome showed that the strain contained at least three copies of anthranilate synthase, indicating that numerous copies of the gene would allow *A. nidulans* to produce several alkaloid secondary metabolites. Through utilizing many different fermentation conditions, the aspoquinolones were identified as novel prenylated alkaloid natural products.[13]

Comparative metabolic profiling is another technique that has arisen with the development and availability of microbial genome sequences. By utilizing mutagenesis of target genes and then comparing metabolic profiles, the natural product linked to the target gene can be identified.[12] These knockout studies remain an important technique for the identification of

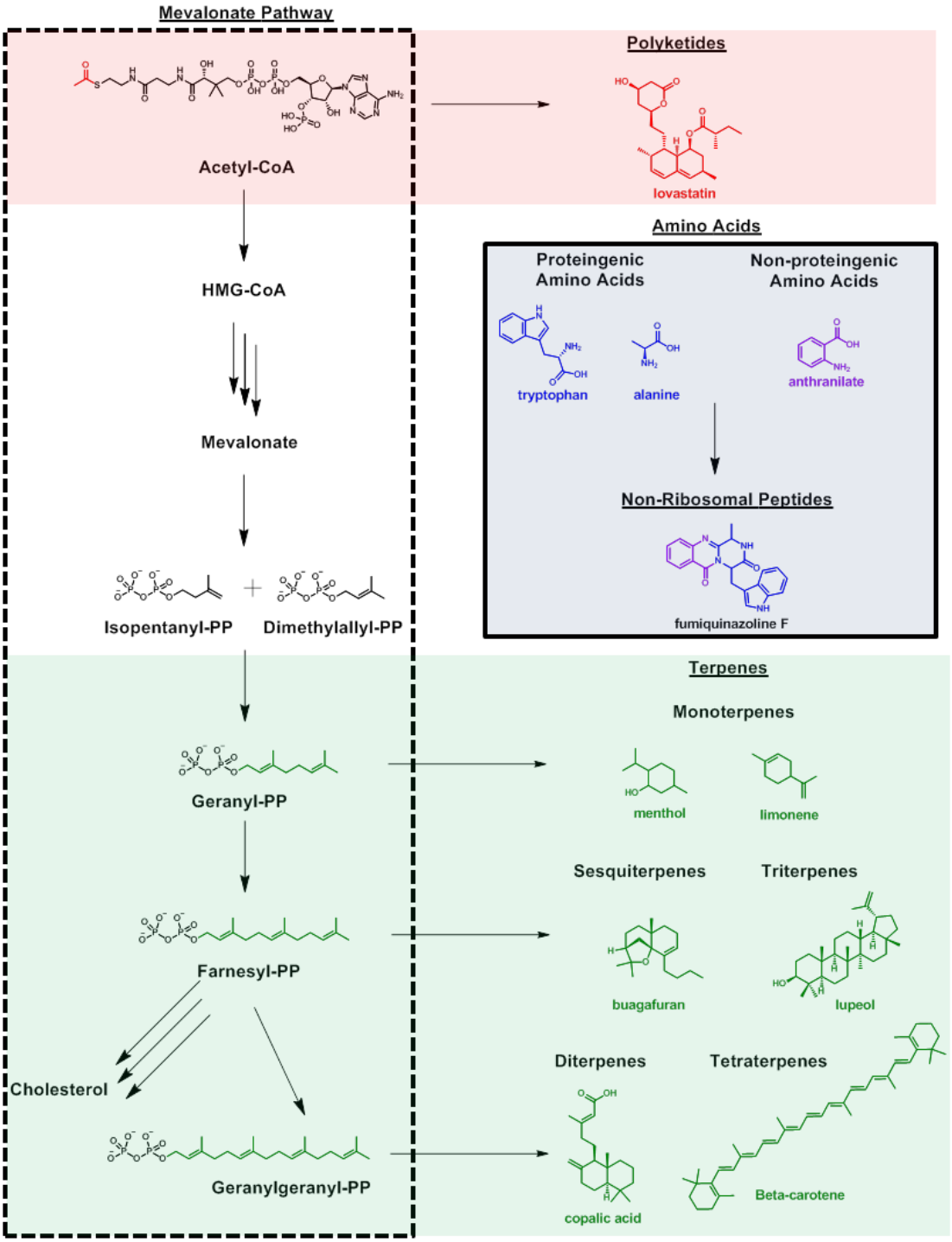


Figure 3. Sources of Natural Products from Primary Metabolism

Natural Products derived their building blocks from various primary metabolic processes. Polyketides and terpenes use intermediates from the mevalonate pathway while Non-ribosomal peptides use both proteingenic amino acids and non-proteingenic amino acids.

natural product gene clusters. The emericellamides were a family of cyclic depsipeptide products that were that were initially found in the marine fungi *Emericella sp.* Through the knockout of a cryptic nonribosomal peptide synthetase (NRPS) gene, the biosynthetic pathway of the emericellamides was realized.[14] In another example, orfamide A was a novel peptide antibiotic isolated by Müller et al. They later identified gene fragments of novel NRPS and polyketide synthase (PKS) encoding gene fragments. Inactivation of these gene fragments in the wild type producing strain *Stigmatella aurantica* and through comparative metabolic profiling saw the loss of orfamide A production in the mutant strains, indicating the role of these gene fragments in the biosynthesis of orfamide A.[2]

1.2 Natural Product Biosynthesis

The molecular scaffolds of most natural products are furnished by different classes of natural product enzymes: 1) Polyketide Synthases (PKSs), 2) Non-ribosomal Peptide Synthetases (NRPSs), and 3) Terpene Synthases (TSs). These "core enzymes" utilize building blocks from central metabolic pathways and primary metabolism (Figure 3). PKSs are fatty acid like multi-domain megasynthases that take acetyl-CoA from the mevalonate pathway to build long polyketide chains with varying degrees of reduction at each polyketide chain segment. NRPSs take both proteinogenic and non-proteinogenic amino acids to form peptide chains. Terpene synthases take 5 carbon unit isoprene building blocks from the mevalonate pathway to undergo cyclization reactions for the synthesis of terpenoid products. These natural product backbones are often further modified by tailoring enzymes to introduce further chemical complexity. Understanding how these specialized core and tailoring genes work together has given us clues into how they are organized, allowing us to better find and study the biosynthetic gene clusters that govern natural product biosynthesis.

1.2.1 Polyketide Synthases

Polyketides are typically long carbon chains reminiscent of fatty acid chains, synthesized by PKSs. These PKSs typically utilize acetyl-CoA and malonlyl-CoA units from the early mevalonate pathway to build the carbon chain products. PKSs can be characterized into three types: 1) Type 1 PKS: multifunctional enzymes with different domains that catalyze specialized reactions in a modular fashion 2) Type II PKS: a set of polyketide domains that only consist of the ketosynthase domain and acyl carrier protein that act iteratively to generate the polyketide product. 3) Type III PKS: single domain ketosynthases domains that do not use acyl carrier proteins.[15] Most fungal polyketide natural products are made by Type I PKSs.

Type I PKSs are multi-domain megasynthases that possess the catalytic domains required for polyketide biosynthesis.[15] In most type I bacterial PKSs,[16] multiple sets of domains are typically compiled into modules and their biosynthesis proceeds in an assembly-line fashion. Each module of the bacterial PKS is responsible for extending the polyketide chain by one ketide unit. In addition to chain extension, each module of the PKS will reduce the extension unit by varying degrees. In contrast to bacterial PKSs, type I fungal reducing PKSs use a single set of domains in a highly programmed and permutative fashion.[17, 18] The architecture of the fungal reducing PKSs consist of the minimal fungal PKS components and the auxiliary tailoring domains (Figure 4). The β-ketoacyl synthase (KS),[19] malonyl-CoA: ACP transacylase (MAT) and acyl carrier protein (ACP)[20] form the minimal fungal PKS components—the basis for the chain-extending iterations through decarboxylative Claisen condensations (Figure 4). During each iteration, the minimal fungal PKS components catalyze the decarboxylative polymerization of malonyl-CoA to elongate the polyketide chain by a ketide (two carbons). Following each chain extension step, the ACP-bound, β-ketothioester intermediate may undergo a series of

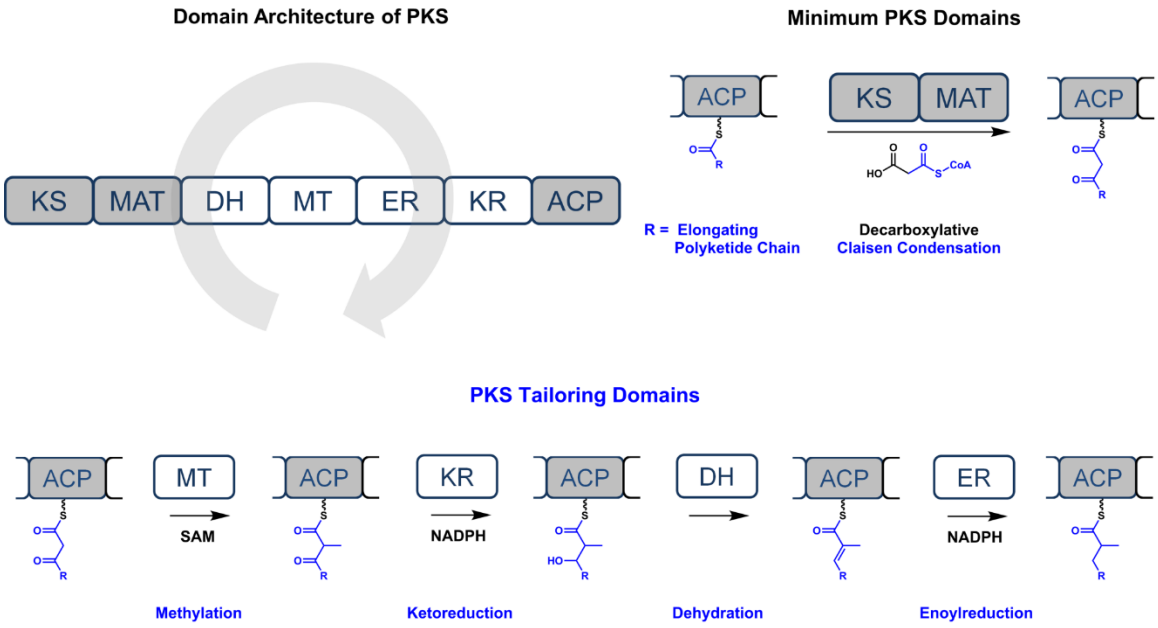


Figure 4. Mechanisms of Type I Polyketides
Fungal Polyketides are typically made from type I iterative polyketide synthases. These iterative PKS use a set of domains which include combinations of the flowing: Ketosynthase (KS), AT(acyltransferase), Dehydratase (DH), Methyltransferase (MT), Enoylreductase (ER), Ketoreductase (KR), and Acyl carrier protein (ACP) domains. These iterative PKSs utilize the domains in a programmed fashion to generate polyketide chains of varying length and reduction at each ketide unit.

modifications from the tailoring domains such as α-methylation by the methyltransferase (MT), β-ketoreduction by the ketoreductase (KR),[21] dehydration by the dehydratase (DH),[22] and enoylreduction by the enoylreductase (ER) domains (Figure 4).[23] The MT domain utilizes ***S***-adenosylmethionine (SAM) as the methylating agent while the reductive domains use nicotinamide adenine dinucleotide phosphate hydride (NADPH) as the reducing agent. The α and β position of each ketide unit will differ depending on the extent of methylation and reduction during each cycle.

Through different permutative tailoring modifications following each chain extension, the same set of tailoring domains can install structural diversity into the α- and β- positions of

polyketide backbones.[18] The elongation-tailoring events proceed iteratively until the polyketide chain extension is terminated through product off-loading such as hydrolysis or reductive release. Currently, underlying programming rules for the iterative catalysis of both bacterial and fungal polyketide synthases remain an active area of research.

Phylogenetic analysis of polyketide synthases offers one method of exploring the programming rules of the iterative fungal PKSs. The analysis of the full length and KS domain sequences of PKSs from *Aspergilli* have shown that the different PKS enzymes found in fungal genomes can be grouped into different clades. As the domain architecture for fungal iterative PKSs can vary greatly, utilizing the consensus KS domain sequences rather than the full-length protein sequences has been shown to be an effective method of "fingerprinting" the different PKSs. Overall, these sequences clade the PKSs into nonreducing PKSs (NRPKS), partially reducing PKSs (PRPKSs), highly reducing PKSs (HRPKS), and PKS-NRPS hybrids.[24] Analysis of the products produced by characterized PKSs has also shown that closely related PKSs based on KS sequence also share similarities in chemical structure. As more PKS products become elucidated, chemical structure prediction of the products from novel PKS genes becomes more and more plausible.

1.2.2 Non-Ribosomal Peptide Synthetases

NRPSs are multimodular enzymes that catalyze the biosynthesis of peptidyl natural products in a ribosome-independent manner. Much like the PKS modules covered in the previous section, each module of the NRPS typically act to extend the peptide chain by an amino acid unit. An NRPS module minimally consists of a condensation (C) domain, an adenylation (A) domain, and a Thiolation (T) domain that is post-translationally modified with a 4'-phosphopantetheine (pPant) arm (Figure 5).[25] The NRPS also often contains a termination

domain, such as a thioesterase (TE) or terminal cCondensation domain (C_T) that will releases the substrate.

The thiolation (T) domain, also known as the peptidyl carrier protein (PCP) domain is used to shuttle substrates from one domain to another. It is a vital component for the NRPS assembly line dynamics that allow for the different domains to act on the growing peptide chain.[26] The *holo*-PCP will be post-translationally modified with the pPant arm, which covalently binds the amino acid or peptide chain substrates through a thioester linkage. This is usually done using a pPant transferase, a highly conserved enzyme across all forms of life in bacteria, archaea, and eukarya. The Sfp gene from *Bacillus sp.* and the NpgA gene from *Aspergillus nidulans* have been identified to be responsible for expressing the pPant transferases vital for secondary metabolism.[27] These two enzymes have been thus critical for developing heterologous platforms to successfully express these megasynthases such as NRPSs and PKSs that utilize the pPant prosthetic.[28] The PCP acts as a "swinging arm" in order to reach into the active sites of the other NRPS domains. The structure of the PCP is highly conserved and quite small (~10 kDA). It has been shown that the modification of the PCP by the pPant arm does not cause any conformational changes in the PCP structure, indicating that the PCP protein acts as a stable platform for the flexible pPant arm to interact with the other domains.[29]

The A domain serves as a gatekeeper to the different amino acid building blocks available for the NRPS. These domains are often highly specific and will activate the appropriate amino acid substrate through adenylation of the carboxyl group by activation with ATP.[30] The ATP-activated amino acid is readily loaded onto the PCP domain. Bacterial A domains are highly conserved and the active site residues allow for prediction of the specific amino acid substrate.[31]

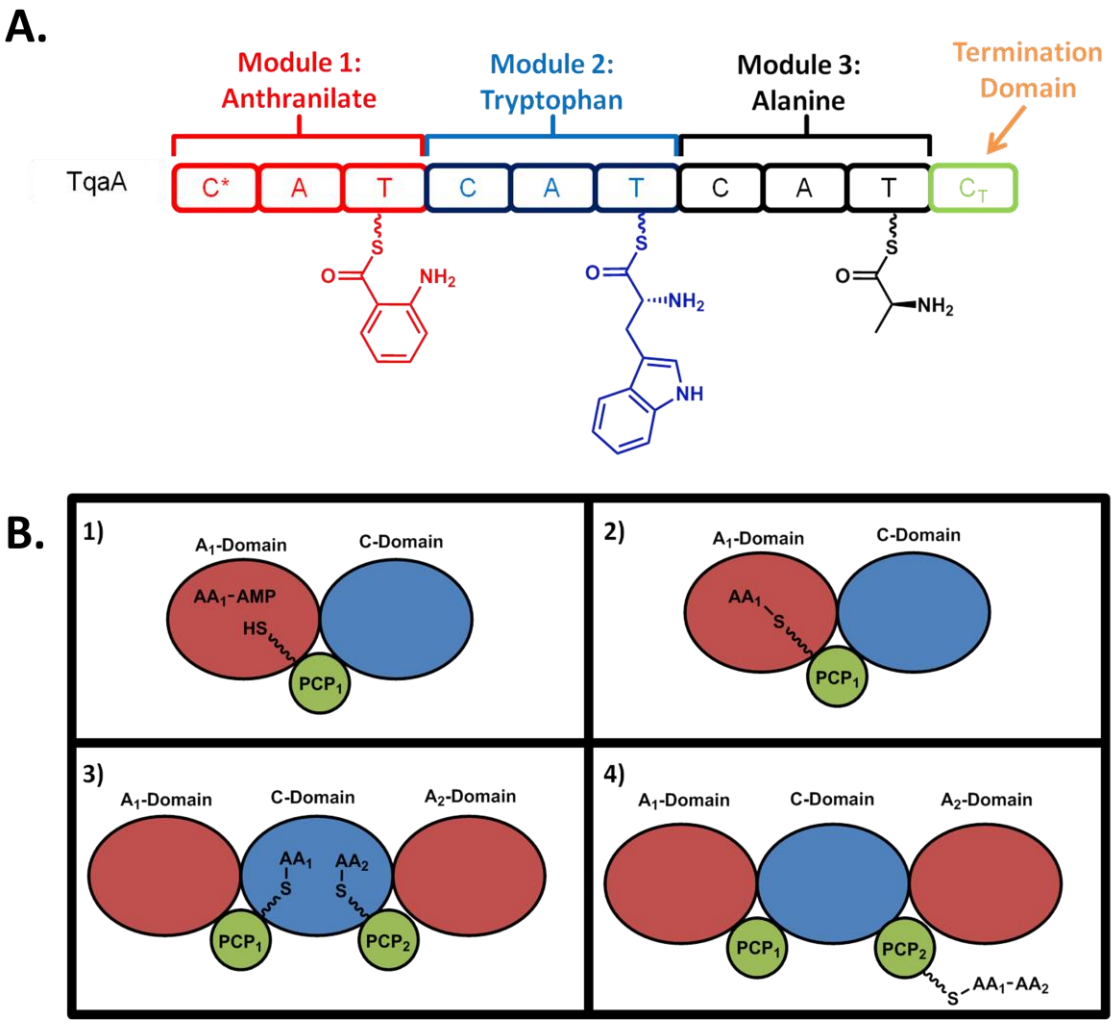


Figure 5. Non-Ribosomal Peptide Synthetases
A. Example of a NRPS organization for the cyclic tripeptide, Fumiquiazoline F. Each module is responsible for extending the peptidy chain by one amino acid. The NRPS will often have a termination domain (C_T or TE) to release the product. **B.** Dynamics of NRPSs. The PCP is responsible for swinging the peptidyl substrate to different domains to perform the necessary reaction. B1. The A domain activates an amino acid (AA) through addition of AMP to the carboxylic acid. B2. The activated AA (AA-AMP) can be tethered to the ACP through a thioester linkage. B3. The upstream PCP and the downstream PCP will enter their AA substrates into the binding pockets of the Condensation domain to be conjugated. B4. After condensation of two AA substrates, the downstream PCP will carry the peptide chain product to be processed in other domains in the assembly line until the final product is matured.

The C domain catalyzes the condensation reaction between a growing peptide that is tethered to the pPant of an upstream T domain and an aminoacyl thioester attached to the pPant

of a downstream T domain.[32-34] The C domains usually contain two active site binding pockets, one for each of the upstream and downstream aminoacyl substrates. The highly conserved enzymes will usually contain the HHXXXDG motif, which sits at the junction between the two binding pockets. After condensing the peptide bond between the upstream and downstream amino acid substrates, the C domain will shuttle the peptide chain product to the downstream T domain in order to be processed by the subsequent module.[35]

After the aminoacyl substrate has been processed by all modules of the NRPS, the peptide chain usually will be released from the NRPS through a termination domain. In most bacterial NRPSs, this is done with a TE domain. The TE domain will catalyze a two half-step reaction to for product release.[36] The first half-step involves the transfer of the peptidyl substrate from the thioester linkage on the PCP to an O-acyl onto the active site serine of the TE domain. After loading onto the TE domain, the product can either be released as a linear peptide through water hydrolysis[37], cyclization through the intramolecular nucleophilic attack of a free peptidyl moiety of the chain[38], or oligomerization[39] through nucleophilic attack from another peptidyl chain.

Whereas bacterial NRPSs use terminal thioesterase (TE) domains to perform cyclization through a nucleophilic serine, many fungal NRPSs use a C_T domain as the terminal domain to produce macrocyclic peptidyl products, including cyclosporine and echinocandin, two of the most clinically relevant fungal NRPS products.[40, 41] Although the C_T domains share the conserved HXXXDXXS motif with canonical C domains in which the histidine serves as the catalytic residue, phylogenetic analysis clearly places C_T domains in a separate clade, indicating functional divergence.[42, 43]

In addition to the minimal C, A, T, and TE or C_T domains, NRPS can also have additional tailoring domains for further functionalization of the peptidyl chain. These include Methyltransferase (MT) domains, oxidase, reductase, halogenase, and epimerization domains. These domains will be found on the module in which the specific modification of the peptidyl unit takes place.[35]

The exploration of novel NRPS products is an ongoing effort to not only discover new natural products of relevant pharmaceutical activity, but to also find new examples of NRPS architecture that can shed more light into the mechanistic rules that govern these megasynthase assembly lines.

1.2.3 Terpene Cyclases

Terpenoid products encompass a group of the most structurally diverse natural products. These terpene products can be made by terpene synthases (TS). TSs differ from PKSs and NRPSs as they do not function as multimodular systems. Rather than build a growing peptidyl/acyl chain, TSs utilize 5 carbon isoprenoid units to be modified through cyclization or rearrangement. As shown in Figure 3, most TSs draw their building blocks from the last few steps leading up to the synthesis of squalene. Different terpene products can be classified by the different building blocks they use. Monoterpenes use geranyl pyrophosphate (2 isoprene units) as building blocks to form C_{10} products. Sesquiterpenes utilize farnesyl pyrophosphate (3 isoprene units) to build C_{15} products. Triterpenes take two farnesyl pyrophosphates (6 isoprene units) to build C_{30} products. Diterpenes take geranylgeranyl pyrophosphate (4 isoprene units) to build C_{20} products. Tetraterpenes take two geranylgeranyl pyrophosphates (8 isoprene units) to build C_{40} products.[44]

TSs perform their reactions through the use of carbocation chemistry. Among TSs, prenyltransferases can catalyze the head-to-tail connections between isoprene units while terpene

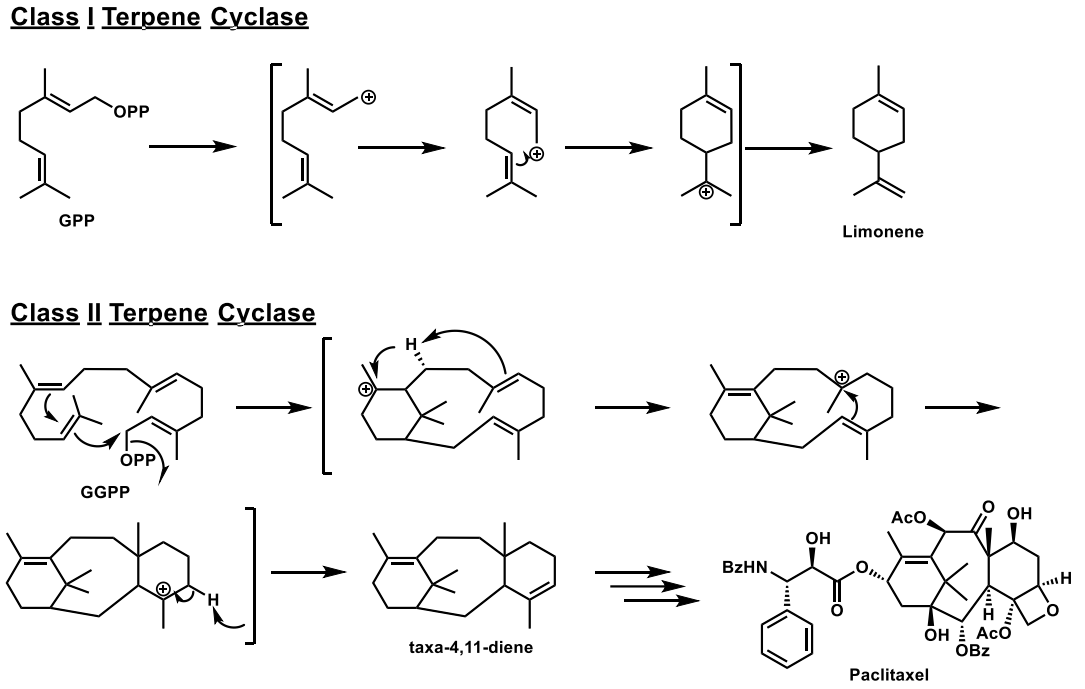


Figure 6. Mechanisms of Terpene Cyclases
Terpene cyclases can be split into two groups: Class I and Class II. Class I terpene cyclases, such as limonene synthase generate a terminal alkene carbocation. Class II terpene cyclases such as the one used in taxol biosynthesis generates a tertiary carbocation on GPP that will be quenched through cyclization of the isoprene unit.

cyclases afford more complicated cyclized terpene products. The positioning of the carbocation ionization varies for individual terpene synthases and determines the cyclization/rearrangement of the isoprene building block. Depending on the type of carbocation formation, terpene cyclases can be grouped into two classes: 1) Class I terpene cyclases which use a trinuclear metal cluster to ionize the isoprene unit by generating an allylic cation and a free pyrophosphate, and 2) Class II terpene cyclases which use a general acid to protonate a terminal alkene to form a tertiary carbocation.[45] Figure 6 shows examples of mechanisms of Class I (limonene synthase)[46] and Class II (taxol synthase)[47] terpene cyclases.

1.2.4 Tailoring Enzymes

Tailoring enzymes are other enzymes that will modify the chemical backbone that is built from the core enzymes (PKSs, NRPSs, and TSs). These include methyltransferases,[48] oxygenases,[49, 50] epimerases,[34] glycosyltransferases,[51] oxidoreductases,[52] acyltransferases,[53] etc. These modifications are the keys that transform the secondary metabolite intermediate generated from the core enzymes to the mature and often bioactive natural product.

Oxygenases are one of the most common tailoring enzymes found in secondary metabolism and are responsible for much of the chemical diversity of natural products.[54] These enzymes often carry out a diverse range of redox reactions including hydroxylations, epoxidations, dehydrogenations, cyclizations, and various rearrangements—often decreasing the lipophilicity of secondary metabolites backbones.[55] Recent discoveries have found several multifunctional oxygenases that can act iteratively on multiple sites of their substrates.[49, 56] Thus, iterative oxygenases are enzymes that can introduce multiple oxygen atoms from molecular oxygen at different sites on a single substrate.

There are several major classes of oxygenases including cytochrome P450 monooxygenases (P450s), flavin-containing monooxygenases (FMOs), and non-heme, iron- and α-ketoglutarate-dependent dioxygenases. Examples of iterative catalysis are found in each of these classes in fungal natural product biosynthesis. Monooxygenases incorporate one oxygen atom from molecular oxygen (O_2) while dioxygenases can incorporate both oxygen atoms.

1.2.5 Natural Product Gene Clusters

The 1965 Noble Prize was awarded for the discovery of the Lac operon, where it was discovered that related bacterial genes are often grouped together. This allows for prokaryotes to

readily regulate the gene expression of these gene clusters under a single promoter. This gene cluster organization was also was found to hold true for proteins that were found in a common pathway.[58] The organization is believed to allow for evolutionary advantages such as the preservation of these gene clusters (and thus the important pathways) during horizontal gene transfer as well as ease of regulation of the cluster when related genes are colocalized.[59] It was believed that the gene clusters were a feature of prokaryotic genomic organization until 1989, with the study of the *Aspergillus nidulans* L-proline catabolic pathway,[60] wherein it was also realized that eukaryotic pathway genes can also be found to be clustered as well. Fungal gene clusters can be defined as the close linkage of two or more genes that participate in a common metabolic or development pathway. Thus biosynthetic gene clusters (BGCs) became attributed to the common gene organization of many secondary metabolites. This started with the study of aflatoxins and sterigmatocystins, mycotoxins derived from the *Aspergillus* genus in which over 25 genes over a 60 kb locus are responsible for generating these natural products.[61]

With the rise of next generation genomic sequencing, the wide range of genomics data has revealed the presence of many natural products BGCs in bacteria, fungi, and some plant pathways. The BGCs organize the genes required for the synthesis of a natural product in a contiguous fashion. A canonical BGC can be expected to have various features including the presence of core enzymes genes, tailoring genes, transport protein genes, transcriptional regulator genes, and in rare cases a gene encoding a self-resistance enzyme. (Figure 7) Usually there are 1 to 2 core enzymes (as described in sections 1.2.1-1.2.3) and a variable number of

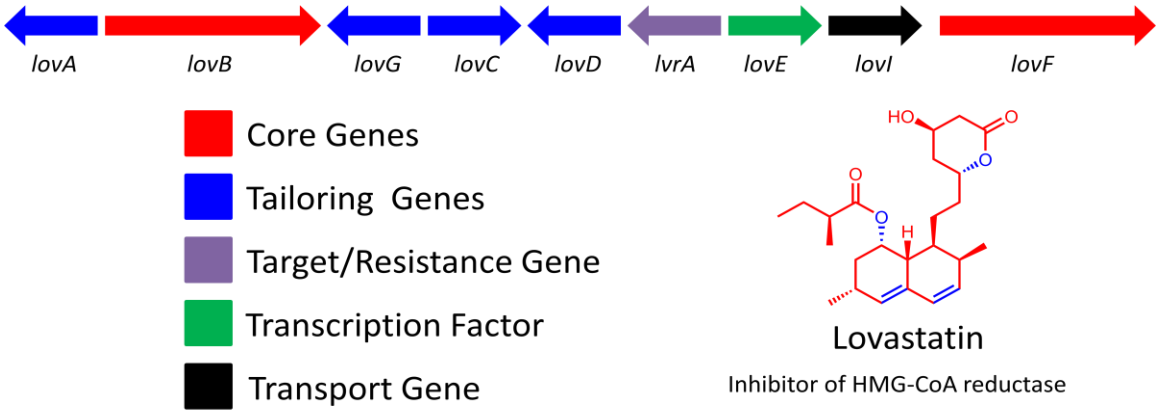


Figure 7. Organization of a Biosynthetic Gene cluster in Fungi
Fungal secondary metabolite genes are usually clustered. These clusters will usually have a core enzyme that makes up the backbone of the natural product along with tailoring enzymes that finish the biosynthesis. In addition, there are regulatory enzymes and transport enzymes. Occasionally there will also be a self-resistant enzyme.

tailoring enzymes (as described in section 1.2.4) that can be found in the gene clusters. These genes are the part of the biosynthetic pathway and provide the essential chemical transformations needed to form the mature natural product. In addition to the biosynthetic enzymes, there are regulatory elements which include transcription factors, transport proteins, and the self-resistance enzymes.

The example in Figure 7 represents the organization of the fungal natural product blockbluster drug, Lovstatin (Mevacor), in *Aspergillus terrerus.* The Lovastatin BGC contains two polyketides, LovB and LovF, which are the core enzymes that synthesize the carbon backbone for the cholesterol lowering drug. In addition to the core enzymes are the other biosynthetic tailoring enzymes such as LovA, LovG, LovC, LovD which correspond to a P450 monoxygenase, thioesterase, trans-acting enoyl reductase, and acyltransferase, respectively. LovA, LovG, and LovC work together on the nonaketide product synthesized by LovB to generate the Monacolin J intermediate of the pathway. Monacolin J is then linked by LovD with

the diketide product synthesized by LovF to furnish the final natural product, Lovastatin.[62, 63] In this gene cluster, LovE is a local transcription factor that upregulates the transcription of all the genes in this cluster when overexpressed.[64] LovI falls under the Major Facilitator Superfamily, a protein family of efflux pumps designed to remove the Lovastatin product from the cell. LvrA corresponds to the self-resistant enzyme, which will be described in the following section.

With more and more information about BGCs being understood from elucidating additional examples of natural product biosyntheses, the ability to search for the BGCs of natural products also has been an expanding field. **Genome mining** for BGCs of natural products has thus also become a newer method of natural product discovery.

1.2.6 Self-Resistant Enzymes

Natural products have a wide variety of bioactivities, including antibiotic, antifungal, herbicidal, and mycotoxic. In the evolutionary landscape of these microbial producers, natural products often serve as methods of chemical warfare to gain an advantage over competitor organisms. The toxic nature of many of these natural products often stems from their ability to inhibit key metabolic processes in competitor organisms. However, the production of these natural product inhibitors is a double-edged sword in that these natural products can affect the host organism as well. To deter these self-harming effects, organisms that produce toxic natural product employ a myriad of techniques including efflux pumps, chemical modification, and the use of self-resistant enzymes. Efflux pumps, such as LovI described in Figure 7 will shuttle the toxic product outside of the cell. Chemical modifications use specialized enzymes that will detoxify the natural product. In the case of the antibiotic chloramphenicol, the natural producer *Streptomyces venezuelae* utilizes a hydrolase that will detoxify chloramphenicol through

acetylation of the C-3 hydroxyl group when the production of the compound reaches high concentrations.[65, 66]

The use of self-resistant enzymes is another technique that fungi and other microbes have evolved to gain resistance toward the toxic natural product they produce. These organisms have evolved targets (typically proteins) that are modified versions of the target the natural product inhibits. Modified protein targets usually contain various mutated residues that impede the natural product inhibitor from binding the enzymes, but still allow for the enzyme to have its original function. These self-resistant enzymes thus serve as an elegant solution for the self-resistance of toxic natural product producers. Furthermore, the studies of the genetic organization of these self-resistant organisms have also shown that the self-resistant enzyme is usually an extra copy of the target. The original housekeeping copy of the target, sensitive to the inhibitor but usually retains higher catalytic efficiency, can be found separate from the self-resistant copy. The resistant copy, on the other hand, can often be found colocalized with the BGC of the natural product inhibitor. It is thus believed that the organism can utilize the optimized housekeeping enzyme most of the time and switch over to the self-resistant copy when it is producing high amounts of the toxic natural product compound.[67-69]

In the case of the Lovastatin BGC shown in Figure 7, the self-resistant copy of the target is encoded by LvrA. Lovastatin has been found to be an inhibitor of the mevalonate pathway intermediate, HMG-CoA reductase. LvrA is mutated version of the HMG-CoA reductase that does not bind the Lovastatin inhibitor but is still catalytically active. The producer strain *Aspergillus terreus* also contains the housekeeping version of HMG-CoA reductase separate from the Lovastatin BGC in its genome, giving it two copies of the HMG-CoA reductase while most fungi only have one copy.[70, 71]

The presence of these self-resistant enzymes in organisms that produce bioactive natural products thus provide a powerful tool in terms of genome mining. By searching through genomes for self-resistant enzymes, we can potentially find natural products of desired bioactivity. **Targeted genome mining** thus utilizes the self-resistant enzyme phenomenon as a method of discovering novel natural products with activity specific toward the queried self-resistant enzyme.

2. The Sterol Biosynthetic Pathway

There are many drug targets of interest that attract medicinal chemists and natural products researchers to design/find compound inhibitors for these targets. One pathway that contains many drug targets of interests is the cholesterol pathway. High cholesterol levels have been linked with coronary heart disease, stroke, and diabetes. This is especially prevalent in the United States as cardiovascular disease remains the leading cause of death in the United States.[10, 72] The cholesterol pathway, in consequence, is a metabolic pathway of interest for the development of drugs to treat hypercholesterolemia.

The sterol pathway, in addition to having valuable targets for cholesterol lowering drugs, also is an attractive area for antifungal agents. Antifungal drugs are in huge demand, with a market totaling $11.3 billion in 2017.[73] Fungal infections are mainly prevalent with immunocompromised patients, often leading to high mortality rates despite the use of current antifungal treatments.[74] Especially with the rise of antibiotic and antifungal drug resistance among fungi, the need for novel antifungal drugs with new modes of action is a growing

concern. Many antifungal drugs work through inhibition of the fungal sterol pathway that synthesizes ergosterol, a metabolite vital for fungal cells.[75] Thus, targeting sterol synthesis is a potentially fruitful strategy for the search for both cholesterol lowering and antifungal drugs.

2.1 Variations in the Sterol Pathway across Kingdoms

Sterol compounds can be found in all eukaryotes and some prokaryotes. Within eukaryotes, sterol compounds are used in a myriad of ways by different phyla. Fungi use ergosterol as a vital structural component of their cell membrane and in cell cycle regulation,[76, 77] while plants and animals use phytosterols and cholesterol, respectively, as not only structural components but also mainly as precursors to developmental hormones and vitamins.[78-81] Though it is similar to the cholesterol and phytosterol pathways found in animals and plants, the biosynthesis of ergosterol is unique for fungi and essential as it regulates membrane fluidity and the regulation of many cell cycle processes. On the other hand, animals can uptake cholesterol through their diet. Consequently targeting the ergosterol pathway can inhibit fungi growth with minimal effect on plants and animals.[82]

Sterols are cyclized triterpenoids products that all contain a tetracyclic frame. This frame is generated through cyclization of the linear C_{30} squalene from the mevalonate pathway.[83] Different sterol molecules vary primarily in the carbon side chain, usually with varying degrees of unsaturation and substitution. There have been over 250 sterol molecules found through chemical screens of prokaryotes and eukaryotes. The major types of sterols in different kingdoms of life can be seen in Figure 8. Animals have cholesterol as the majority of their sterol content. The C_{27} cholesterol structure contains a fully saturated side chain and an unsaturated ring system. Fungi primarily have ergosterol, which is structurally very similar to cholesterol, with a degree of unsaturation across the C22 and C23 carbons as well as an additional methyl group on the C24

position.[77, 83] Phytosterols have many variants including sitosterol, stigmasterol, and campesterol, which are the most copious in the plant kingdom.[84] Plants sterols usually have highly branched side chains compared to their zoosterol counterparts and are much more varied, tallying more than 100 different discovered sterols.[85] This variation can also be seen in the gene organization of sterol pathway genes. While animals will have only one copy of cholesterol pathway genes, plants and fungi have many more sterol genes, which result in the branching of the sterol biosynthetic pathway in different plants and fungi.[77, 84] More recently, studies on protozoa have revealed that these organisms also contain sterol content that can be both biosynthesized or taken from their environments. Trypanosomatid parasites contain ergostane and stigmastane as the major components of protozoan sterol content.[80, 86]

2.2 The Steps Involved in Sterol Metabolism

Sterol biosynthesis is a long pathway that takes many steps to reach ergosterol, cholesterol, or phytosterols. The pathway can be divided roughly into several main sections: 1) the initial mevalonate pathway that generates the isoprenoid precursor squalene, 2) the oxidation and cyclization of the linear squalene to generate the sterol intermediate lanosterol, and 3) the successive steps to process the lanosterol tetracylic core that finish the biosynthesis of various sterol products, depending on the organism.[83, 87]

The early steps of the sterol metabolism involve the conversion of acetyl-CoA to isoprenoid products through the mevalonate pathway. Acetyl-CoA is first condensed with another unit of acetyl-CoA to form acetoacetyl-CoA with the use of acetoacetyl-CoA thiolase. Hydroxymethylglutaryl-CoA (HMG-CoA) synthase then condenses another unit of acetyl-CoA to acetoacetyl-CoA to form HMG-CoA. HMG-CoA is then reduced by the highly regulated HMG-CoA reductase to form mevalonate. Mevalonate is then used for constructing isoprenoid

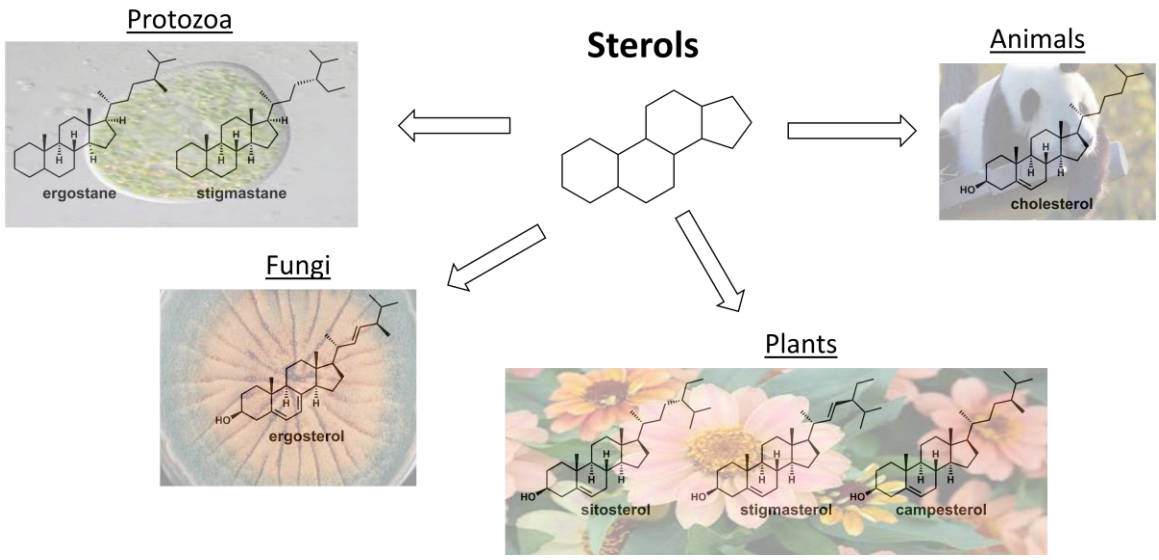


Figure 8. Distribution of Sterols in Life
Sterol compounds are diverse and can be found in many different forms of life. The major sterol compounds found in each kingdom are shown.

building blocks that can be either shuttled toward making terpenoid primary and secondary metabolites or directed toward being synthesized for squalene. Mevalonate requires two kinases and a decarboxylase to generate isopentyl-pyrophosphate, the minimum C_5 isoprenoid unit. The steps from isopentyl-pyrophosphate to squalene can be seen in Figure 3.[88-90]

The next steps of the sterol pathway serve to cyclize the linear C_{30} squalene skeleton into the tetracyclic lanosterol that serves as the branching point for the synthesis of different sterols in organisms. Squalene is cyclized through a two-step process, starting with 2,3 epoxidation of the linear chain through the first oxidase in the pathway, squalene monooxygenase. This step is often considered to be one of the rate limiting steps of the pathway.[83, 91] Next, lanosterol synthase takes the 2,3 oxidosqualene product, generating carbocation intermediates to successfully perform a cascade of cyclizations to form the tetracyclic ring system of sterol compounds. Lanosterol is

then

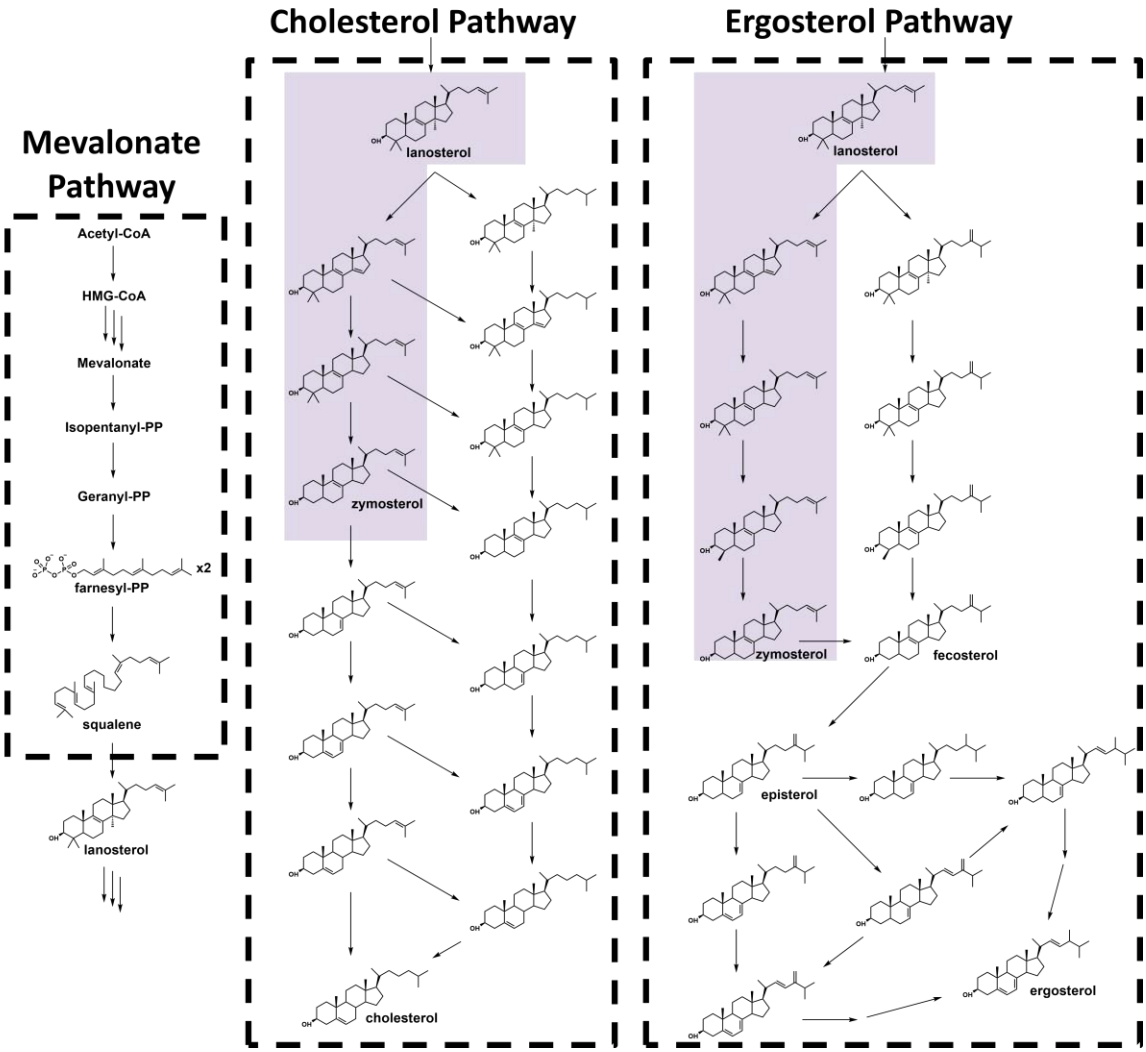


Figure 9. The Sterol Biosynthetic Pathway
The biosynthesis of sterols is a multi-step pathway that is highly regulated. The pathway can be split into three major parts: 1) The early stage steps during the Mevalonate pathway. Here acetyl-CoA is metabolized all the way to isoprenoid precursors and the C30 linear squalene. 2) The formation of the sterol tetracylic backbone through the cyclization of squalene into lanosterol. 3) The late stage pathway stpes that take lanosterol to the final sterol drugs. In mammals, cholesterol is the final product while in fungi ergosterol is the final sterol product. The steps that lead up to zymosterol are conserved between the cholesterol and ergosterol pathway.

converted to various sterols, usually starting with the demethylation of the C14 carbon with the highly conserved Cyp51 lanosterol 14 α-demethylase enzyme.[92]

These late steps of sterol synthesis are divergent in eukaryotic organisms. In mammals, cholesterol in synthesized from lanosterol in a 19 step sequence, executed through 9 different enzymes.[93, 94] These steps work to saturate the side chain, remove the C14 methyl and C4 gem-dimethyl groups, and desaturate the ring system. There is believed to be two major pathways to get to cholesterol, with the timing of the modifications on the side chain dictating how the pathway proceeds.[93] These late step pathways have also been shown to be quite important in regulation of sterol synthesis. Mutations in these enzymes cause genetic sterol disorders, such as Smith-Lemli-Optiz syndrome.[93, 95] In fungi, lanosterol proceeds to ergosterol in a multi-step pathway that differs depending on the organism. The ergosterol pathways of *Saccharomyces cerevesiae*[77] is very well characterized and the sterol pathways of other fungi such as *Aspergillus fumigatus*[76] have also been elucidated. The ergosterol pathway and cholesterol pathways share the steps leading up to the biosynthesis of zymosterol. From there, the pathways diverge to offer different side chains functionality and degrees of saturation on the tetracyclic ring core (Figure 9). The exact mechanisms of these steps are still being studied and divergent pathways from what is understood so far have also been found in different organisms, making the study of sterol pathways an ongoing effort.

2.3 Inhibitors of the Sterol Pathway

Inhibitors of the sterol pathway are valuable pharmaceutical drug leads as they can have anti-cholesterol effects or anti-fungal effects. Some of the largest blockbluster drugs developed with these bioactivities include the cholesterol lowering statin medication and the azole antifungal drugs. Inhibitors of these sterol pathways have also additionally been found to have

anti-tumor activities as well. This is due to the correlation between tumor cells and high cholesterol levels, as tumor progression is highly dependent on cell division which requires high cholesterol uptake.[96] Many sterol pathway inhibitors have been discovered and developed, resulting in a myriad of synthetic, semi-synthetic, and natural product drugs with different modes of action and biological activities for inhibition on various steps of the pathway. In general, inhibitors of early steps in the pathway have better cholesterol lowering activity while later stage inhibitors are more specific toward ergosterol biosynthesis and act as better antifungal agents (Figure 10).

2.3.1 Inhibitors of the Early Mevalonate Pathway

As described in section 2.2, the sterol pathway can be divided into several main parts, with the mevalonate pathway being the initial steps of the pathway. In these early steps, one of the most important conversions is the reduction of the C5 ketone of HMG-CoA to yield mevalonate, done by HMG-CoA reductase. This key step is the rate limiting step of the entire cholesterol biosynthesis pathway, regulated through a negative feedback loop.[97, 98] High levels of cholesterol and other sterol pathway intermediates will lead to a decrease in transcription of HMG-CoA reductase.[99]

The statins are cholesterol lowering drugs that were first discovered in the 1970's as natural products from filamentous fungi. Compactin was first discovered from *Pencillium citrinum* and showed antimicrobial activity during the first screens.[10, 100] The compound was soon shown to also lower plasma cholesterol levels, offering potential as use for hypercholesterolaemia. In 1987, lovastatin was first approved by the FDA for cholesterol lowering medication and had very limited side effects, and was able to achieve a 40% reduction

in LDL levels with a daily dosage of 80 mg.[101] These two natural products drove the development of statin drugs.

Simvastatin was the next approved statin, a semisynthetic version of lovastatin that feature an additional methyl group on the side chain in 1988. Pravastatin was discovered in 1991 as another semisynthetic drug. In addition, purely synthetic statins were developed as well, including fluvastatin in 1994, atorvastatin in 1997, cerivastatin in 1998, and rousvastatin in 2003.[100] The natural products and semi-synthetic analogs contain decalin cores that anchor the monacolin structure. The synthetic analogs on the other hand differ in the decalin core portion of the molecule, containing additional chemical moieties such as the cyclopropyl group in pitavastatin to better interact in the binding pocket of HMG-CoA reductase.[96] The lactone group is structurally unchanged in both the natural and synthetic statins. This lactone group when opened through hydrolysis mimics the structure of the HMG-CoA substrate. This lactone thus serves as the molecular warhead that allows for the inhibitory activity of the statins.[102] As cardiovascular heart disease remains one of the leading causes of death in western countries, these drugs remain important pharmaceuticals.

In addition to the HMG-CoA reductase inhibitors, there are also other compounds that have been found to inhibit the first few steps of the mevalonate pathway. The butyrolactols are butyryl lactones with a hydroxyalkyl side chain found to suppress the growth of various fungal species.[103] They are structurally similar to Antibitoic F-244 (hymeglusin) an antimicrobial that inhibits both bacterial and fungal growth. The compounds showed activity toward lowering mevalonate levels and thus the mode of action was explored. F-244 was shown to bind to HMG-CoA synthase, offering an inhibitor of the step before HMG-CoA reductase.[104]

2.3.2 Inhibitors of the Late Mevalonate Pathway

Early mevalonate pathway inhibitors have the disadvantage in that they greatly decrease the production of the isoprenoid precursors (isopentyl-PP, geranyl-PP, farnesyl-PP) that are

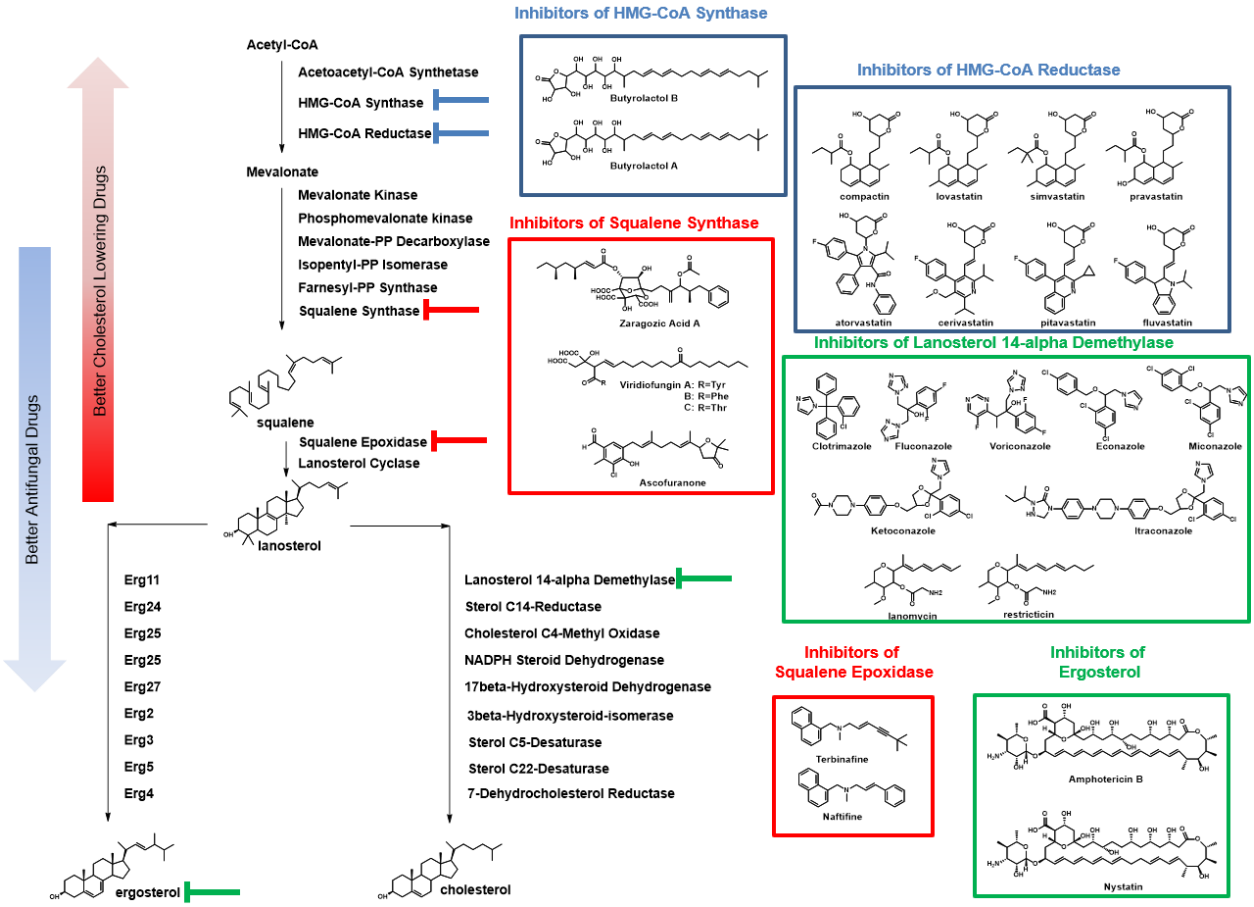


Figure 10. Inhibitors of the Sterol Pathway
Many compounds have been discovered and developed for the purposes of inhibiting the sterol pathway. Inhibitors of the early steps of the pathway are boxed in blue and have cholesterol lowering activity. In the middle steps of the pathway, there have been many natural products discovered that are used for a mix of cholesterol lowering agents and antifungal. Finally toward the late stage steps of the sterol pathways are inhibitors that serve as very efficient antifungal medications.

important toward many biological functions. Late stage cholesterol pathway inhibitors also have the issue of accumulating toxic rigid sterol cyclic intermediates. Thus the steps in the late stages

of the mevalonate pathway leading up to the synthesis of the first sterol cyclic product lanosterol hits a sweet spot for ideal inhibitors of sterol metabolism.

Squalene synthase is the step that joins two units of farnseyl-PP to form the linear precursor to lanosterol, squalene. The squalestatins, also known as zaragozic acids were discovered in the early 1990's in various different filamentous fungi. Over 20 different squalestatins and zaragozic acids have been discovered, with variations mainly among the side chains. The squalestatins all contain the same highly oxygenated 2,8-dioxabicyclo-[3.2.1]octane-3,4,5-tricarboxylic acid core. The squalestatins are highly potent and selective inhibitors of squalene synthase. This inhibitory activity gives them biological applications such as cholesterol lowering, antifungal agents, and farnesyl transferase inhibitors.[105] More about the squalestatins will be discussed in section 4.

Other than the squalestatins, the structurally similar Viridiofungins are also another group of natural products discovered to have squalene synthase inhibitory effects. They displayed strong antifungal activity, but no antibacterial activity. Structurally the viridofungins contain a saturated lipid tail with a dicarboxylic acid head, which is also attached with an amino acid. These amino alkyl citrate compounds show micromolar inhibitory activity against squalene synthase though they are much less potent than the squalestatins.[106, 107] Ascofuranone, an antibiotic and anti-tumor drug, was first discovered through hypolipidemic screens and also has squalene synthase inhibitory activity.[108]

Squalene epoxidase is the step following squalene synthase and the first of two concerted steps to synthesize the cyclized sterol intermediate lanosterol. The allylamines are squalene epoxidase inhibitors that act as potent antifungal agents. Naftifine is an allylamine drug serendipitiously discovered through an accidental synthesis reaction by Sandoz. It was found to

have both antibacterial and antifungal effects. Its antifungal effects are pronounced, effectively having fungistatic effects on dermatrophytes by inhibiting ergosterol biosynthesis. Terbinafine is the synthetic analog of naftifine, which is more potent and able to be administered orally as well as topically in contrast to naftifine which can only be administered topically.[109-112]

2.3.3 Inhibitors of the Late Sterol Pathways

The late stages of the sterol pathways are divergent in different organisms. Therefore, targeting late stages of the ergosterol pathway are a desirable way to develop antifungal agents. One important step in the late stage pathway is catalyzed by Cyp51, the lanosterol 14-α demethylase p450 enzyme. The Cyp51 family is conserved across all life and is believed to be the ancestor of all p450s. The inhibition of this target has potential to have both cholesterol lowering effects and antifungal activity, although the current major inhibitors of this target are primarily used for their antifungal activities.[113]

Azole drugs are the most broadly used Cyp51 inhibitors currently on the market. They are able to inhibit the p450 enzyme through binding to the coordinated heme iron competitively to block out the natural substrate. The azole antifungals are all synthetic drugs that were designed to specifically inhibit Cyp51. Azoles have two classes, the imidazoles and the triazoles. The imidazoles contain two nitrogen ring systems while the triazoles have three nitrogen rings systems. The nitrogens on these rings serve as the chemical warheads that bind to the heme iron complex of Cyp51 to block its activity. The imidazoles include ketoconazole, clotrimazole, econazole, and miconazole. The triazoles include fluconazole and voriconazole. The triazoles are newer developed drugs that have a better pharmokinetic results and generally better efficacy.[114]

In addition to the synthetically developed azoles, there are also natural products derived from filamentous fungi that have also been found to inhibit Cyp51. These include lanomycin and

restricticin, which contain a glycine ester moiety that is required for their inhibitory activity.[96] These natural products will be discussed further in Section 5.

Finally, there are inhibitors that bind directly to the final product of the sterol pathway in fungi, ergosterol. These are the polyene antifungals, amphotericin B and nystatin. Amphotericin B directly binds to ergosterol, forming aqueous pores in the fungal cell membrane. Both drugs were isolated from bacterial cultures through phenotypic screens during the early 1950's when natural product discovery was still in its nascent form. They are both large macrolide polyenes that are usually glycosylated. These drugs contain both a polyol and polyene portion, corresponding to having highly hydrophilic and hydrophobic portions.[115] These antimycotic drugs have high activity and are still widely used for antifungal treatments today.

3. Targeted Genome Mining Method Development

Current technological advances have made it possible to develop a highly efficient pipeline for natural product discovery. Our natural product discovery pipeline uses bioinformatics, synthetic biology, and chemical characterization tools in concert to not only discover new natural products, but also elucidate the biosynthetic pathways that manage their production. Our natural product discovery pipeline consists of four major parts (Figure 11). This pipeline begins with the genomic characterization of fungi usually done through next generation sequencing techniques. Along with the publicly available genome databases from the National Center for Biotechnology Information (NCBI)[116] and Joint Genome Institute (JGI)[117], we have access to over 5000 characterized fungal genomes to search and study from. With this fungal genomics information at our fingertips, there are also various bioinformatic tools available that allow us to predict and find gene clusters of interest. After a gene cluster of interest is identified,

we can begin to study the gene cluster using synthetic biology tools. Specifically, we utilize our robust heterologous expression systems in *Escheria coli, Saccharomyces cerevisae,* and *Aspergillus nidulans.* Then through metabolic analysis using chromatographic instrumentation such as LC-MS or HPLC, we can identify new chemical peaks that correspond to our gene cluster of interest. Chemical characterization through nuclear magnetic resonance (NMR) spectroscopy, X-ray crystallography, mass spectrometry, etc. allows us to determine the chemical structure of the peaks of interest and identify key intermediates to help us build the biosynthetic pathway leading up to the final natural product. This section will cover technological developments in our natural product discovery pipeline aimed to specifically tailor our efforts to find natural products through targeted genome mining.

3.1 Development of *in silico* Targeted Genome Mining

Some secondary metabolite gene clusters have been elucidated, but these BGCs still remain an enigmatic space of genomics, especially in fungi and plants. In contrast to the well conserved primary metabolite pathways that are very well understood, secondary metabolite gene clusters can vary from organism to organism even within members of the same species. Next generation sequencing has given us a wealth of genetic information on many of these mycotic genomes, but there remains the need to properly annotate and assign function to the DNA sequence. Proper identification of BGCs and their key components are the crucial steps to discovering and studying natural product biosynthesis.

3.1.1 Bioinformatics tools for Gene Annotation

There are many publicly available bioinformatics tools that have been developed toward analyzing natural product gene clusters. To correctly identify genes in a newly sequenced genome, comparative genomics with known gene sequences can help infer the boundaries and

functions of uncharacterized genes. One popular method is the use of Basic Local Alignment Search Tool (BLAST). BLAST takes advantage of characterized genome databases from NCBI to compare queried sequences. By looking at sequence similarity of the query to known

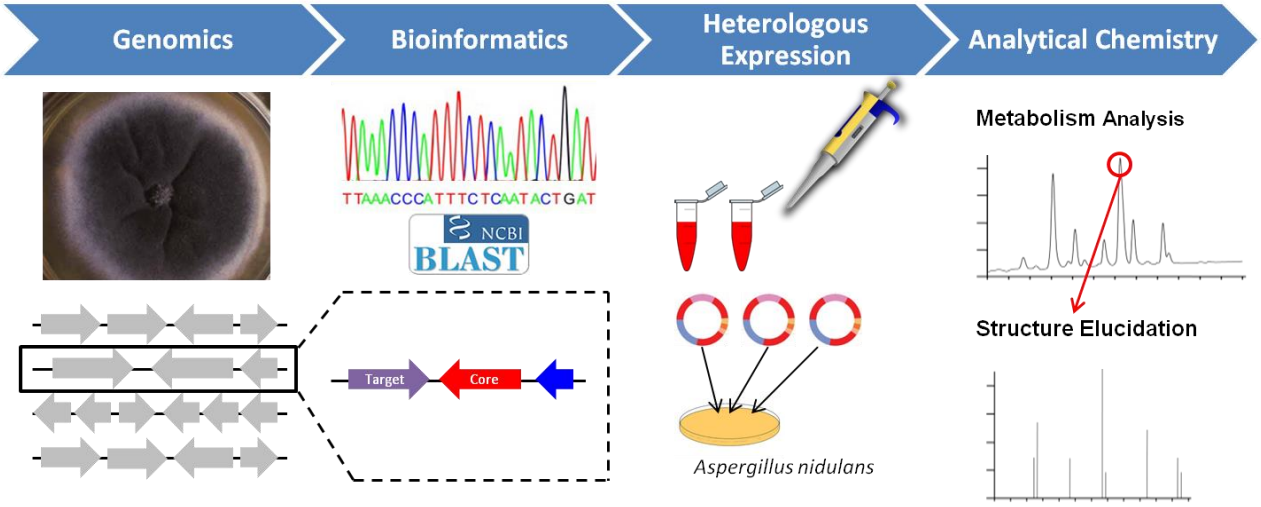


Figure 11. Natural Products Discovery Pipeline
Our natural products discovery pipeline utilizes a combination of genomics, bioinformatics, synthetic biology, and analyticial chemistry techinques to identify new natural product compounds of interest for study.

sequences, BLAST can draw conclusions on the closest homologs to the query and give function and boundary. BLAST utilizes local alignments that check for functional domains in the protein sequences that are often conserved through members of the same protein family. BLAST also can compare protein and nucleotide sequences in different combinations using multiple algorithms such as BLASTN (nucleotide vs nucleotide), BLASTP (protein vs protein), BLASTX (nucleotide query vs protein database), and TBLASTN (protein sequence vs nucleotide database. These results are outputted and calculated for an expect value that judges the confidence in an alignment match.[118]

In addition to BLAST, another gene annotation algorithm that can be used is AUGUSTUS. AUGUSTUS uses a hidden Markov Model (HMM) that defines statistical probabilities for different DNA types such as introns, exons, intergenic regions, etc. This puts more stringent constraints on the search and prediction patterns the algorithm uses for more accurate gene predictions. This allows the user to accurately predict gene boundaries and the intron positions by comparing to a reference genome.[119, 120]

There are a few algorithms developed that can predict entire secondary metabolite gene clusters. Two of these are antibiotics and Secondary Metabolite Analysis Shell (antiSMASH) and Secondary Metabolite Unknown Regions Finder (SMURF). antiSMASH can take genomic FASTA (the most basic DNA/protein file format) files to be processed for gene cluster identification. It is able to identify PKS and NRPS genes readily through a combination of domain analysis, HMM prediction, and BLAST analysis. The outputs are in a user-friendly interface where the gene cluster can be viewed and navigated through.[121] SMURF is another algorithm that evaluates gene clusters in a similar way, scoring the proximity of backbone core genes with the tailoring genes found around them.[122] To avoid dereplication of gene clusters, the online database of characterized gene clusters, Minimum Information about a Biosynthetic Gene Cluster (MIBiG) is available for comparison with these algorithms.[123]

More specified genome mining algorithms have been also recently developed. The Antibiotic Resistance Target Seeker (ARTS) is an engine that specifically searches in bacterial genomes for antibiotic resistant enzymes to direct toward clusters with possible new drug leads.[124]

3.1.2 Development of Targeted Genome mining Information Finder (TGIF)

Currently, genome mining is mainly done through manual methods; identifying one or two query genes of interest to use a search against publically available genomes using software listed in the previous section to screen through possible gene clusters containing the queries of interest. However, with the growing knowledge of gene cluster organization through elucidated biosynthetic pathways and the identification of self-resistance enzymes present in these BGCs,

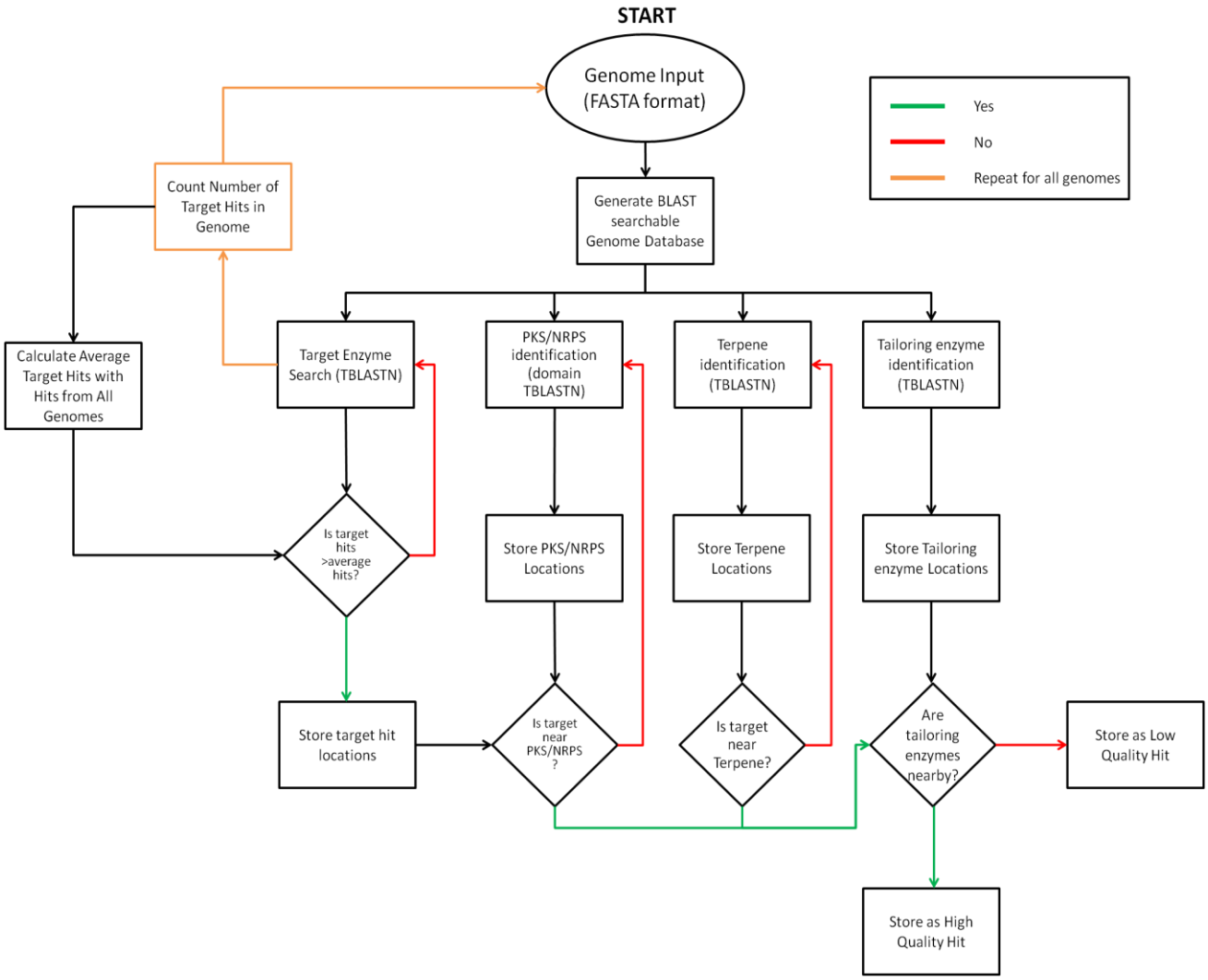


Figure 12. TGIF algorith flow diagram
The steps the TGIF algorithm takes to identify gene clusters of interest with self-resistance enzymes.

there is the potential to search for BGCs with desired bioactivity. Targeted Genome mining Information Finder (TGIF) aims to leverage the information currently known about self-resistant enzymes to identify novel BGCs in an automated fashion.

MATLAB® is a computing environment and programming language that is able to execute algorithms and data presentation. One of the major advantages of MATLAB® is its wealth of toolboxes that allow users to adapt their algorithms for specific fields of study. The Bioinformatics Toolbox™ includes the ability to use BLAST to analyze FASTA format DNA sequences, thus giving us a good foundation to develop our targeted genome mining engine.

The TGIF algorithm searches for gene clusters that contain possible self-resistance enzymes that imply the biological activity of that particular cluster. A flow diagram outlining the steps is shown in Figure 12. For all the searches done, the TBLASTN algorithm is used for alignment searches. TBLASTN compares a translated protein sequence to a nucleotide database, allowing for a broader search due to codon degeneracy. The TGIF algorithm utilizes various scripts to run through the commands necessary to perform the targeted genome mining search. All scripts can be found in the Appendix.

The first step is to build searchable databases for BLAST using the makeblastdb command. This is iterated through for all genomes in the custom database and done using the "mbmadedb.m" script.

The next step is to generate target hits in all the genomes using the "mbblast.m" script. The protein sequences of these targets are inputted to the MATLAB algorithm. Self-resistant enzymes that were described in Section 1.2.6 have been shown to be additional copies of the housekeeping enzyme the natural product inhibits. Thus, when searching for these self-

resistant enzymes, we should be looking for genomes that contain an above average number of the target enzyme. In addition, these self-resistant enzymes are usually mutated versions of the housekeeping enzyme. Therefore, we want to find BLAST hits of the enzymes that can give us a sufficiently high identity to qualify as a homolog, but also not too much homology to be considered a typical housekeeping enzyme. To do this, we used MATLAB® to iteratively BLAST through all the genomes in our customized database and count the number of hits in each genome, where a hit is counted when the expect value is about 1E-50, the identity is at least 30%, the coverage is at least 20% of the query, and a BLAST score of at least 50. These parameters are adjustable and can be customized for different targets. Storage of the results, removal of duplicate hits, and parsing of the output data is done through "targethitchecks.m".

The next step is to identify the core enzymes to determine secondary metabolite clusters. This is done with the "colocalblast_mb.m". As PKS and NRPS core enzymes are multi-domain enzymes, they are identified by using BLAST with reference domain sequences of the PKS and NRPS and searching for colocalization of these domains. As discussed in Section 1.2.3, there are various forms of terpene cyclase enzymes including monoterpenes, sesquiterpenes, triterpenes diterpenes, and tetraterpenes. Reference examples for these enzyme sequences are used to find terpenoid gene clusters. Tailoring enzymes are also identified by BLAST using reference sequences. Currently, TGIF searches for methyltransferases, transcription factors, p450 monoxygenases, and flavin dependent monoxygenases. These are performed with the "auxgeneblast.m" script. Results for PKS, NRPS, Terpenes, and tailoring genes hits are stored, duplicates are removed, and output results are parsed using the "secondmetcheck_PKS_test.m", "secondmetcheck_NRPS.m", "secondmetcheck_terpene.m", and "auxgenecheck.m" scripts, respectively.

The final step is to use all the data generated through the BLAST function and organize it to determine gene clusters of interest. The “targetclusterfindv3.m” script first uses the target lists generated through “targethticheck.m” and compares these with the core enzymes hits to determine colocalization of the target and core enzymes. This is done by checking the base pair (bp) distance between the target and core enzymes. As the average fungal cluster size is around ~20k bp, this is the default distance allowed to be considered a cluster in the algorithm. This value however can be adjusted for customization. If the target enzyme and core enzyme are colocalized, the algorithm will then search for any tailoring enzymes around them. If there are tailoring enzymes, this result is stored as a high-quality hit. In contrast, if there are no tailoring enzymes, the result is stored as a low-quality hit. For all the positive cluster hits, an output is generated that includes information about the statistical parameters, the genome the cluster is found in, the specific location of the cluster, the type of core enzyme cluster, and the tailoring genes found around the cluster. Another novel feature is the ability to search for specific residue mutations of the target. If known site mutations of a specific target enzyme are known to confer resistance, the residue position can be checked in the target cluster hits to have higher confidence of a true positive if the point mutation exists in the results.

The results file allows for the quick screening of many genomes to identify gene clusters that contain self-resistance enzymes. After identification of a gene cluster of interest, the gene cluster is fully annotated and checked with literature and the MiBIG database to determine if the cluster has been characterized. If not characterized, the cluster is then a strong candidate for further analysis in the genome mining pipeline (Figure 11).

3.2 Development of Synthetic Biology Tools for Targeted Genome Mining

After identification of a target gene cluster of interest, the next step is to study the genes in the cluster using synthetic biology techniques. There are two general methods to study gene clusters and the molecules that are made from them. The first is using a top-down approach which involves deconstructing the molecule from the final product. This is usually done through the use of systematic genetic knockouts in the producing organism to identify intermediates of the pathway by going backwards from the final molecule.[125, 126] The second method is to use a bottom-up approach to build up to the final molecule gene by gene. This is usually done through heterologous expression of the cluster, starting by finding the combinations of genes to generate early pathway intermediates.[127]

3.2.1 Heterologous Expression in *Aspergillus nidulans*

To elucidate the biosynthesis of target gene clusters of interest, we use a heterologous platform in *Aspergillus nidulans* to express these genes in a foreign host. Heterologous expression is often used in lower order organisms such as *Escherichia coli*[128, 129] or *Saccharomyces cerevisiae*[130, 131] to express foreign genes due to the expansive literature regarding their genetic and metabolic profiles. Working with a fungal host such as *A. nidulans*, however, allows for more complex genes to be expressed more readily as the phylogeny of fungal strains would be closer in relation. Using a fungal host also allows us to use express genes that can be accurately spliced by the host. In lower organisms, RNA transcripts must be used to accurately express the proteins, which would be an issue especially for silent gene clusters.

The *Aspergillus nidulans* A1145 strain is a genetically modified strain auxotrophic for uracil, riboflavin, and pyridoxine. These auxotrophic markers can then be utilized to episomally express foreign genes across three plasmids: pYTU, pYTR, and pYTP. With access to up to four

different types of constitutive fungal promoters to use, each plasmid can handle four genes, giving access to heterologous expression of up to 12 genes from the cluster (Figure 13). The *A. nidulans* platform also utilizes the AMA1 fungal plasmid replicator gene, which improves extrachromosomal replication and transformation efficiency by up to 2000 fold[132]. In addition to the AMA1 gene, the plasmids also contain the yeast 2µ origin of replication and the bacterial *E. coli* ColE1 origin of replication, allowing these plasmids to serve as shuttle vectors between the three species for assembly, amplification and expression. These properties make the *A. nidulans* platform an effective means for heterologous expression of fungal gene clusters.

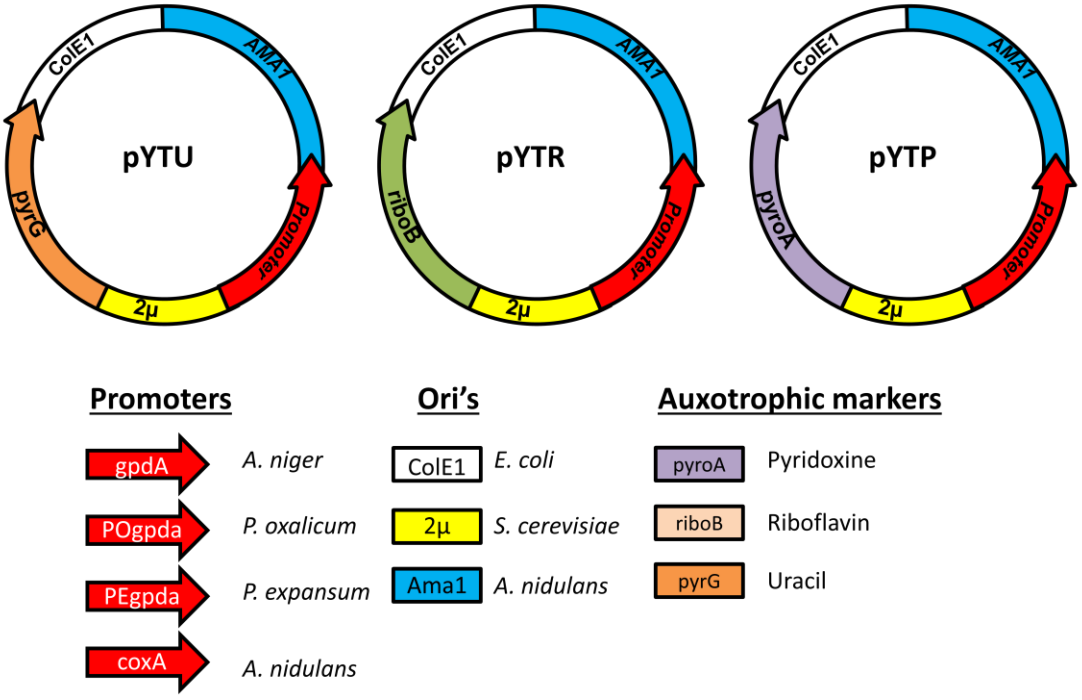


Figure 13. The *A. nidulans* heterologous platform
The episomally based heterologous expression platform in *A. nidulans* utilizes three shuttle vectors that are capable of expressing multiple genes from BGCs for study.

3.2.2 Engineering of the *Aspergillus nidulans* heterologous expression platform

One of the disadvantages of using a heterologous host in a higher organism such as *A. nidulans* rather than *E. coli* or *S. cerevesiae* is that the metabolic profile of the filamentous fungi strain is quite diverse, causing a high background for metabolite analysis. There are many secondary metabolite gene clusters in filamentous fungi, with up to 71 clusters found in *A. nidulans*. This issue was exemplified when performing heterologous expression of genes from the zaragozic acid A cluster. The plasmid system induced significant production of both sterigmatocystin (ST)[133] and emericellamide (EM),[14] which contributed to a high background TIC (Figure S8). This amalgam of endogenous metabolites, compounded by the unknown product mass made identification of any new product difficult. To expunge this issue, we removed these metabolites through genetic knockouts of ST and EM, facilitated by CRISPR-Cas9 to delete *stcA* and *easA* and yield the strain *A. nidulans* A1145ΔSTΔEM.[134, 135] Since ST and EM are also polyketide products, removing these products would ideally not only clean the metabolic profile, but also increase flux of acetate units to the production of polyketide products rather than to the production of ST and EM.

To perform the gene knockouts of ST and EM, the CRISPR/Cas9 vector pFC330 (containing the pyrG auxotrophic marker) was used. Knockout procedures are similar to ones described previously by Nodvig et al.[134] Briefly, the protospacer within the sgRNA was determined by CHOPCHOP[135] to minimize the number of promiscuous targets. The sgRNA was amplified from gBlocks (IDT, USA) and ligated to pFC330 upon double digestion with *Bgl*II and *Pac*I. The linear, marker-less knockout cassette was constructed via splicing by overlap extension PCR (SOE-PCR), with 2 kb upstream and downstream of the knockout region respectively. Primers used for the construction of the CRISPR plasmid and knockout cassette are

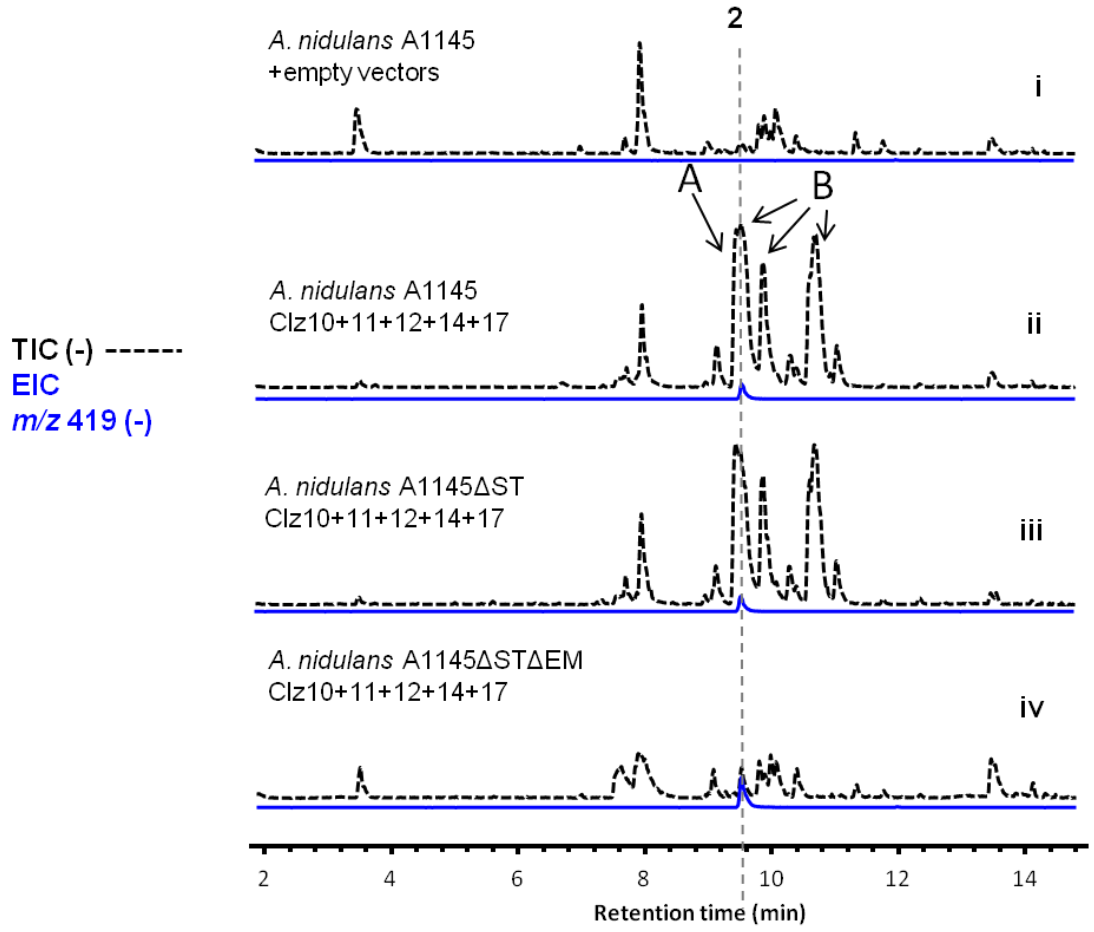


Figure 14. Engineering of *A. nidulans* strains.
[i] *A. nidulans* A1145 with empty vectors as a negative control. [ii-iv] Engineered *A. nidulans* A1145 strains with the cassette to produce the intermediate with a mass of 420. A and B shows location of metabolites signals removed upon deletion of ST and EM genes, respectively. TIC: total ion chromatogram shown in dotted lines to show the overall metabolite profiles. EIC: extracted ion chromatography m/z values filtered for the mass of 420.

listed in Table S5. Both the CRISPR/Cas9 plasmid and linear knockout cassette were co-transformed to *A. nidulans* A1145 with the transformation procedures described below. For negative selection to cure the CRISPR plasmid, solid plates of glucose minimal media (GMM) containing uridine, uracil, pyridoxin, and riboflavin, with 5-fluoroorotic acid (5-FOA) to a final concentration of 1 mg/mL. Colonies were screened for knockout by PCR from the genomic

DNA, and knockout efficiencies were 50% and 30% for sterigmatocystin (ST) and emericellamide (EM), respectively to generate *A. nidulans* A1145ΔST and *A. nidulans* A1145ΔSTΔEM.

Removal of these excess secondary metabolites greatly purged the high background that was viewed with the original A1145 strain, allowing for identification of new compounds introduced through heterologous overexpression, especially of compounds which signals overlay with EM and ST products (Figure 14). The new engineered platform greatly assisted in the identification of key intermediate compounds discussed in the following sections.

4. Identification and Elucidation of the Zaragozic Acid A Biosynthetic Gene Cluster

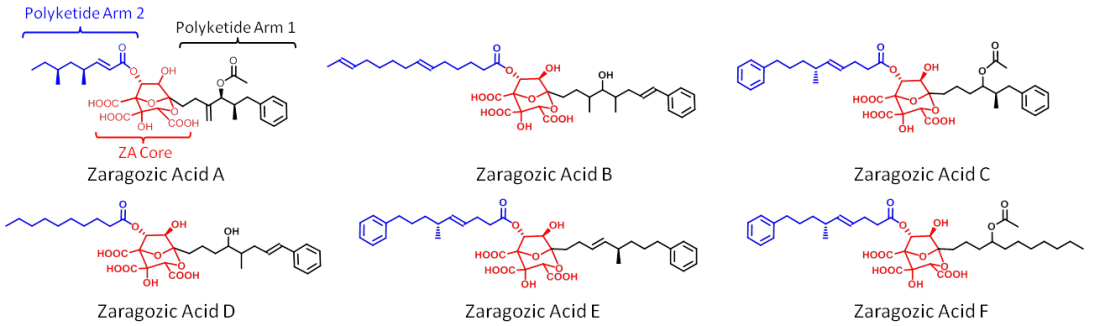
Zaragozic acid (ZA) A (**1**) (also known as squalestatin S1 is a heavily oxidized fungal polyketide that offers potent cholesterol lowering activity.[136] Though various total syntheses of **1** have been reported,[137-148] a complete understanding of its biosynthesis remains elusive. To further study ZA and its biosynthesis, we utilized targeted genome mining to first identify the BGC that produces the molecule. As ZA is a known inhibitor of squalene synthase, we utilized the TGIF algorithm to search through the genome of a known producer of ZA, *Curvularia lunata.* The results showed the presence of two squalene synthases in the genome of the filamentous fungi, one more than the average number of copies in fungi. One of the squalene synthases showed the presence of many secondary metabolite genes in close proximity, indicating a possible candidate cluster for ZA. Here, we utilized an engineered *Aspergillus nidulans* heterologous host to reconstitute the biosynthesis of ZA.

4.1 Introduction to the Zaragozic Acids

Fungal polyketide natural products have been an important source of pharmaceutical drugs due to their wide range of bioactivities.[149] The diverse and complex structural features have also attracted intense research efforts towards understanding the biosynthetic logic.[150-152] The carbon scaffolds of many fungal polyketide natural products, including lovastatin[131] and cytochalasans,[153] are built from the iterative functions of highly reducing polyketide synthases (HRPKSs). These scaffolds are typically oxidatively modified by subsequent downstream tailoring enzymes, such as oxidases and oxygenases, to furnish the mature product.[49, 50, 63, 154] ZA is one of such fungal polyketide natural products, first discovered in 1992, that showed potent cholesterol lowering activity with its picomolar inhibition toward squalene synthase.[136] This molecule's bioactivity was of particular interest due to its target (squalene synthase) being part of the middle stage of the cholesterol pathway. Since there were already effective drug inhibitors of the early stage of the sterol pathway (statins), a drug capable of impeding the cholesterol pathway could have cholesterol lowering effects without obstructing the synthesis of important isoprenoids products (IPP, GPP, FPP) that come after the early stages of the sterol pathway.[155] Inhibitors of the middle cholesterol pathway also would ideally not cause the buildup of rigid toxic cyclized sterol products that accumulate with the use of late stage sterol pathway inhibitors.[83]

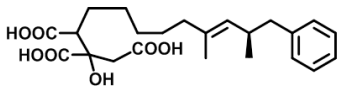
In addition to the noteworthy bioactivity, ZA's have unique chemical features that drew the attention of many synthetic chemists. Members of the ZA family of molecules share a 2,8-dioxobicyclic[3.2.1]octane-3,4,5-tricarboxyclic acid core that is connected to two lipophilic polyketide or fatty acid derived arms (Figure 1).[156] The unique structural features of ZA mimic

presqualene diphosphate, the product of the head-head condensation of farnesyl-diphosphate, and make ZA potent inhibitors of squalene synthase.[157] Labeling studies have shown the

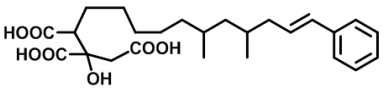


Zaragozic Acid producing Fungi

Organism	Principal Product	Source
Sterile Fungus MF5453	ZA-A	River filtrates
Curvularia lunata var. lunata	ZA-A	Living
Curvularia lunata var. aeria	ZA-A	Living Bark
Exserohilum rostratum	ZA-A	Living Bark
Setosphaeria khartoumensis	ZA-A	Not reported
Drechslera biseptata	ZA-A	Living Bark
Pseudodiplodia sp.	ZA-A	Living Bark
Phoma sp.	ZA-A	Soil
Sporomiella intermedia	ZA-B	Herbivore dung
Leptodontidium elatius	ZA-C, ZA-E, ZA-F	Dead wood and fungi
Amauroascus niger	ZA-D	Soil baited with hair
Libertella sp.	ZA-D	Living Bark
Cladopsorium cladosporioides	ZA-D	Soil



Isolate from MF 5453 extract



Isolate from *S. intermedia* extract

Figure 15. Representative compounds from the zaragozic acids family of polyketides.

tricarboxylic acid core is partially-derived from oxaloacetate, an intermediate found in the citric acid cycle.[155] Cox et al have shown the tetraketide arm (Figure 15) in **1** is synthesized by a HRPKS and enzymatically esterified to the core in the last biosynthetic step.[158] However, formation of the other polyketide arm of **1** is unresolved.

4.2 Results and Discussion

4.2.1 Identification of the *clz* Cluster

We sequenced the genome of the fungal pathogen *Curvularia lunata* (also known as *Cochliobolus lunatus* ATCC 74067), which was previously identified as a producer of **1**.[159] Since

1 is a known inhibitor of squalene synthase, we used the TGIF algorithm with squalene synthase as a query to search for clusters candidates. Searching through the assembled genome showed the presence of two copies of squalene synthase, more than the average number of copies in fungi. This led to the identification of a gene cluster (*clz)* that is likely responsible for the biosynthesis of **1** (Figure 16) that was next to the second non-housekeeping copy of squalene synthase in the filamentous fungi. This cluster is similar to the recently reported squalestatin S1 cluster identified from *Phoma* sp. C2932 and unidentified fungus MF5453.[158] Among the ~20 genes in the cluster (Figure S1), there is a gene that encodes the squalene synthase (Clz20) as a potential resistance enzyme to **1** and two HRPKSs (Clz2*)* and (Clz14). Clz2 is highly homologous to the previously identified squalestatin tetraketide synthase.[160] Also present is the acyltransferase Clz6 that catalyzes transferring of the tetraketide product from Clz2 to the hydroxyl group in the tricarboxylic acid core.[158] Therefore, we designated the *clz* cluster to be responsible for **1** in *C. lunata*.

4.2.2 Elements of the *clz* Biosynthetic Gene Cluster

We focused on identifying the enzymes that assemble the benzyl containing polyketide arm. We reason that the uncharacterized HRPKS, Clz14, should be involved, but the structure of the potential polyketide product is unknown, especially with respect to the extent of reduction at the different C_β-carbons. One anticipated feature of the product is the presence of the benzyl group, which is proposed to be a starter unit for the HRPKS.[105] Although nonacetate starter units, such as propionate, have been observed in priming of fungal HRPKSs,[161] a benzoate unit would represent the most significant departure from the canonical starter unit acetate. The *N*-terminus of Clz14 contains a ~90 residue segment before the KS domain that bears no secondary structure or signal peptide sequences, and is not found in many other fungal PKSs (Figure S2).

Two genes in the *clz* cluster, Clz10 and Clz12, encode phenylalanine ammonia lyase (PAL) and 4-courmarate-CoA ligase, respectively. These two enzymes may be involved in transforming

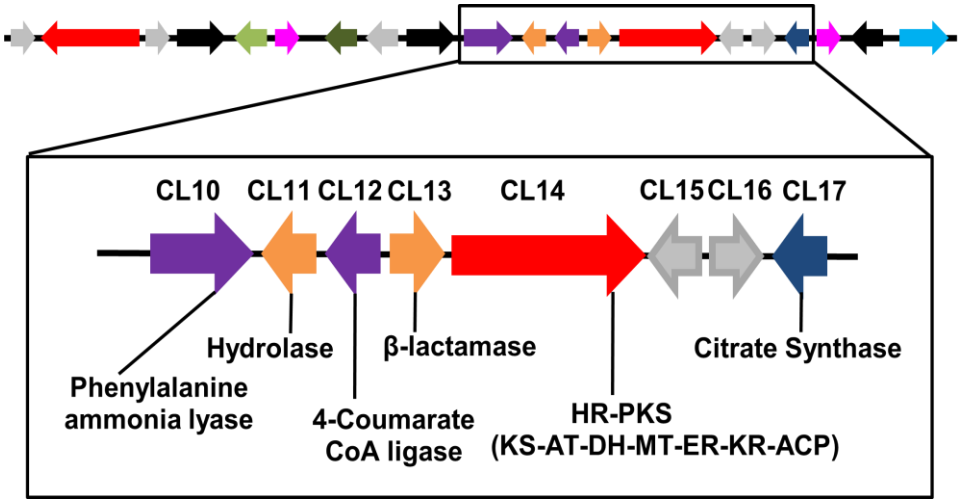


Figure 16. Zaragozic Acid A cluster
Organization of the zaragozic acid A (*clz*) gene cluster found in *C. lunata*. The magnified region contains genes that were hypothesized to be responsible for the biosynthesis of the benzoyl-primed, tricarboxylic acid intermediate.

phenylalanine into benzoyl-CoA, a strategy that is used in the biosynthesis of the enterocin natural products in *Streptomyces.*[162] Also present in the gene cluster is a homolog of citrate synthase, Clz17, which may be involved in connecting the C_α carbons of the polyketide chain and citrate to afford the tricarboxylic acid unit. Homologs of this enzyme are found in gene clusters of nonadride-containing polyketides, and often function in tandem with an alkylcitrate dehydratase to form maleic anhydride.[163-165] No dehydratase homolog is found in the *clz* cluster, consistent with the presence of the tricarboxylic acid moiety in **1**. Instead two potential hydrolytic enzymes, Clz11 and Clz13, which are α/β hydrolase and β-lactamase, respectively, are in close proximity to Clz14, and may participate in product release.

4.2.3 Heterologous expression of the *clz* cluster

To elucidate the function of Clz14, we used an episomally based heterologous system in *A. nidulans* A1145 that is capable of expressing up to 12 genes using 3 plasmids.[166] We first introduced five genes (*clz*10, 11, 12, 14 and 17) and monitored total ion count (TIC) to detect

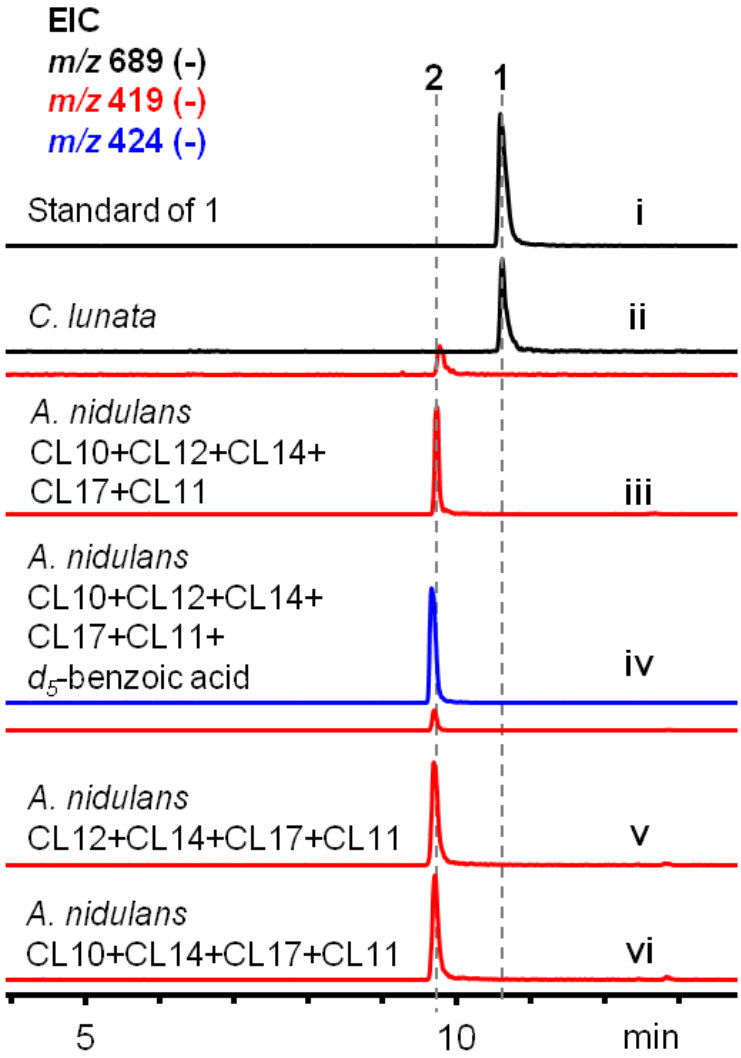


Figure 17. Zaragozic Acid A cluster and Metabolic Traces
Extracted ion chromatography of LC-MS traces from both *C. lunata* (13 days) and *A. nidulans* (3 days) showing production of metabolites of interest. All masses shown correspond to *m/z* $[M-H]^-$. **i**: Standard of **1**; **ii**: production of **1** and **2** from *C. lunata*; **iii-vi**: production of **2** from different combinations of genes from *C. lunata* reconstituted in the *A. nidulans* A1145ΔSTΔEM. **iv**: production of d_5-labeled **2** upon feeding of d_5-benzoic acid.

formation of new products. Introduction of the *clz* genes into the engineered *A. nidulans* A1145ΔSTΔEM strain led to the identification of a new mass peak (*m/z* $[M-H]^-$ 419) that was previously buried under the signals of sterigmatocystin and emericellamide B (Figure 14 and Figure S7). Exclusion of *clz14* abolished the production of this compound **2**, confirming it is derived from the HRPKS.

Large-scale fermentation was performed to isolate a sufficient amount of **2** (titer of ~ 0.1 mg/L) for NMR characterization and structural elucidation. **2** was found to have the molecular formula $C_{23}H_{32}O_7$ based on positive HRESIMS data (Figure S5). A database search for ZA related compounds revealed that a previous characterized compound, L-731.120, was isolated from the ZAA producing strain MF5453 (ATCC 20986),[167] which could potentially match **2.** Detailed analysis of the 1D and 2D NMR data of **2**, particularly the COSY and HMBC spectra, revealed the presence a monosubstituted phenyl ring and one trisubstituted double bond, which led to the full assignment of C-8 to C-19 fragment (Table S4 and Figure S13-17). Further analysis of the ^{1}H NMR data of **2** revealed the methylene group C-2 is bonded to two quaternary carbons, which is supported by the splitting pattern of H-2 (doublet) and its large coupling constants resulting from self-correlation (J = 16 Hz). This evidence combined with HMBC correlations from H-2 to C-20 and from H-4 to C-3, C-20 and C-21 established the tricarboxylic acid substructure. This moiety was connected to the benzyl-containing fragment via three methylene groups, which accounted for the rest of the unassigned atoms in the molecular formula. Thus, the panel structure of **2** was assigned. EIC analysis of the metabolite extract from *C. lunata* also revealed the presence of **2** (Figure 17 ii) along with **1**, further corroborating the relationship of **2** to the set of *clz* genes.

4.2.4 Investigation of benzoic acid starter unit priming

To investigate the priming pathways and incorporation of the benzoate starter unit by Clz14, we performed labeling studies using either d_5-benzoic acid or d_8-phenylalanine (Figure 17

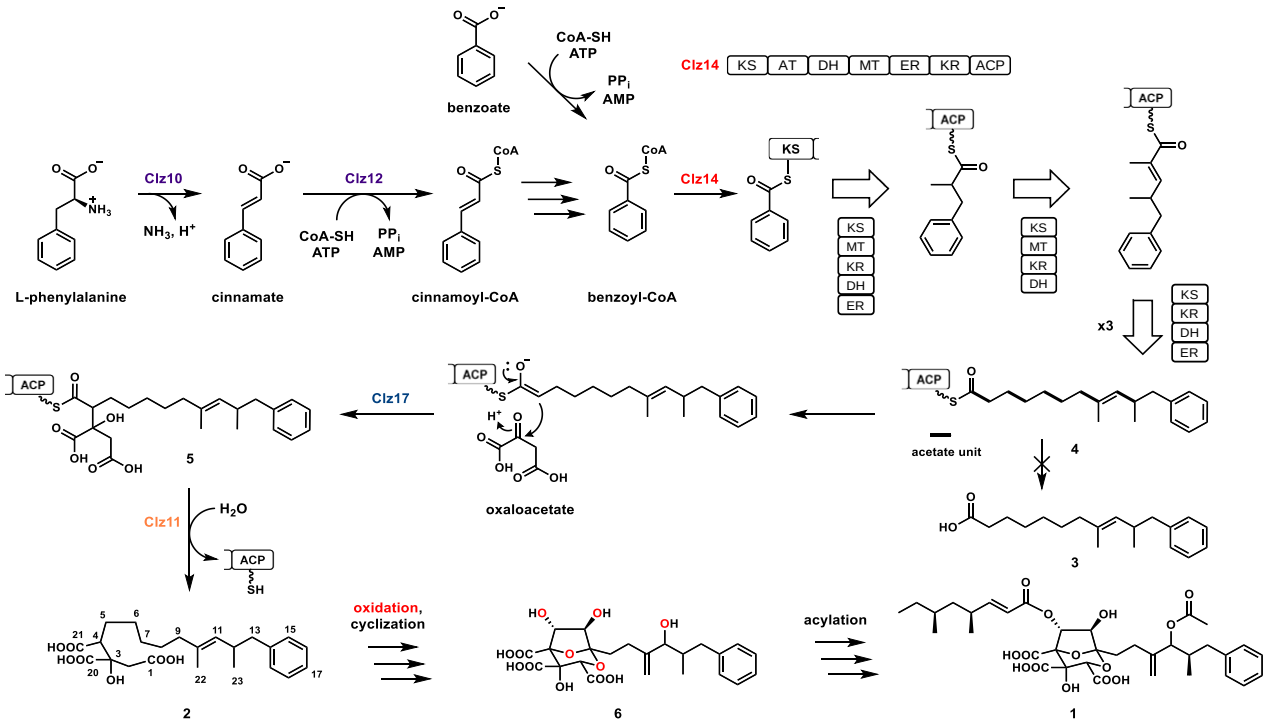


Figure 18. Proposed biosynthesis of zaragozic acid A (1)
Clz14 is the HRPKS involved in the first steps of the biosynthesis. The domains and steps of Clz14 are shown as white arrows. HRPKS domain abbreviations: KS: ketosynthase; AT: acyltransferase; DH: dehydratase; MT: methyltransferase; ER: enoylreductase; KR: ketoreductase; ACP: acyl carrier protein;

iv and Figure S4). In both cases, we observed the increase in molecular weight of **2** by 5 mu. In the case of d_8-phenylalanine feeding, retention of five deuterium labels is consistent with the proposed conversion to benzoic acid, during which PAL (Clz10) yields cinnamate, followed by esterification to yield cinnamoyl-CoA, which can undergo β-oxidation to yield benzoyl-CoA.[168] Incorporation of d_5-benzoate into **2** suggests the presence of an endogenous aryl-CoA ligase that

can afford benzoyl-CoA (Figure 18). When d_5-benzoic acid was fed at a high concentration (1 mg/mL), a significant amount (~30%) of 2 remained unlabeled, which represents the unlabeled benzoyl-CoA pool derived from phenylalanine. This parallel pathway to benzoyl-CoA was also observed in the priming steps of enterocin in *Streptomyces*,[169] and suggests Clz10 or Clz12 may not be absolutely required in *A. nidulans*. Indeed, removing either Clz10 or Clz12 from the *A. nidulans* constructs retained production of **2,** while removing both enzymes led to ~5 fold decrease in the titer of **2** (Figure 17). To probe if **2** biosynthesized from the minimal construct (Clz11, Clz14 and Clz17) is derived from benzoate in *A. nidulans*, we repeated the feeding of d_5-benzoate into this host. As shown in Figure S4, unlabeled **2** is nearly abolished, confirming that in the absence of Clz10 and Clz12, the level of benzoyl-CoA derived from phenylalanine is very low. These studies therefore indicate the importance of Clz10 and Clz12 in the priming pathways in the biosynthesis of **2**. These two enzymes are particularly essential if the level of benzoate in the native host is low compared to that in *A. nidulans*.

To also further explore the substrate scope of the unique priming mechanism of this polyketide system, we decided to try feeding different isosteres of benzoate that might be able to be incorporated into the polyketide arm. Feeding of furoic acid, thiophene carboxylic acid, and fluorobenzoic acid into the strain containing Clz10, Clz11, Clz12, Clz14, and Clz17 led to the formation of new products **7**, **8**, and **9** in addition to **2** (Figure 19). These masses detected corresponded to the incorporation of these acid substrates respectively. The fluorobenzoic acid and thiophene carboxylic acid were readily incorporated into the polyketide chain, while the furoic acid integrated polyketide was produced at much lower yield. These products represent novel chemical analogs of **2** and indicate a flexible substrate scope of the polyketide priming.

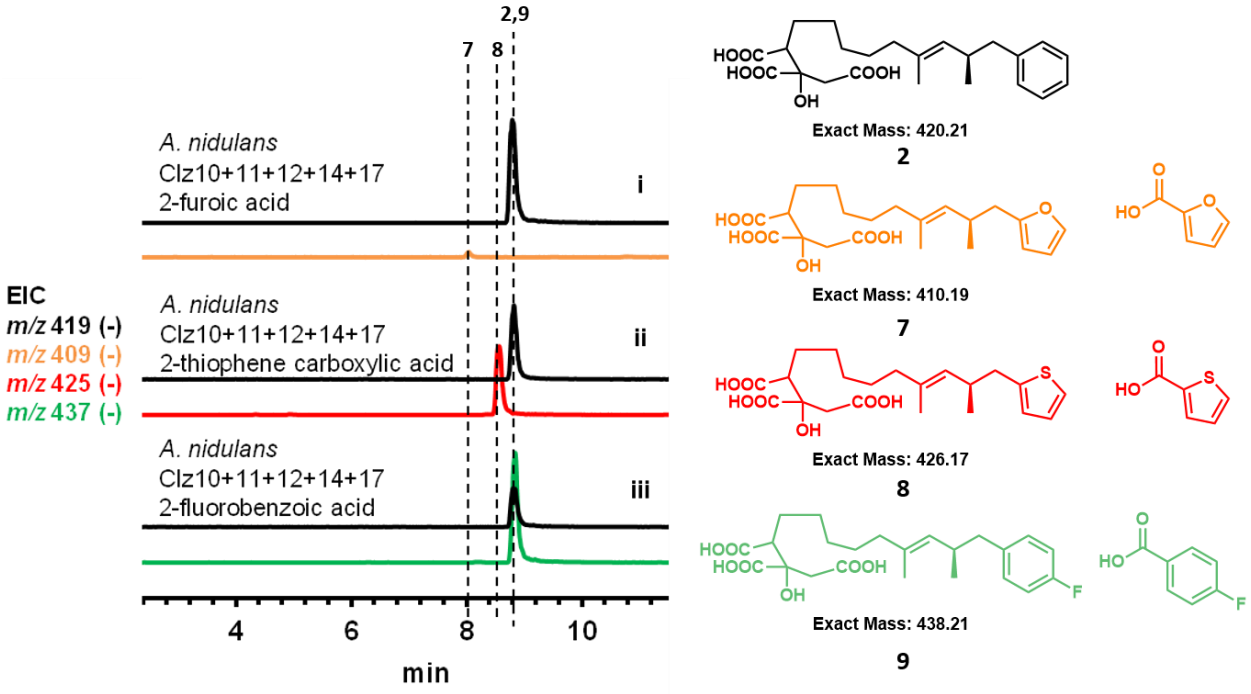


Figure 19. Generation of Chemical Analogs of 2
Precursor directed biosynthesis was introduced into the heterologous expression system of the ZA intermediate **2.**

4.2.5 Polyketide product release

We also investigated the product releasing steps that lead to biosynthesis of **2**. The citrate synthase Clz17 is essential in the biosynthesis as removing the gene from *A. nidulans* abolished production (Figure S3). We propose that following completion of the polyketide synthesis, which yields **4**, Clz17 catalyzes the addition to oxaloacetate to yield **5**. This is consistent with the role of homologous enzymes in the nonadride biosynthetic pathways, including those of phomoidride,[163] byssochlamic acid[164] and rubratoxin.[165] Whereas in these pathways, formation of the anhydride catalyzed by alkylcitrate dehydratase is the release mechanism, here **5** is directly hydrolyzed to yield **2.** We propose that the hydrolase Clz11 is responsible for this reaction and product turnover. Clz11 shares sequence homology to other uncharacterized fungal hydrolase enzymes such as in *Lepidopterella palustris* (62% id) or *Glarea lozoyensis* (48% id), but only very weak identity to LovG from the lovastatin pathway.[170]

Removal of Clz11 from the *A. nidulans* construct that produced **2** led to ~99% reduction of the product level (Figure S3), indicating while spontaneous or nonspecific enzyme hydrolysis is present, Clz11 significantly accelerates product turnover. Substituting Clz11 with the β-lactamase homolog Clz13 also resulted in only background level of product formation (Figure S3). The timing of Clz11 activity is precise and only acts to release **2** from **5**. In the construct without the citrate synthase Clz17, we could not detect any trace of the free polyketide **3**. The combination of citrate synthase and hydrolase therefore represents a unique mode of product release from the HRPKS.

4.3 Conclusions

We had successfully used targeted genome mining to identify the *clz* cluster. Using heterologous expression, we have shown that only three genes, Clz14, Clz17 and Clz11 are required to synthesize **2**, utilizing the benzoyl-CoA pool that is naturally present in *A. nidulans*. This represents an exceptionally concise pathway to a structurally complex, amphiphilic polyketide product. Benzoyl-priming of Clz14 is unprecedented among fungal PKS systems, and further biochemical characterization of the KS domain and the unique N-terminal region may reveal the molecular basis using the non-acetate starter unit. From **2**, a series of hydroxylation reactions must take place to furnish the other features of the proposed intermediate **6**, including the 2,8-dioxobicyclic[3.2.1]octane-3,4,5-tricarboxyclic acid core and the C-10/C-22 *exo*-methylene. In particular, hydroxylations at five sp^3 carbons C-2, C-4, C-5, C-6, C-7 are proposed to take place, representing a remarkable cascade of C-H activation steps on vicinal carbon atoms. These reactions may be iteratively catalyzed by two enzymes with homology to non-heme iron and α-ketoglutarate dependent oxygenases Clz15 and Clz16, which are also found in the homologous pathway from *Phoma* sp. C2932.[158] The identification and reconstitution of

2 described here set the stage for delineating the order and enzymology of these enigmatic reactions.

4.4 Materials and Methods

Strains and culture conditions

Curvularia lunata (Wakker) Boedijin var. *lunata* anamorph (MF5573) was purchased from ATCC® (74067™). *Curvularia lunata* was grown on Difco™ PDA (Potato Dextrose Agar) Plates from BD biosciences. To produce zaragozic acid A (**1**), *C. lunata* was inoculated into 20 mL of KF seed medium for 2 days before transferring 2 mL of the seed culture into LSF1 media. Production of **1** was seen from days 11-15. Media and steps used in this procedure are outlined by Bills *et al*.[171] *Aspergillus nidulans* A1145 was purchased from the Fungal Genetics Stock Center and used for heterologous expression of genes from *C. lunata. Escherichia coli* strain XL1 Blue was used for cloning. *Saccharomyces cerevisiae* strain BJ5464 was used for homologous recombination of DNA fragments to assemble the vectors used in heterologous expression.

C. lunata and *A. nidulans* gDNA extraction, RNA extraction, and RTPCR

The Zymo ZR Fungal /Bacterial DNA Microprep™ kit was used to extract gDNA from *C. lunata*. The Invitrogen Ribopure™ kit was used to extract RNA from *A. nidulans*. Superscript® III Reverse Transcriptase Kit from Life Technologies was used to synthesize cDNA from the RNA extracted from *A. nidulans*.

Genome sequencing, assembly and biosynthetic gene cluster prediction

The *C. lunata* genomic DNA was sequenced with three other fungal strains multiplexed on a single lane of Illumina HiSeq 2000 with the standard 2 x 100 bp paired end run mode at the Biomolecular Resource Facility (BRF) at The Australian National University. The resulting

FASTQ reads were first processed with Trimmomatic[172] to trim off the adaptor sequences and assembled using SPAdes.[173] Scaffolds containing PKS genes were first retrieved by performing a TBLASTN on the *C. lunata* genomic scaffolds with an arbitrary conserved fungal KS domain as query sequence. Gene predictions were performed on these PKS gene-containing scaffolds using AUGUSTUS[174] using the *Aspergillus nidulans* species parameters. Simultaneously, the scaffolds were submitted for secondary metabolite biosynthetic gene clusters prediction using AntiSMASH 3.0.[175] Among the predicted biosynthetic gene clusters, a gene cluster with two HRPKS genes were identified, in which one of the HRPKS gene exhibits high homology to the SQTKS from *Phoma* sp.[160] The gene cluster also contains a citrate synthase, which corresponds to presence of tricarboxylic acid unit in **1** and hence was assigned as the putative zaragozic acid gene cluster (named *clz* cluster). The *clz* gene cluster was later shown to be similar to the squalestatin S1 cluster identified from *Phoma* sp. C2932 and the unidentified fungus MF5453.[158]

Plasmid construction for heterologous expression

Plasmids pYTU, pYTP, pYTR were used as vectors to insert genes which contain auxotrophic markers for uracil (pyrG), pyridoxine (pyroA), and riboflavin (riboB), respectively.[176] Genes to be expressed were amplified through Polymerase Chain Reaction (PCR) using the gDNA of *C. lunata* as a template. The PCR products and the corresponding backbone digested with *Pac*I and *Swa*I were assembled with the Frozen-EZ Yeast Transformation II Kit™ (Zymo Research) by using yeast homologous recombination with BJ5464-NpgA, which contains a copy of *A. nidulans* phosphopantheteinyl transferase gene *npgA* integrated in the chromosome.[131]

Genetic transformation and heterologous production in *A. nidulans*

Protoplasts were generated by scraping spores from a solid CD Medium (10 g/L glucose, 50 mL/L 20x Nitrate Salts, 1mL/L Trace elements, 20% agar) Plate. The spores were transferred to 25 mL of liquid CD minimal medium and incubated for 12-13 hours at 37°C at 250 rpm. After incubation, the germlings were collected and washed with 10 mL of Osmotic Medium (1.2M $MgSO_4$, 10 mM $NaPO_4$) twice. The germlings were then transferred into 10 mL of Osmotic Medium containing 30 mg of Lysing Enzyme from *Trichoderma* and 20 mg of Yatalase. The culture was incubated for 12 hours at 28°C at 80 rpm. The cells were poured into a 30 mL Corex tube and overlayed with 10 mL of Trapping Buffer (0.6 M Sorbitol, 0.1 M Tric-HCl). The tube was centrifuged at 5000 RPM. The protoplasts were then removed from the interface of the two buffers and transferred to sterile tubes. 2x volume of STC Buffer (1.2 M Sorbitol, 10 mM $CaCl_2$, 10 mM Tric-HCl) was added to the protoplasts. DNA and 60% PEG4000 solution were added to the protoplast solution and incubated at room temperature for 20 min. The cells were then plated onto solid CD-Sorbitol Medium (10 g/L glucose, 50 mL/L 20x Nitrate Salts, 1mL/L Trace elements, 20% agar, 1.2 M Sorbitol). After transformants appeared on the plates, the spores were restreaked onto solid CD-ST production medium at 28°C for 4 days (20g/L Starch, 20g/L Peptone, 50mL/L Nitrate Salts, 1mL/L Trace elements). For isotope feeding studies, 1 mg/mL of deuterium labeled benzoic acid and phenylalanine were added to the solid CD-ST before streaking cells. Deuterium labeled precursors used for feeding were purchased from Cambridge Isotope Laboratories.

Sample analysis of *A. nidulans* transformants

A. nidulans transformants were grown on CD-ST for 2-4 days at 28°C for small scale analysis. Samples were extracted using 1 mL of acetone. After centrifugation, the supernatant was then dried and resuspended into equal volume methanol before injection for LC-MS

analyses. LC–MS analyses were performed on a Shimadzu 2020 EV LC–MS (Kinetex 1.7 m C18 100 Å, LC Column 100 × 2.1 mm) using positive-and negative-mode electrospray ionization with a linear gradient of 5–95% acetonitrile MeCN/H_2O with 0.5% formic acid in 15 min followed by 95% MeCN for 3 min with a flow rate of 0.3 mL/min. Standard of **1** purchased from Cayman Chemicals was used for comparison.

Compound isolation and structure elucidation

16 L of CD-ST medium were poured on solid plates. Spores from solid CD media plates were restreaked onto the production medium and grown in 28°C for 3 days. Samples were extracted using acetone. The acetone was evaporated, leaving an aqueous solution of extracted metabolites. The pH of the aqueous solution was adjusted to ~pH 10 then washed with ethyl acetate. The aqueous solution was then adjusted to ~pH 3 and extracted with ethyl acetate. This organic layer was dried *in vacuo* and resuspended in methanol, and fractionated using Sephadex LH-20 size exclusion column, a CombiFlash® System, and semi-prep reverse phase HPLC sequentially. The CombiFlash® System used a 100 g HP CL18 reverse phase column with a linear gradient of 5–95% acetonitrile MeCN/H_2O with 0.1% formic acid in 60 min at 60 min/mL. HPLC fractionation used a semi-preparative C18 column of Kinetics New column, 5m, 10 × 250 mm with an isocratic method of 45%/55% acetonitrile/water with 0.1% formic acid for 45 min at 4 mL/min.The collected fractions were analyzed using LC-MS with a linear gradient of 5–95% acetonitrile MeCN/H_2O with 0.5% formic acid in 15 min and desired fractions were pooled. For elucidation of chemical structures, 1D and 2D NMR spectra were obtained on Bruker AV500 spectrometer at the UCLA Molecular Instrumentation Center. High resolution mass spectra were obtained from Thermo Fisher Scientific Exactive Plus with IonSense ID-CUBE DART source at the UCLA Molecular Instrumentation Center.

5. Targeted Genome Mining for the Discovery of Natural Product Cyp51 Inhibitors

We had successfully identified and characterized the Zaragozic Acid A biosynthetic gene cluster by using squalene synthase as our search through the genome of a known producer of the cholesterol lowering natural product. However, we wished to find novel sterol pathway inhibiting fungal natural products. Using the TGIF algorithm, we evaluated enzymes found in the sterol biosynthetic pathway as targets to search for possible clusters making sterol pathway drugs. One target that gave several cluster hits was lanosterol 14-α demethylase p450 (Cyp51). Among these, we heterologously expressed a cluster from Scaffold15 of *Aspergillus nomius*, leading to the production of analogs of restricticin, a natural product inhibitor of Cyp51. Here we identified the BGC of an inhibitor of Cyp51 and showed several biosynthetic steps of the molecule, evaluated the self-resistant copy of Cyp51 in the cluster, and found other clusters with potentially novel Cyp51 inhibitors.

5.1 Introduction to Antifungal drugs and Lanosterol α-14 demethylase Cyp51

Fungal infections are worldwide issue that plague over a billion people, leading to up to 1.5 million deaths annually.[177] The diversity of fungal species is estimated to be around 1.5 million different types, many of which are opportunistic plant and human pathogens that cause damage in agricultural and clinical settings. The most common fungal human pathogens include *Candida albicans, Aspergillus fumigatus, Aspergillus flavus,* and *Cryptococcus neoforms* that have exacerbated effects on immunocompromised patients.[178] The clinical armamentarium available to combat mycoses include a broad variety of pharmaceuticals with different modes of action, including the common classes of antifungals: 1) Polyenes, 2) Azoles, 3) Allylamines, 4) Echinocandins, 5) Griseofulvin, 6) Flucytosine, etc.[74, 179] Despite these drugs, the rise of

antifungal resistance remains a challenge that requires the search for novel antifungals with new modes of action.

Many of the classical antifungals (polyenes, azoles, and terbinafines) are specifically aimed at disruption of the cell membrane integrity of the fungi through disuprtion of ergosterol biosynthesis. Cyp51 (erg11) is a key enzyme in the sterol pathway, performing the C14 demethylation of important sterol intermediates through a multistep oxidative reaction. Cyp51 belongs to the cytochrome p450 superfamily, which is represented by heme-dependent, membrane bound monooxygenses whose functions sweep through both primary and secondary metabolism. Though there is significant variation in the p450s among different organisms, Cyp51 is believed to be the ancestor to all p450s, as it is the only one found through all biological kingdoms.[180] The structural features of Cyp51 are highly conserved, which include six regions that are used as substrate recognition sites (SRS). Among these regions is SRS1, which forms the upper surface of the binding cavity; conservation of these residues contributes greatly to the rigid substrate specificity of the p450 enzyme.[92] The conservation of the structural features of Cyp51 also allow for a narrow specificity of only four known substrates: lanosterol, 24,25-dihydrolanosterol, 24-methylenedihydrolanosterol, and obtusifoliol throughout all biological kingdoms.[181] The ubiquity of this enzyme makes it an important checkpoint, since evolutionary divergence of the sterol pathway occurs in the steps following 14-α demethylation.

Inhibitors of Cyp51 represent an ongoing effort to find cholesterol lowering, herbicidal, and antifungal agents. Currently the development of Cyp51 inhibitors with antifungal activity has been the most successful and prevalent, mostly represented by the azole drugs. As discussed previously, these azole drugs consist of imidazole and triazole moiety containing molecules that are potent inhibitors of Cyp51. These azoles have high efficacy against a broad range of fungal

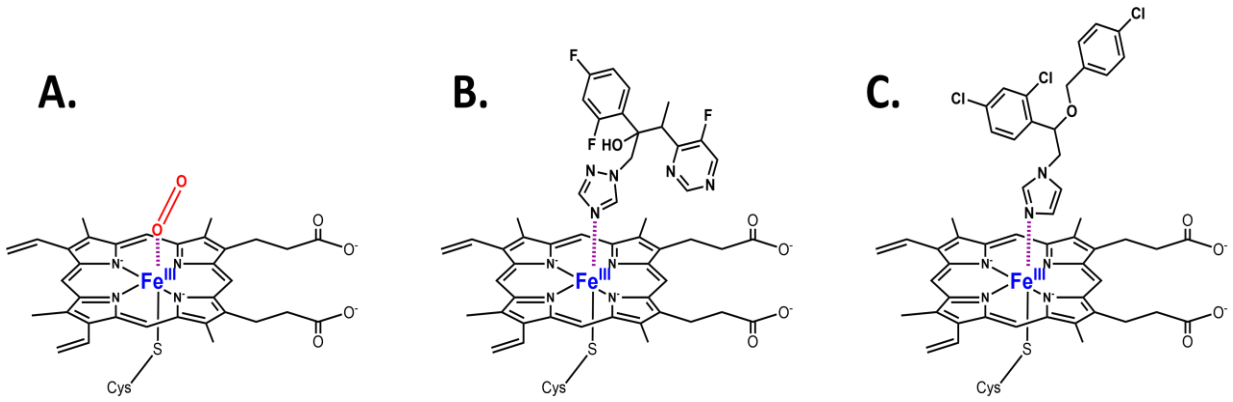


Figure 20. Cy51 Inhibition by Azole drugs
Azole drugs work by inhibiting the heme-iron complex of Cyp51 through binding of the axial substrate position through the nitrogen found in the **B.** triazole group or **C.** the imidazole group. This position is usually binded by **A.** molecular oxygen which is incorporated into the substrate through oxidation.

pathogens. The nitrogen of the azole groups on the drugs serve as the warhead of the molecules for inhibiting the enzyme through coordination to the heme iron complex of the p450 enzyme at axial ligand position where oxygen usually binds. (Figure 20) The anchor of the azole molecules then interacts with the binding pocket of the enzyme through Van der Waals, hydrophobic, and aromatic stacking.[182]

As with many antibiotics, fungal azole resistance has been an issue that has emerged with many of the virulent fungi pathogens and is an active area of study. In eukaryotes, organisms usually maintain one copy of Cyp51. In fungi, however the distribution of Cyp51 is varied between phylogeny groups. This was first noticed in 2001 that *Aspergillus fumigatus*, an azole resistant human pathogen strain, had two homologs of Cyp51, Cyp51A and Cyp51B, hinting at a possible means of resistance through the duplicate copies.[183] Further phylogenetic analysis of these protein sequences showed the same dual copy Cyp51 in filamentous fungi such as *Aspergillus, Magnaporthe, Penicillium, Pyrenphora, Trichoderma, and Fusarium*.[184] Though Cyp51A and Cyp51B cladded differently, both were found to be replaceable, as only one copy of

either Cyp51A or Cyp51B was needed to maintain ergosterol biosynthesis activity. In some strains, another homolog of Cyp51, Cyp51C, has been found in some fungi, such as in *Aspergillus flavus*.[185] Genetic analysis of azole resistance strains has been an ongoing effort to elucidate the gain of function mutations that allow for resistance in *Candida albicans*, *Aspergillus fumigatus*, *Histoplasma capsulatum*, etc. The mutations found usually affect the azole efficacy through disruption of amino acids that interact with the docked azole ligand in the binding pocket, structural rearrangement of the residues anchoring the heme binding, or alterations in the substrate entrance cavity.[185-188]

5.2 Results and Discussion

5.2.1 Targeted Genome Mining of Cyp51 with TGIF

We aimed to identify natural products that would have capabilities to inhibit the sterol pathway in hopes of discovering novel anti-cholesterol or antifungal agents. To do this, we decided to evaluate popular drug targets of the sterol pathway that have been used for the development of small molecule pharmaceuticals. As introduced in the previous section, Cyp51 is an important drug target that has been capitalized on to develop the azole drugs whose fungistatic capabilities derive from their ability to inhibit the biosynthesis of ergosterol. We used a database of 80 fungal genomes of strains from our laboratory that we would be able to quickly evaluate and study once gene clusters of interest were identified. As mentioned previously, a couple of Cyp51 homologs, Cyp51A and Cyp51B are commonly found among fungal species. This was confirmed when running the TGIF analysis on our fungal database, which showed that the average number of Cyp51 copies across our fungal database was 1.912 hits per genome, while the median number of Cyp51 copies was 2. To find possible cluster candidates, we

identified genomes that contained >2 copies of Cyp51. In addition, the genomes must contain

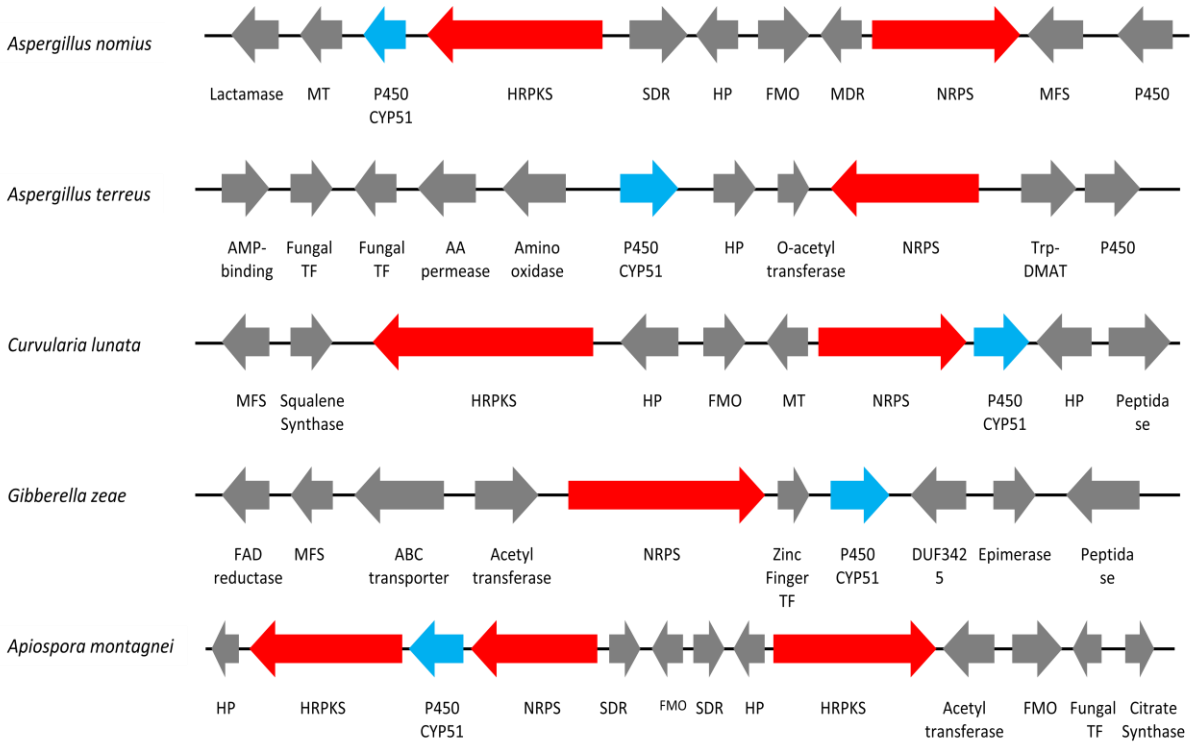


Figure 21. TGIF results for CYP51 Targeted Mining
Clusters identified through using the TGIF algorithm for targeted genome mining of Cyp51. The clusters in *A. nomius*, *C. lunata*, and *A. montagnei* are homologous. Core enzymes are highlighted in red while the Cyp51 target is highlighted in turquoise.

Clusters that colocalize core secondary metabolite enzymes with a copy of Cyp51. The colocalized distance was defined to be 20,000 nucleotide bp allowed between the target enzyme and a core enzyme.

With these parameters we were able to identify five cluster candidates (Figure 21), two of which were from *Aspergillus nomius and Curvularia lunata* that were homologous clusters. The Cyp51 of *A. nomius* and *C. lunata* had an identity of 68% and 60%, respectively, to the Cyp51A housekeeping gene of *A. nidulans*. This range of homology gave us confidence that the Cyp51 genes found in these clusters were close enough to housekeeping Cyp51 that they were not other

members of the P450 superfamily (members of the same p450 subfamily have >40% identity), yet mutated enough that it could be a self-resistance enzyme. The conservation of the two clusters among *A. nomius* and *C. lunata* also gave us confidence that these were secondary metabolite gene clusters and we thus decided to pursue the study of these two as possible biosynthetic gene clusters of a Cyp51 inhibitor natural product.

5.2.2 Genetic analysis of the *Aspergillus nomius* Biosynthetic Gene Cluster

Bioinformatic analysis and a search through the public fungal genome databases revealed other homologous gene clusters found in *Epicoccum nigrum*, *Aspergillus bertholletius*, and *Aspergillus pseudonomius*. (Figure S8). By comparing the genes conserved through the three genomes, we were able to draw the gene boundaries of the cluster based on the genes conserved through all of the different species. There are 6-8 genes total in the cluster, including two core enzymes, a HRPKS (*rstn3)* and a single modular NRPS (*rstn8).* In *A. nomius* and *A. pseudonomius*, the NRPS follows a A-T-C domain organization sequence. Whereas in the other strains the NRPS contains additional terminal domains; in *A.bertholletius* the NRPS follows a A-T-C-TE organization and in *E. nigrum* and *C. lunata* the NRPS follows a A-T-C-R domain organization. The changes in how these domains are constructed can lead to differences in the biosynthetic mechanisms of these steps.

In addition to the core enzymes, there are several other biosynthetic enzymes conserved including a flavin dependent monooxygenase (*rstn5)*, methyltransferase (*rstn1)*, and reductase enzymes that are not present in *E. nigrum* and *C. lunata* (*rstn4* and *rstn7*). There is also a hypothetical protein (*rstn6*) which has function that could not be predicted using primary or secondary sequence alignment. Finally there is the proposed resistance enzyme, the Cyp51 p450

monoxygenase (*rstn2)*. With the combination of an HRPKS and NRPS core enzymes both in the cluster, we hypothesized that a lipopetide like product could be produced by this cluster.

5.2.3 Heterologous expression of the entire *rstn* cluster

The genes from *A. nomius* and *C. lunata* discussed above were thus reconstituted in the *A. nidulans* A1145ΔEM heterologous expression host to determine the products of these clusters. Efforts to produce the compound using the *C. lunata* cluster genes provided a new product upon the expression of the HRPKS. However, when more genes were added, the formation of new product intermediates was not observed. The polyketide product was isolated, but too unstable to get structural data from.

Heterologous expression of the cluster from *A. nomius* in *A. nidulans* A1145ΔEM on the other hand produced new hydrophobic compounds in CD minimal media. These two peaks corresponded to molecules with molecular weights of 705 and 719 (Figure 21 iii). The detection of these molecular weights was surprising at first, as the relatively small cluster in *A. nomius* did not seem capable of producing such large molecules. Nevertheless, large scale isolation was performed to accumulate the compounds and through NMR analysis the new peaks were solved to be corresponding to **20, 21** (MW: 705) and **19** (MW: 719). These structures were novel compounds that contained elements of the three major types of secondary metabolites: polyketide, non-ribosomal peptide, and terpenes, an unprecedented combination of these core enzymes. **19-21** all contained a tetrahydropyran ring extended by a methylated triene C9 tail on the C5’ position, likely to be the polyketide product introduced by the *A. nomius* cluster. The C3’, C4’ diol on the tetrahydropyran are modified through O-methylation and glycyl esterification, respectively. The glycine bridges the polyketide product to a prenylated isoindolinone product. Compounds **19-21** differ in methylation of the C3’ hydroxyl and C7

hydroxyl. A search of this molecule through the chemical database revealed that the structure consisted of two known natural products: aspernidine and restricticin (Ro-1470). To verify that

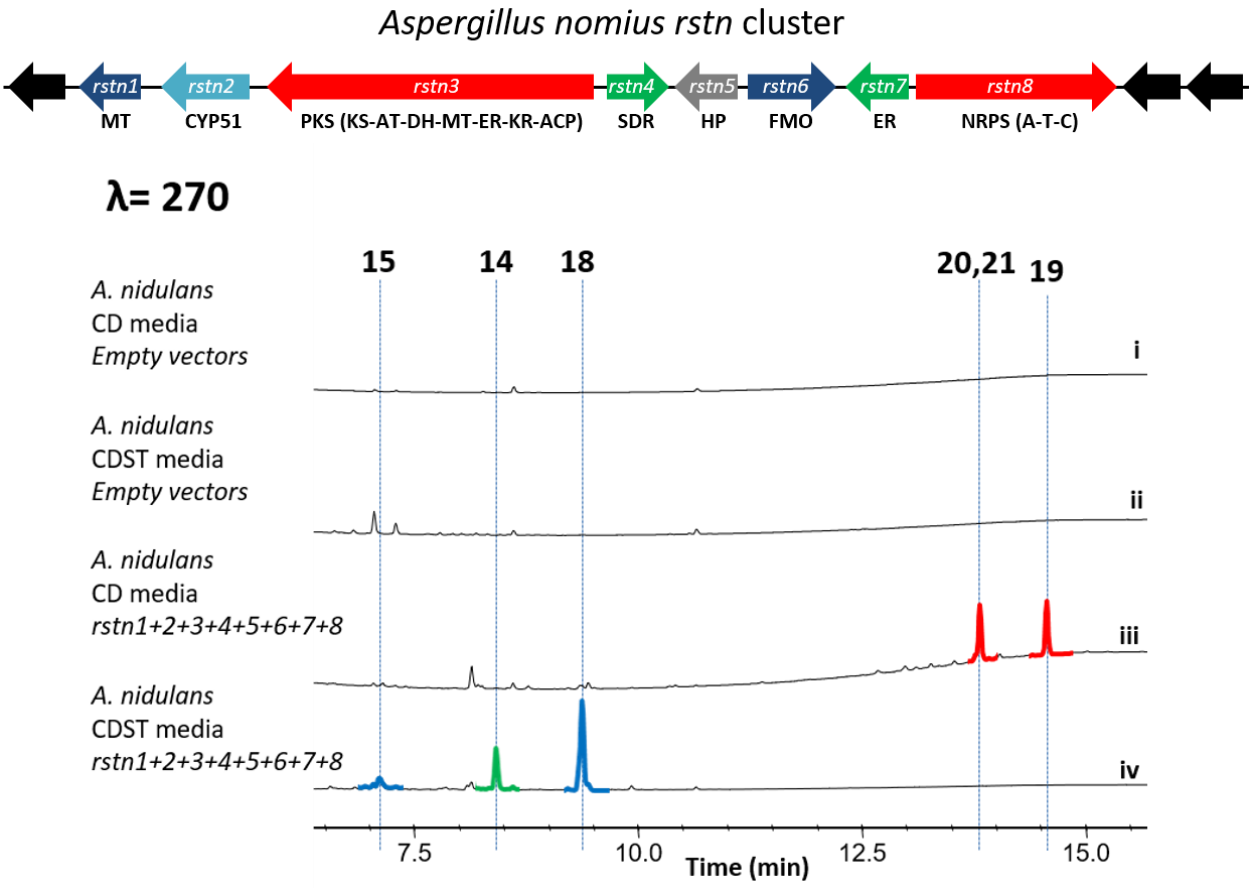


Figure 22. Heterologous expression of the entire *rstn* cluster in *A. nidulans*
Heterologous expression of *rstn1-rstn8* produced different compound depending on the production media used. **(i)(ii)** Negative controls with empty vectors transformed into *A. nidulans*, grown on CD and CDST production media **(iii)** In minimal media CD, the large hybrid molecules **19, 20,** and **21** were produced. **(iv)** In CDST, restricticin (**15**), and intermediates and derivatives of restricticin were produced (**14** and **16**).

the bridge was indeed derived from glycine, we fed d_2-glycine into the *rstn1-8* producing strain nd found an increase of the masses of **19-21** by 2 mu, indicating the incorporation of the labeled glycine into the structure. This confirmed that glycine was indeed the amino acid connection between the tetrahydropyran polyketide and the prenylated isoindolinone (Figure S10).

Aspernidine is a secondary metabolite discovered from *A. nidulans* through generation of an extract library through a variety of culture conditions. The compound is not found at normal

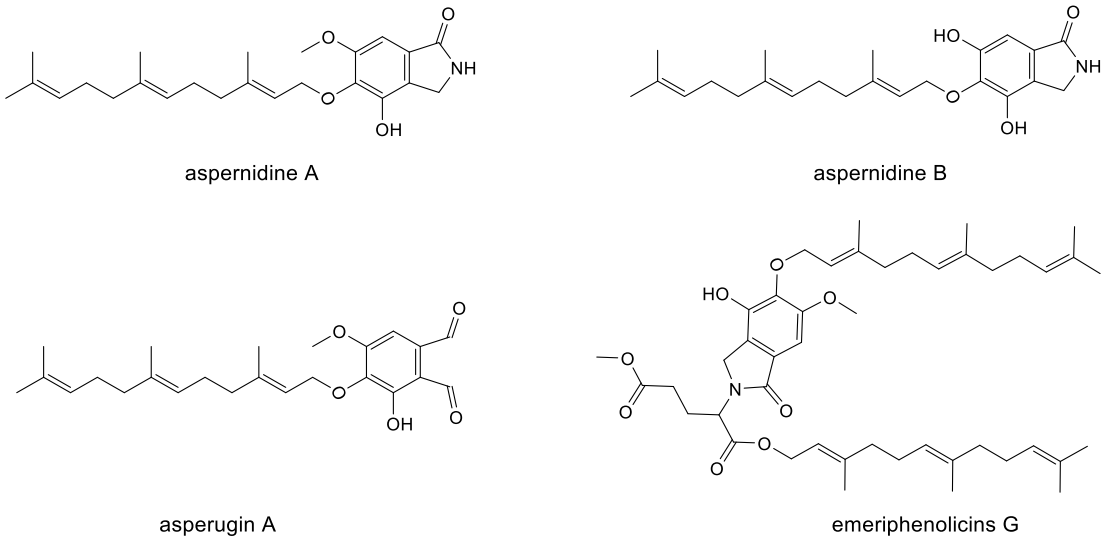


Figure 23. Aspernidine related secondary metabolites isolated from *A. nidulans*

culturing conditions and was screened for only when the media was supplemented with uridine and aminobenzoate.[189] The biosynthetic gene cluster was first elucidated through screening of a genome-wide kinase knockout library that was generated in hopes of activating cryptic gene clusters. A Δ*mpkA* strain was discovered to produce the aspernidine A and B (Figure 23), and thus served as model strains for the study of the BGC that produced the molecule. The *mpkA* gene encodes mitogen-activated protein kinase which is vital for cell wall integrity signaling and germination. The isoindolinone core was proposed to start from an orsellinic acid aldehyde, generated from a NRPKS. Nitrogen from ammonia could be then captured by the aldehyde and closure of the ring through oxidoreduction and condensation. KO studies confirmed that *pkfA,* a NRPKS, is responsible for the biosynthesis of aspernidine and *pkfE* from the same cluster, is a prenyltransferase that connects a farnesyl group to the benzylic hydroxyl.[190] Other isoindole containing products have been isolated from other *A. nidulans* strains which contain different

amino acid substitutions at the nitrogen of the pyrroline ring.[191, 192] These related compounds also indicate the possibility of incorporating different amino acid into the isoindole ring.

The second half of **19-21** corresponds to the natural product restricticin. Restricticin (Ro 09-1470) is a tetrahydropyran containing polyketide product that was first discovered in 1991 by two separate research groups.[193-195] Restricticin and its related compound,

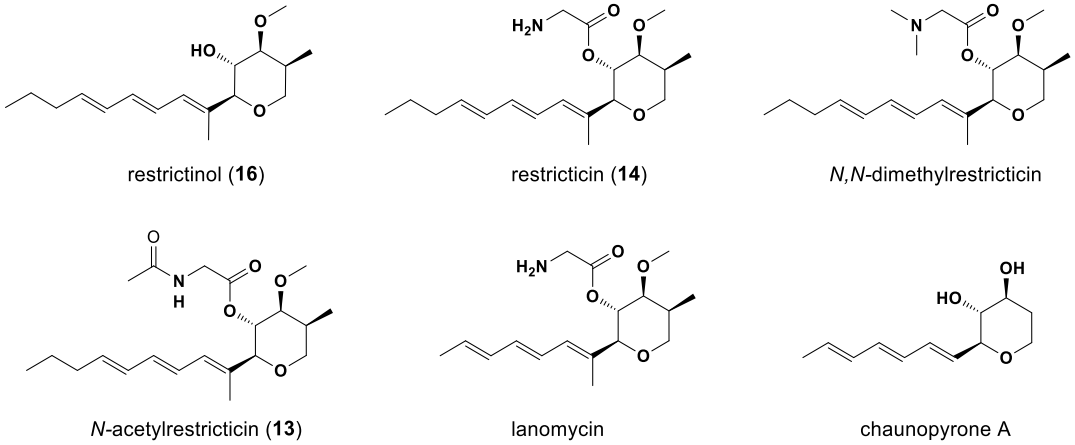


Figure 24. Isolated restricticin and related natural products

lanomycin, are the only natural product inhibitors of Cyp51 that have been discovered so far (Figure 24). As the aspernidine molecule is a natural metabolite from the *A. nidulans* heterologous host, we concluded the restricticin portion of **19-21** should be derived from the gene cluster from *A. nomius* that we introduced. Analysis of the same strain in rich CDST production media resulted in the increased production of two new peaks with molecular weights of 337 and 379, corresponding to possible structures of **15** and **18**, respectively (Figure 22 iv). We performed large scale isolation of these molecules and found **18** to be indeed corresponding to the *N*-acetylated restricticin. However, the yield of **15** was low and subsequently was lost throughout the purification process. This was corroborated by the literature in which the authors indicated the instability of restricticin due to the labile ester bond and the tendency of the triene

to decompose.[194] Compounds **18-21** can be found in both CD and CDST media, although the yields of both changed depending on the media type. Confirmation of the production of these compounds led us to designate the gene cluster as the *rstn* cluster that produces restricticin.

Restricticin was isolated from *Penicillium restrictum* and other *Penicillium spp*[193, 196] and had potent antifungal activity against a broad range of fungi including *Candida glabrata, Cryptococcuse neoformans, Rhodotorula rubra, Trichosporon cutaneum,* etc. The antifungal spectrum of restricticin was found to be close to that of ketoconazole (KCZ), though the potency of restricticin was about 2x less than KCZ based on the MIC tested for the fungal panel. Evaluation of *Candida albicans* 652 treated with restricticin showed marked decreases in sterol content from the metabolic profile, similar to the effects of KCZ.[194, 197] These studies suggested that restricticin could have a similar mode of action to KCZ in the inhibition of ergosterol biosynthesis. Based on these studies, Aoki et al. investigated the specific mode of action studies with restricticin, using purified microsome fractions containing Cyp51 from *Saccharomyces cerevisiae* and from rat liver. Restricticin effectively bound to the heme of the *S. cerevisiae* Cyp51 at a 1:1 ration based on the spectral change measurements, indicating specific inhibition of lanosterol 14-α demethylase. However the IC_{50} values for inhibition of the rat liver p450 were 300X than the *S. cerevisiae* p450, implying a much better specificity for antifungal activity.[198] Structure activity relationship studies also showed that restricinol had no antifungal or Cyp51 inhibitory activity while *N*-acetyl restricticin had much weaker inhibitory activity. This strongly suggests the importance of the free glycyl ester of restricticin for bioactivity. The nitrogen of the glycine likely coordinates to the heme of Cyp51 in similar fashion to the imidazole and triazole groups of the azole drugs.

Due to its attractive bioactivity, various total syntheses studies were also performed to restricticin,[199-201] however the biosynthetic route to restricticin was never elucidated. Therefore, the identification of **19-21** and **18** confirms that we have successfully utilized targeted genome mining with TGIF to locate the BGC of a Cyp51 inhibitor fungal natural product. As *A. nomius* is not a known producer of restricticin and growth of *A. nomius* under various conditions did not yield production of restricticin, we have also successfully activated a silent gene cluster by introducing it into our *A. nidulans* heterologous host.

5.2.4 Identification of intermediates of the *rstn* cluster

After successfully identifying the *rstn* cluster, we wished to further elucidate the biosynthetic steps that lead up to the formation of restricticin. Different combinations of genes were tested to evaluate the steps leading up to the formation of **13**. We hypothesized that the pathway should start with the formation of the polyketide product using the HRPKS *rstn3* and may require the reductase enzymes *rstn4* and *rstn7* for release off the polyketide synthase. We first tested combinations of these genes to see if any new metabolite could be formed. Even with the introduction of *rstn3* we begin to see the emergence of new peaks containing high UV signals (over 300 λ). These peaks also contained characteristic UV profiles that were consistent with conjugated polyene products that would indicate possible linear polyketide precursors for restricticin (Figure 25), including **10** (MW: 246). Attempts to isolate **10**, however, were unsuccessful as the compound was unstable and degraded during the purification process.

Only with the addition of the FMO *rstn5* and hypothetical protein *rstn6*, did we see the production of a new set of peaks, corresponding to **13** and **16** (Figure 26 ii). Both *rstn5* and *rstn6* are both required for the formation of **13** and **16**, as leaving either gene out did not produce these peaks (Figure S9). Large scale fermentation was done to isolate sufficient amounts of these

compounds to verify the structures of these molecules using NMR. We found **13** (MW: 266) and **16** (MW: 282) to be corresponding to the diol contacting tetrahydropyran products that

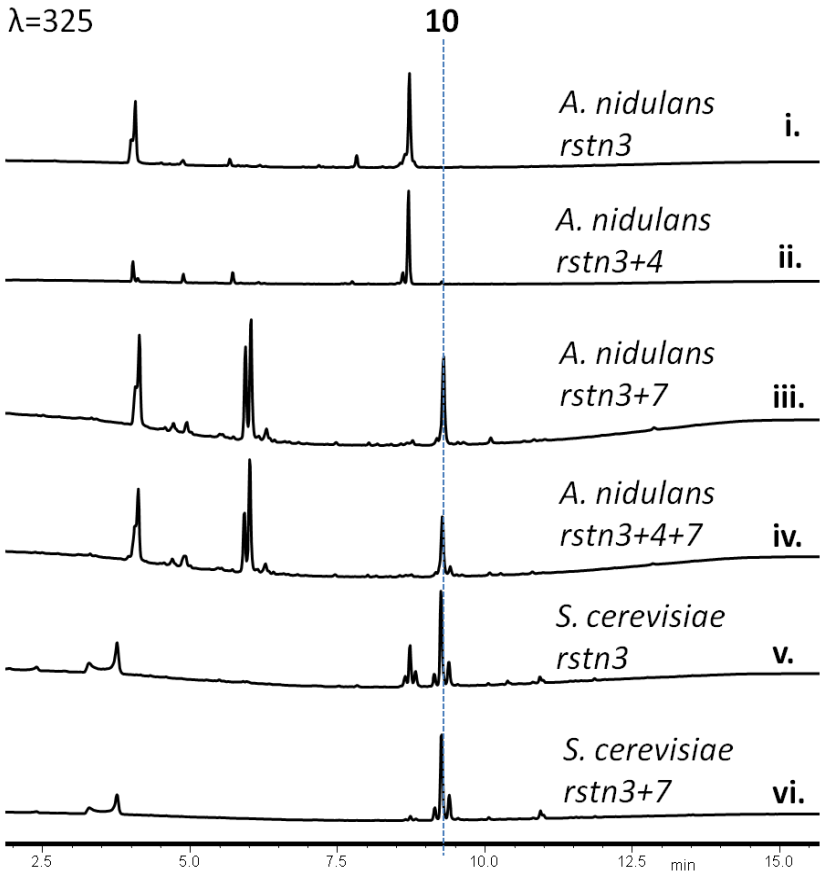


Figure 25. Heterologous expression of early steps of the restricticin pathway
Early steps of the *rstn* pathway expressed in both *A. nidulans* and *S. cerevisiae*. **10** is proposed to be the true intermediate of the pathway that is produced with the combination of *rstn3 + rstn7* and has a MW of 246. The other peaks are believed to be shunt products of the pathway.

make up the core of restricticin. **16** is the terminal C14 hydroxylated product of **13**, likely a result of endogenous enzymes from *A. nidulans*, as versions of this molecule have never been isolated from restricticin producing hosts. Addition of the methyltransferase *rstn1* resulted in the production of **14** (MW: 280) and **17** (MW: 296) (Figure 26 iii). The strains producing these

molecules were also fermented at large scale and isolated for NMR characterization, verifying these products as restrictinol **14**, and the terminal C14 hydroxylated restrictinol **17**.

The isolations of these compounds allow us to propose a biosynthetic route that the *rstn* gene cluster takes to synthesize restricticin and its derivatives found in the *A. nidulans* heterologous host (Figure 27). We propose the biosynthesis begins with the HRPKS *rstn3* which works in tandem with *rstn7* to form the heptaketide linear product **10**. **10** can then be further modified with *rstn4* to get **11** by reducing the C3 ketone to a hydroxyl and the terminal C1 aldehyde to an alcohol. The FMO *rstn5* and *rstn6* work together to facilitate the epoxidation across the C4, C5 double bond which sets up for the intramolecular cyclization through opening of the epoxide ring with the terminal hydroxyl to generate the tetrahydropyran ring with a C3, C4 diol. The C3 hydroxyl is then further modified through O-methylation using *rstn1*. The C4 hydroxyl is functionalized using the NRPS *rstn8* which attaches a glycine through esterification of the amino acid to the hydroxyl group to finish restricticin. Restricticin is then further metabolized by the heterologous host to be either acetylated on the free amine of the glycyl ester to form **18** or combined with the aspernidine pathway to capture the amine of the glycyl ester in the isoindolone ring to form **19-21**. It is likely the restricticin derivatives are an effort by the *A. nidulans* host to detoxify the antifungal agent through protecting the free amine that serves as the warhead of restricticin in interacting with the heme group of the target enzyme Cy51.

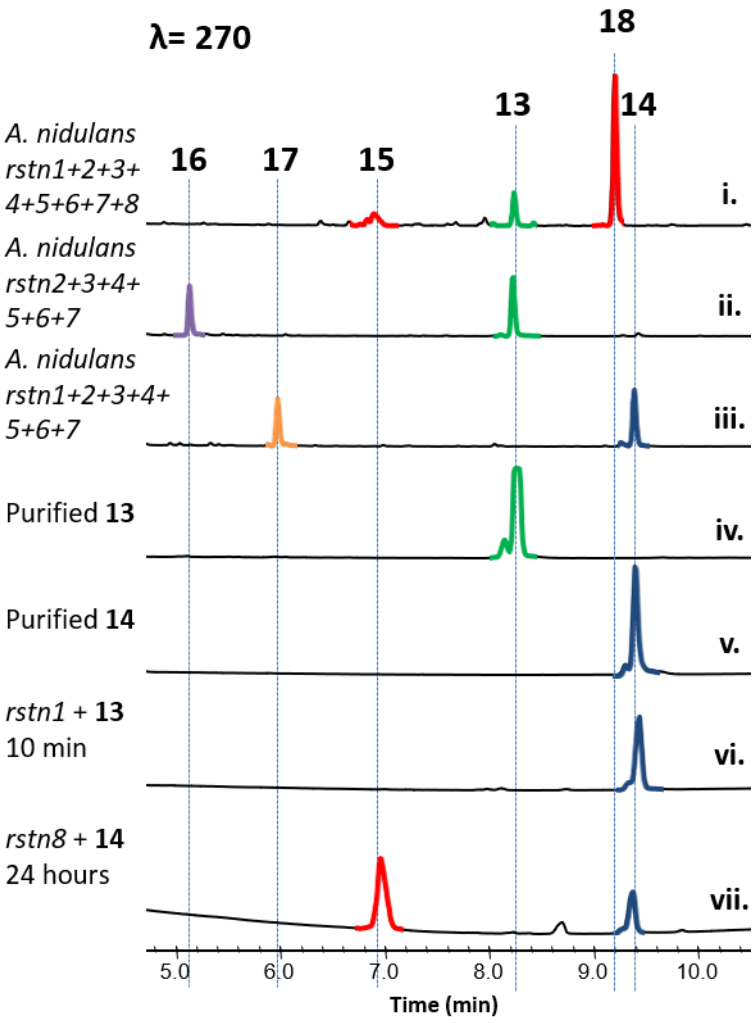


Figure 26. Heterologous expression of early steps of the restricticin pathway
Late stages of the *rstn* biosynthetic pathway were done in *A. nidulans*. Only with the addition of *rstn5* and *rstn6* were able to form a cyclized product, **13. 13** is then methylated at the C3 hydroxyl followed by esterification with glycine to **14.** Metabolized products with a terminal hydroxyl group were also identified in the culture medium from the heterologous expression.

5.2.5 *In vitro* verification of final steps of the pathway

To confirm that the intermediates that we isolated, **13, 14,** and **17** were true intermediates of the pathway, we decided to study the enzymes, *rstn1* and *rstn8* further. To perform these *in vitro* studies, we heterologously expressed the two genes in *Escherichia coli*, specifically a BL21

strain that contains the *npgA* gene from *A. niger* that allows for the 4'-Phosphopantetheine prosthetic group to the T domain of the NRPS (*rstn8*). The proteins were then expressed at large scale culture and purified for testing. We first evaluated the methyltransferase *rstn1* by performing the *in vitro* reaction with **13** and S-adenosylmethionine, the cofactor for methyl transfer enzymatic reactions. Immediate conversion was seen of the substrate **13** to **14** (Figure 26 vi). Next, we tested the NRPS *rstn8*. By reacting *rstn8* with **14** and glycine, we were able to convert **14** to a product with the same retention time and mass as **15**, restricticin (Figure 26 vii). To further confirm that **15** was the product of the *rstn8* reaction, we added acetic anhydride to the reaction mixture to functionalize the amine group of **15** to yield **18.** Indeed, the addition of acetic anhydride was successful in acetylation of **15** to form **18** (Figure S12).

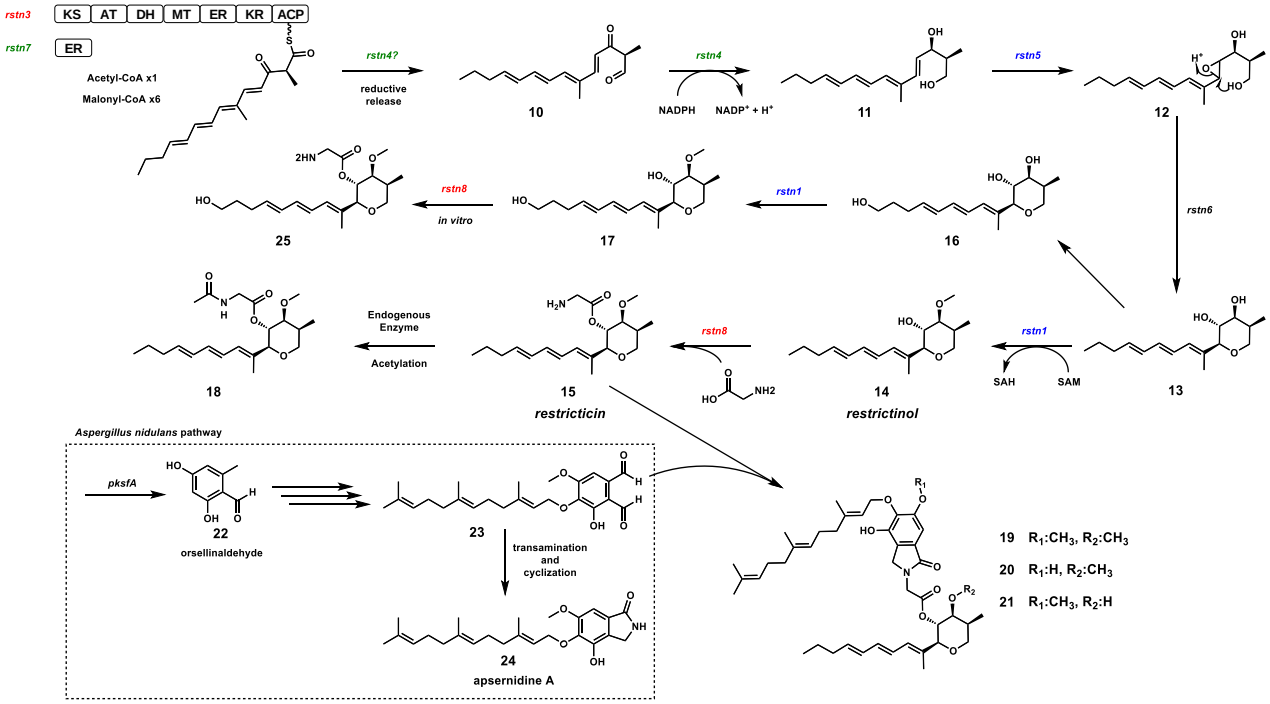


Figure 27. Proposed Biosynthetic Mechanism of restricticin
Rstn3, *rstn4*, and *rstn7* are believed to start the biosynthesis by synthesizing the linear polyketide aldehyde product. The FMO, *rstn5*, can then generate the epoxide across the C4-C5 double bond. This is led by the C1 hydroxyl attack to open the epoxide the to form the tetrahydropyran cyclized product and the C3,C4 diol with opposite stereochemistry. The diol is then

functionalized by first methylation of the C3 hydroxyl with *rstn1* followed by attachment of a glycine through an ester bond with *rstn8* to form restricticin. Restricticin is further metabolized by the *A. nidulans* host to detoxify the compound through protection of the free amine in the forms of **18** and **19-21.**

The *rstn8* reaction thus represents a rare case of esterification performed by a NRPS in contrast to the more common amide bond condensation reactions done by canonical NRPS enzymes.

We further decided to test the substrate scope of *rstn8* by testing different amino acid substrates as well as different polyketide substrates. For the amino acids, we tried glycine, beta-amine alanine, aminovaleric acid, valine, leucine, lysine, alanine, and methylated versions of glycine with the natural substrate **14**. For the polyketide substrates we tried compounds **13, 14,** and **17** with the natural amino acid substrate glycine. The NRPS was not able to take any amino acid substrate other than glycine, showing a very specific substrate preference (Figure S11). The NRPS was able to take the terminal hydroxylated product **17** and convert it to **25**. This shows that the polyketide backbone substrate scope may be more flexible than that of the amino acids. In contrast, *rstn8* was not able to esterify glycine to **13**, showing the rigid timing and requirement of the C3 methyl before attachment of the glycyl ester.

5.2.6 Evaluation of the self-resistant Cyp51, *rstn2*

We also wished to evaluate the proposed resistance Cyp51 gene, *rstn2*. In *A. nomius*, there are an above average number of Cyp51 copies at three. Phylogenetic analysis of these copies of Cyp51 shows that each copy from *A. nomius* clade into the different fungal Cyp51 variants (Figure 28. The scaffold292 copy of Cyp51 is closest to Cyp51A and the scaffold 229 copy of Cyp51A. The cluster copy of Cyp51, *rstn2*, on the other hand, is the more distant relative, closest in homology to the rare Cyp51C copy found in some strains, such as *A. flavus*. A sequence alignment shows that *rstn2* contains a N360 residue that is usually the location of a conserved histidine. The location of this residue is in the SRS5 section[202], near one of the

substrate binding site in the catalytic pocket. The drastic change from the positively charged histidine at this position to the polar asparagine residue may be clue to an ability of *rstn2* to be invulnerable to Cyp51 inhibitors. To investigate the antifungal resistance of *rstn2*, we built plasmids that contained the *rstn2* and the native housekeeping copy of Cyp51 from yeast into pxp318, a low copy yeast vector (CEN/ARS), as overexpression of normal Cyp51 genes have been shown to induce fungicidal resistance.[203, 204] The genes were all under the expression of the constitutive TEF1 promoter. We obtained a heterozygous knockout strain of Cyp51 of the yeast

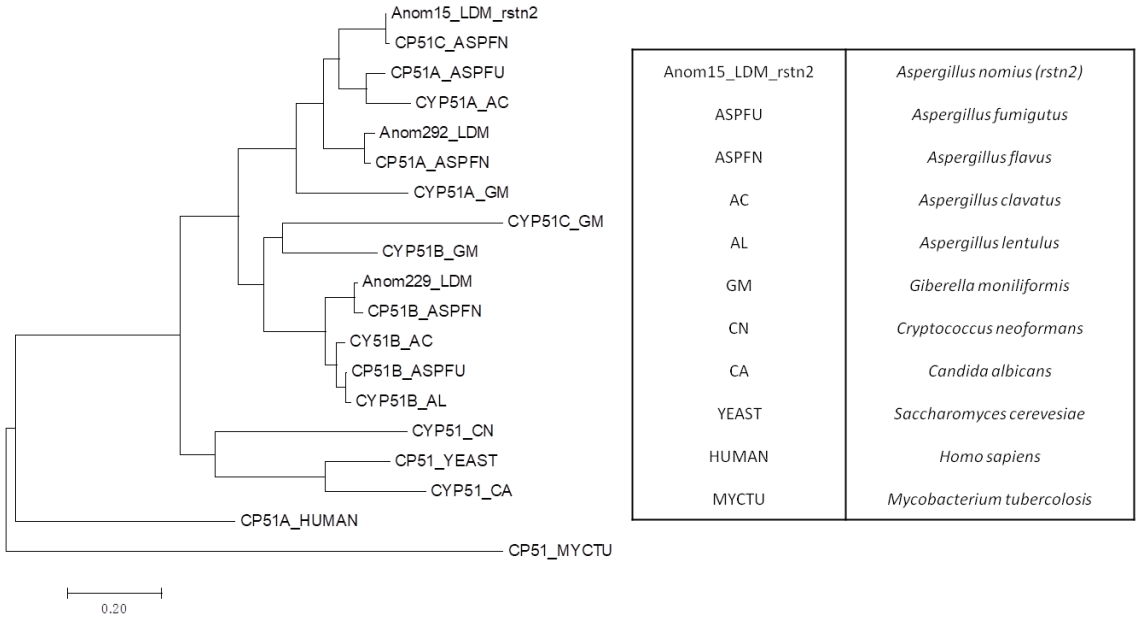


Figure 28. Phylogenetic tree of Cyp51 from different species
Cyp51 of different organisms in different kingdoms is shown. In fungi, there are usually two copies of Cyp51 in the forms of Cyp51A and Cyp51B. Some fungi contain a third Cyp51C copy, such as *A. nomius*

strain BY4743ΔYHR007C from the Stanford Genome Technology Center. These strains were supplemented with the plasmids and the growth curves of these strains of yeast were tested with different antifungal compounds, amphotericin B (amphB) and Fluconazole (FCZ) over 48 hours.

AmphB and FCZ are known antifungal drugs with different modes of action. AmphB works by binding ergosterol while FCZ is a known inhibitor of Cyp51. The effects of both were

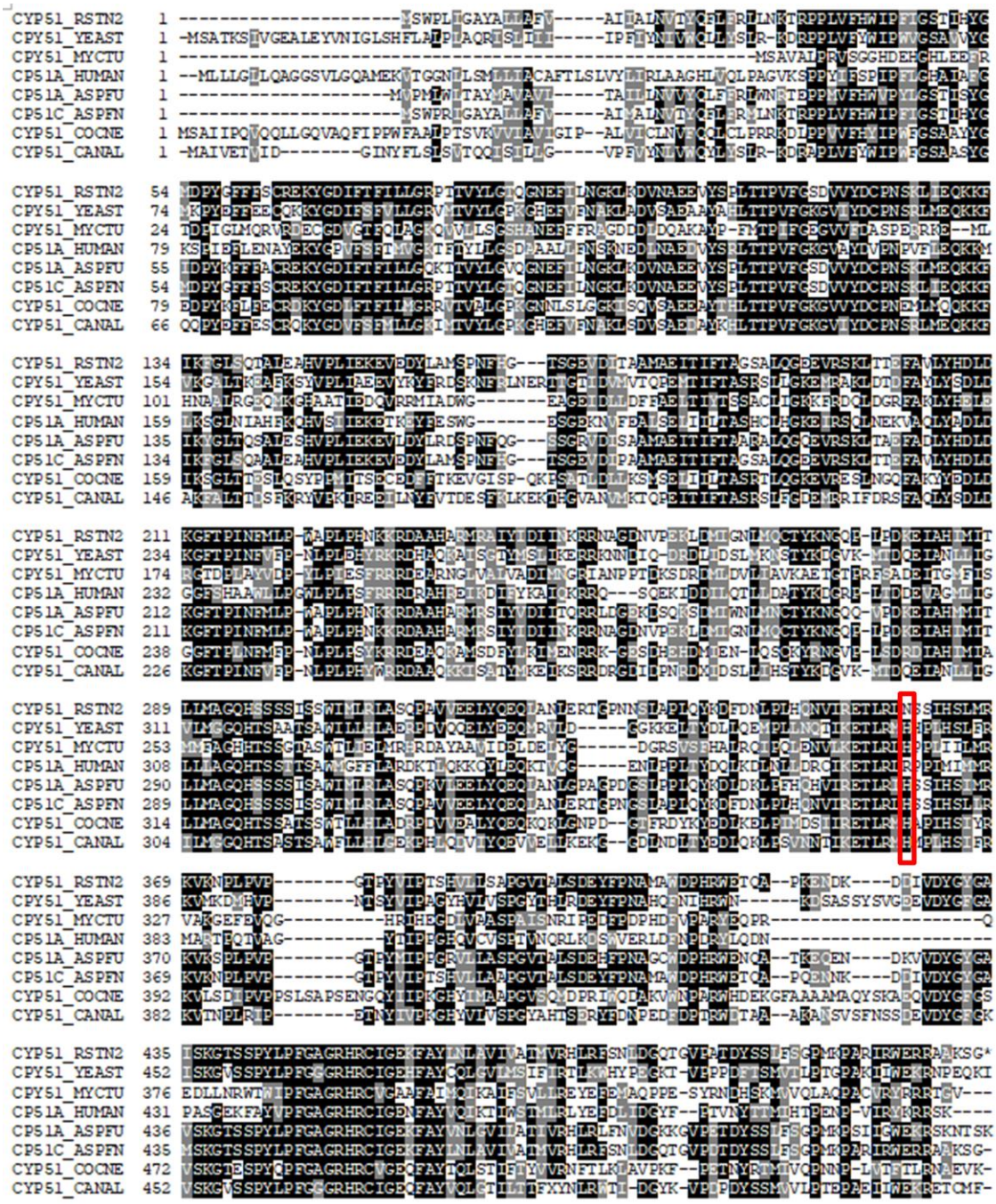


Figure 29. Sequence Alignment of various Cyp51

Cyp51 of different organisms in different kingdoms is shown. In *A. nomius*, the *rstn2* copy contains an Asparagine at N360 position, where this is usually a conserved histidine in other species.

prevalent as growth inhibition was seen in the strains containing only the empty vector pxp318. However, the treatment of FCZ had varying effects on the different strains of yeast. Yeast harboring *rstn2* showed recovery in growth around 20 hours while yeast harboring an additional

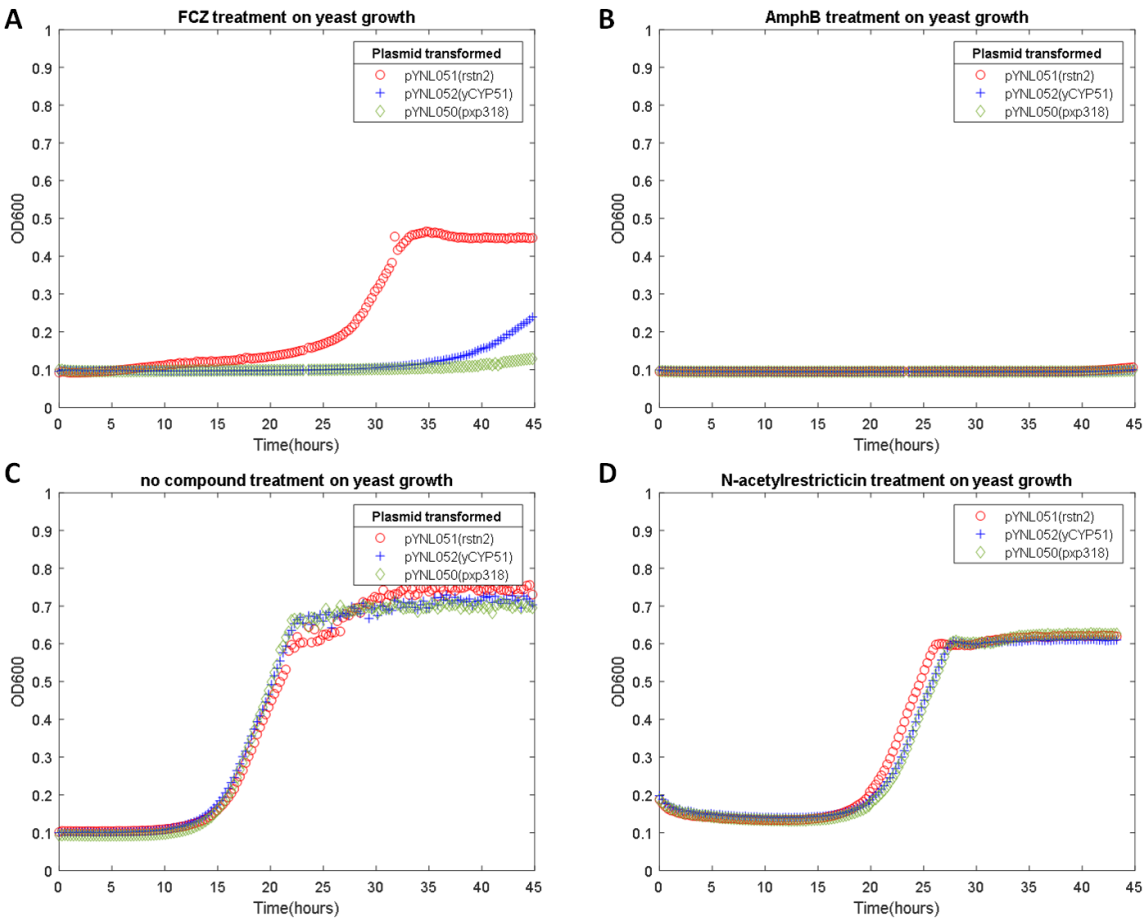


Figure 30. Effects of antifungal compounds on yeast strains
Yeast Cyp51 (yCyp51), *rstn2*, and pxp318 (empty vector) were transformed into yeast and tested for growth inhibition under various compounds. Amphotericin B (**A**), Fluconazole (**B**), **18** (**D**), and the negative control (no compound added) (**C**) were used to test growth inhibition.

copy of the native yCyp51 on the plasmid showed little signs of recovery past 35 hours of growth. These results show the ability of *rstn2* to have resistant properties against inhibitors of Cyp51. On the other hand, AmphB as a positive control caused all the yeast strains to not grow. Compound **18** was tested as well, but as was seen in previous studies, any antifungal effects are

very limited if any at all.[197] This supports the hypothesis that **18** is a metabolized form of the natural product as a way for the *A. nidulans* host to detoxify the antifungal restricticin.

5.2.7 A novel Cyp51 inhibiting gene cluster found in *Apiospora montagnei*

The TGIF results for Cyp51 targeted cluster searches also revealed another cluster found in *Apiospora montagnei* that was similar to the *rstn* cluster in *A. nomius*. We designated this new cluster the *Apm* cluster (Figure 31A). This cluster has very similar features to the one in *A. nomius*, including a single module NRPS, two reductive enzymes, an FMO, a Cyp51 gene, and

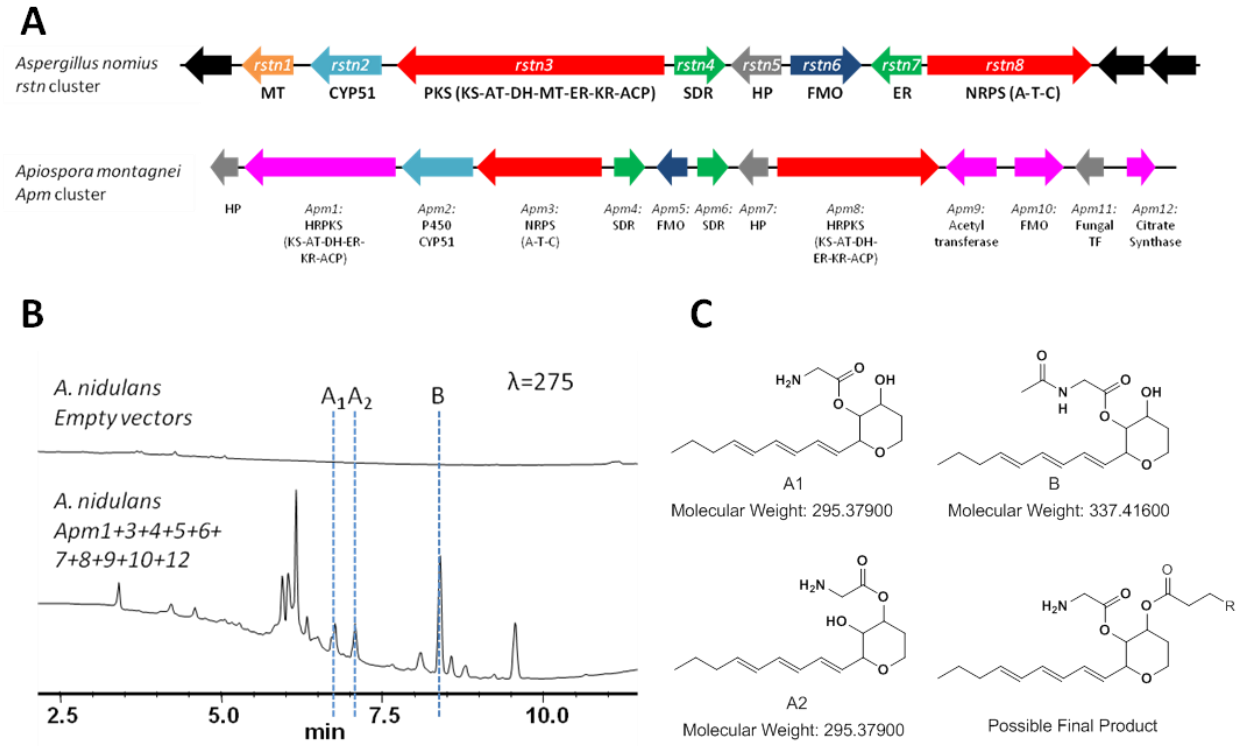


Figure 31. The *Apiospora montagnei Apm* cluster
TGIF showed the another similar cluster to the *rstn* cluster that contained a Cyp51 gene. **A**. Comparison of the *Apm17* and *rstn* clusters. Similar genes are coded in the same color. Unique genes in *Apm17* are coded in magenta. **B.** Introduction of all the proposed genes in the cluster into *A.nidulans* shows an increase in many metabolites. **C.** Proposed compounds for some of the compounds found based on the m/z values detected.

a HRPKS. However, this cluster is different in that it has multiple additional genes including a second FMO, a second HRPKS, an acetyltransferase, and homocitrate synthase. The *Apm* cluster also lacks the O-methyltransferase (*rstn1*) and the two HRPKS enzymes lack methyltransferase domains. This implies that the polyketide product made by these enzymes may form a backbone scaffold similar to that of chaunopyrone A (Figure 24) that has been isolated from fungal cultures.

Introduction of the proposed genes in this cluster into *A. nidulans* yielded a great deal of new metabolites (Figure 31B). These metabolites show a UV profile with a $\lambda_{max} = 275$, much like the UV profiles of the restricticin related compounds. Based on the proposed genes in the cluster, there were possible intermediates that we believed could be found. Some of these masses could be identified by their m/z values in the metabolic profile: **A_1**, **A_2**, and **B** (Figure 31C). As there is no O-methylation protection of the hydroxyl group, it is possible that the NRPS (*Apm3*) is capable of forming the glycyl ester at both hydroxyl positions to form **A_1** and **A_2. B** would be the expected *A. nidulans* metabolized N-acetylated compounds found. Since the O-methyltransferase does not protect the hydroxyl group in this cluster, the additional acetyltransferase *Apm9* may deliver a linear polyketide chain made by *Apm1* to furnish a final product similar to what is shown in Figure 31C. If the new metabolites are similar to the structure proposed, these would represent novel Cyp51 inhibitor natural products. Further characterization of these new metabolites need to be performed to confirm these results.

5.3 Conclusions

We have identified and activated the BGC of restricticin from the fungi, *A. nomius*, which is not a known producer of this Cyp51 inhibitor natural product using targeted genome

mining with the TGIF algorithm. Introduction of the genes in the pathway into our heterologous host *A. nidulans* resulted in the unexpected production of compounds **18-21.** Compound **18-21** are derivatives of **15**, likely metabolized by the heterologous host to detoxify the antifungal restricticin. All these compounds find a way to protect the free glycyl amine which is proposed to be the molecular warhead that inhibits substrate binding of molecular oxygen to the heme prosthetic group of the Cyp51. Compound **18** acetylates the free amine, while compounds **19-21** take a more creative approach to detoxify restricticin by capturing the amine group into the isoindolinone core of apsernidine, an *A. nidulans* metabolite that is produced when the cell membrane integrity of the fungi is compromised. We also evaluated the biosynthetic steps that lead up to restricticin, including the functionalization of the diol on the tetrahydropyran ring system. From *in vitro* assays, we see that the timing of the two functionalization steps are important in the biosynthesis, with the O-methylation of the C3 hydroxyl being a essential step before the glycyl ester can be formed by the NRPS on the C4 hydroxyl. The NRPS specifically activates glycine and no other amino acid analogs, likely due to the importance of this chemical group as the warhead of the molecules. Evaluation of the Cyp51 in the cluster, *rstn2*, also shows that it has resistant capabilities against commercial Cyp51 inhibitors. Finally, through the TGIF algorithm, we were also able to identify another similar cluster to that of restricticin, which is likely to produce several novel natural product inhibitors of the Cyp51.

5.4 Materials and Methods

Strains and culture conditions

Curvularia lunata (Wakker) Boedijin var. *lunata* anamorph (MF5573) was purchased from ATCC® (74067™). *Curvularia lunata* was grown on Difco™ PDA (Potato Dextrose Agar) Plates from BD biosciences. *Aspergillus nomius* was obtained through the NRRL (13137) fungal

collection. *Aspergillus nomius* was grown on Difco™ PDA (Potato Dextrose Agar) Plates from BD biosciences for activation. *Apiospora montagnei* was obtained through NRRL (25634) fungal collection and activated using PDA Plates. Aspergillus *nidulans* A1145 was purchased from the Fungal Genetics Stock Center and used for heterologous expression of genes from *C. lunata*. *Escherichia coli* strain XL1 Blue was used for cloning. *Saccharomyces cerevisiae* strain BJ5464 was used for homologous recombination of DNA fragments to assemble the vectors used in heterologous expression.

A. nomius and _A. montagnei_ gDNA extraction, RNA extraction, and RTPCR

The Zymo ZR Fungal /Bacterial DNA Microprep™ kit was used to extract gDNA from *A. nomius* and *A. montagnei*. The Invitrogen Ribopure™ kit was used to extract RNA from *A. nomius*. Superscript® III Reverse Transcriptase Kit from Life Technologies was used to synthesize cDNA from the RNA extracted from *A. nomius* and *A. montagnei*.

Genome sequencing, assembly and biosynthetic gene cluster prediction

The genome sequence for *A. nomius* was obtained through the NCBI database, genbank accession code: GCA_001204775.2. The genome sequence for *Apiospora montagnei* was obtained through the JGI portal database. Biosynthetic gene cluster prediction was done through using online services from NCBI BLAST, NCBI conserved domain search, 2ndfind, and Softberry (FGNESH) gene prediction.

Plasmid construction for heterologous expression

Plasmids pYTU, pYTP, pYTR were used as vectors to insert genes which contain auxotrophic markers for uracil (pyrG), pyridoxine (pyroA), and riboflavin (riboB), respectively.[176] Genes to be expressed were amplified through Polymerase Chain Reaction (PCR) using the gDNA of *A.nomius* and *A. montagnei* as a template. The PCR products and the

corresponding backbone digested with *Pac*I and *Swa*I were assembled with the Frozen-EZ Yeast Transformation II Kit™ (Zymo Research) by using yeast homologous recombination with BJ5464-NpgA, which contains a copy of *A. nidulans* phosphopantheteinyl transferase gene *npgA* integrated in the chromosome.[131] *E. coli* vectors were assembled through digestion ligation of the pet28A bector backbone. Genes to be expressed were amplified through PCR using *A. nomius* and *A. montagnei* cDNA as template. Genes to be expressed in yeast were amplified through the cDNA of *A. nomius* as template and gDNA from *S. cerevesiae*. Vectors were assembled using pxp318 as backbone through yeast homologous recombination.

Genetic transformation and heterologous production in *A. nidulans*

Protoplasts were generated by scraping spores from a solid CD Medium (10 g/L glucose, 50 mL/L 20x Nitrate Salts, 1mL/L Trace elements, 20% agar) Plate. The spores were transferred to 25 mL of liquid CD minimal medium and incubated for 12-13 hours at 37°C at 250 rpm. After incubation, the germlings were collected and washed with 10 mL of Osmotic Medium (1.2M $MgSO_4$, 10 mM $NaPO_4$) twice. The germlings were then transferred into 10 mL of Osmotic Medium containing 30 mg of Lysing Enzyme from *Trichoderma* and 20 mg of Yatalase. The culture was incubated for 12 hours at 28°C at 80 rpm. The cells were poured into a 30 mL Corex tube and overlayed with 10 mL of Trapping Buffer (0.6 M Sorbitol, 0.1 M Tric-HCl). The tube was centrifuged at 5000 RPM. The protoplasts were then removed from the interface of the two buffers and transferred to sterile tubes. 2x volume of STC Buffer (1.2 M Sorbitol, 10 mM $CaCl_2$, 10 mM Tric-HCl) was added to the protoplasts. DNA and 60% PEG4000 solution were added to the protoplast solution and incubated at room temperature for 20 min. The cells were then plated onto solid CD-Sorbitol Medium (10 g/L glucose, 50 mL/L 20x Nitrate Salts, 1mL/L Trace elements, 20% agar, 1.2 M Sorbitol). After transformants appeared on the plates, the spores were

restreaked onto solid CD-ST production medium at 28°C for 4 days (20g/L Starch, 20g/L Peptone, 50mL/L Nitrate Salts, 1mL/L Trace elements) and solid CD medium (20g/L Glucose, 50mL/L Nitrate Salts, 1 mL/L Trace elements). For isotope feeding studies, 1 mg/mL of deuterium labeled glycine were added to the solid CD-ST before streaking cells. Deuterium labeled precursors used for feeding were purchased from Cambridge Isotope Laboratories.

Sample analysis of *A. nidulans* transformants

A. nidulans transformants were grown on CD and CD-ST for 2-4 days at 28℃ for small scale analysis. Samples were extracted using 1 mL of acetone. After centrifugation, the supernatant was then dried and resuspended into equal volume methanol before injection for LC-MS analyses. LC–MS analyses were performed on a Shimadzu 2020 EV LC–MS (Kinetex 1.7 m C18 100 Å, LC Column 100 × 2.1 mm) using positive-and negative-mode electrospray ionization with a linear gradient of 5–95% acetonitrile MeCN/H_2O with 0.5% formic acid in 15 min followed by 95% MeCN for 3 min with a flow rate of 0.3 mL/min.

Compound isolation and structure elucidation

For isolation of **13, 14,** and **17** 8 L of PDB liquid medium was made in 2L flasks. For isolation of **18, 19, 20,** and **21,** 4L of CD liquid medium was made in 2L flasks. Spores from solid CD media plates were restreaked onto the production medium and grown in 28°C for 3 days. The fungal culture was filtered through cheesecloth to separate the cell mass from the media. The media fraction was extracted with ethyl acetate while the pellet was exctracted with acetone. The acetone fraction was evaporated, leaving an aqueous solution of extracted metabolites. This aqueous fraction was then extracted with ethyl acetate. Both fractions were then combined and dried *in vacuo* and resuspended in methanol, and fractionated using Sephadex a CombiFlash® System, and semi-prep reverse phase HPLC sequentially. The CombiFlash

system (Teledyne) uses normal phase column chromatography with hexane and acetone as the mobile phase with a gradient of 30-70% of acetone.

HPLC fractionation used a semi-preparative C18 column of Kinetics New column, 5m, 10 × 250 mm with an isocratic method of 50%, 71%,and 38% acetonitrile/water with 0.1% formic acid for 45 min at 4 mL/min for **13, 14,** and **17**, respectively. The collected fractions were analyzed using LC-MS with a linear gradient of 5–95% acetonitrile MeCN/H_2O with 0.5% formic acid in 15 min and desired fractions were pooled. For elucidation of chemical structures, 1D and 2D NMR spectra were obtained on Bruker AV500 spectrometer at the UCLA Molecular Instrumentation Center. High resolution mass spectra were obtained from Thermo Fisher Scientific Exactive Plus with IonSense ID-CUBE DART source at the UCLA Molecular Instrumentation Center.

Protein expression, purification, and enzymatic assays

To express and purify *rstn1* and *rstn8*, primers were used to amplify the transcripts from *A. nomius* cDNA. The inserts were digested and ligated into digested pet28A vector which contained C-terminal His tag. The vectors were then transformed into *E. coli* BL21 and BAP1 for *rstn1* and *rstn8*, respectively. 4L of LB medium with kanamycin antibioitic added were used to grow the culture at 37°C to an OD of 0.6 before cooling and inducing with 0.1 mM isopropyl thio-β-D-galactoside (IPTG) overnight at 16°C. The cell pellet was harvested by centrifugation (3500 rpm, 10 mins), and it was then suspended in 100 mL of a buffer containing 50 mM NaH2PO4, 150 mM NaCl, and 10 mM imidazole at pH 8.0. The suspended bacterial cells were lysed using sonication and the cellular debri was removed using high-speed centrifugation (17,000 rpm, 1 hr). The enzyme of interest was then purified from the supernatant using Ni-NTA agarose affinity chromatography to near homogeneity. The purified protein was concentrated,

exchanged into a buffer container 50 mM NaH2PO4 and 50 mM NaCl at pH 8.0, aliquoted and then flash frozen in liquid nitrogen. SI-8 For the protein expression and purification of the MatB enzyme, the protocol detailed by Ma et al was followed.[131] The purified protein was concentrated, exchanged into a buffer container 50 mM NaH2PO4 and 50 mM NaCl at pH 8.0, aliquoted and then flash frozen in liquid nitrogen.

Assays were performed using 10 mM of NaH2PO4 buffer. For *rstn1* reactions, 50 uM of enzyme, 200 uM of **14**, and 500 uM of SAM was added and incubated at room temperature for 1 hour, taking intermittent time points. For *rstn8* reactions, 5 uM of *rstn8* was incubated with 400 uM coenzyme A, 8 mM MgCl2, and 50 uM of purified npga for 1 hour. Then 4 mM glycine, 4 mM ATP, 8 mM MgCl2, and 20 uM of **15** was added to the reaction and incubated overnight.

Yeast Growth Curves

S. cerevisiae was transformed with the various plasmid harboring *rstn2*, yeast Cyp51, and empty pxp318 backbone. The yeast was grown up in Uracil dropout media overnight. The OD600 values were measured and diluted to a starting OD600 value of 0.005. Samples were grown in 96 well plates. Compounds to be tested against (FCZ, AmphB, and **18**) were dissolved in 1 ul of DMSO and fed at final concentrations of 50 ug/ml, 2ug/ml, and 200 ug/ml respectively. The negative control was performed with just adding 1 ul of DMSO into the drop out media. FCZ was purchased from RPI and amphotericin B was purchased from Alfa Aesar. The total 96 well plate volume was added to 100 ul. The 96 well plates were put in the TECAN M200 plate reader and grown at 28°C, with shaking. OD600 values were taken every 840 seconds. The 96 well plate was grown for 48 hours.

6. Final Conclusions

With the improvement of genome sequencing, we now have access to the entire repertoire of microbial compounds. Targeted genome mining is a recently developed method that utilizes the new wealth of genome information to discover natural product desired bioactivity. Here we demonstrated two examples of how targeted genome mining can be an effective strategy in pinpointing a specific BGC of interest.

The first utilizes targeted genome mining to identify the biosynthesis of a molecule with known activity. The squalene synthase inhibitor, zaragozic acid A, is of great interest due to its complex chemical structure and potent bioactivity. To elucidate the biosynthesis of this molecule, the targeted genome mining approach was utilized to search for squalene synthase containing clusters in the natural producer, *Curvularia lunata*. This allowed for the identification and verification of the zaragozic acid A biosynthetic gene cluster through heterologous expression of a benzoyl alkylcitrate intermediate of the pathway.

The second example serves as a more general approach in trying to discover novel sterol pathway inhibitors. Using the TGIF algorithm, we were able to identify Cyp51 containing fungal gene clusters of interest. Cyp51 is an important drug target utilized by many commercial antifungal azole drugs. This strategy identified the restricticin biosynthetic gene cluster in *A. nomius*. Elucidation of the biosynthetic steps through *in vivo* and *in vitro* methods gave us insight into how the restricticin is furnished. An analysis of the Cyp51 gene in the BGC also showed that *rstn2* has azole resistant properties. With the rest of the TGIF results, we were also able to identify a Cyp51 containing biosynthetic gene cluster in *Apiospora montagnei* with the potential to produce a novel Cyp51 inhibitor.

These results prove that the targeted genome mining methodology is an effective one at searching for sterol pathway drugs. With further developments of search algorithms and the growth of biosynthetic knowledge of secondary metabolism, we grow closer to drawing the connections between natural compounds and genetic information. This offers a huge potential for us to truly tap into the natural chemical resources from all of life. The implications of such potential could be the solution to many therapeutic problems and hopefully the work done here serves as a steppingstone for future developments in this new era of natural product discovery.

7. Appendices

Supplementary Tables

Supplementary Figures

Scripts

Table S1. Primers used in this study

Primer Name	Sequence (5' to 3')
Amyb-F-1-Orf11	CAATGGAGAATCTGCCATAAATGCCTTCTGTGGGGTTTATT
Amyb-F-1-Orf16	TGATCTTCACGACATGATAGATTAAAGGTGCCGAACGAGC
Orf12-F-1-AmyB	AATAAACCCCACAGAAGGCATTTATGGCAGATTCTCCATTGG
Orf12-R-1-pytu	GGACATACCCGTAATTTTCTGGGCATTTCAATGAACCGTG
Orf17-F-1-glaA	GCATCATTACACCTCAGCATGGCTACCGTCAACGGCGCAG
Orf17-R-1-Amyb	CGTTCGGCACCTTTAATCTATCATGTCGTGAAGATCATAG
gpdA-F_1	TCCCCTCCCAGCTCCTCCC
PYTR-4-R-14	TCCCAGGGATAGGTAGGTATGTCTGG
PYTR-5-F-14	GGGAGATGGGATCAAACACAGCAC
PYTR-5-R-14	GGTATCATCGAAAGGGAGTCATCCACTCATGGGTTAGTAAAAAAGTTCATGAATGGCTTC
Orf10-F-1-Trpc	CTTACCTATTCTACCCAAGCATATGTCCTCGCGTCCGTTG
Orf10-R-1-pytr	CATCGAAAGGGAGTCATCCAATTTCTCTTCTGCACCCAGAG
Trpc-F-1-Orf10	ACTTTTTTACTAACCCATGAGTCGACAGAAGATGATATTG
Trpc-R-1-Orf10	CAACGGACGCGAGGACATATGCTTGGGTAGAATAGGTAAG
NL-Amyb-F_Orf6	CAAGGACGGACTCGGCCCCATGATTAAAGGTGCCGAACGAGC

NL-Amyb-R_Orf17	TTGGTATCGGCTTGTGTATCATAAATGCCTTCTGTGGGG
Orf11-F-1-Amyb	CTTCTCTGAACAATAAACCCCACAGAAGGCATTTATGGATTTTCCCGGGGACTC
Orf11-R-1-pytpa	AGACCCAACAACCATGATACCAGGGGATTTAAATGCTGGTACCTTCGTCGCAG
NL-Amyb-F_Orf6	CAAGGACGGACTCGGCCCCATGATTAAAGGTGCCGAACGAGC
NL-Amyb-R_Orf17	TTGGTATCGGCTTGTGTATCATAAATGCCTTCTGTGGGG
Orf13-F-1-Amyb	TTCTCTGAACAATAAACCCCACAGAAGGCATTTATGTGCTTGCTTAGTATGCGATTTACC
Orf13-R-1-pytp	AGACCCAACAACCATGATACCAGGGGATTTAAATGACAGCAGAGCAGCAGCG
gBlocks_F	ATAAGATCTGCGTAAGCTCCCTAATTGGCC
gBlocks_R	ATATTAATTAAGAGCCAAGAGCGGATTCCTC
ST_SOE_HR1_F	ATTCCCTGTGGCGTGGTGAC
ST_SOE_HR1_R	CGTTAGGGCCATTATGACAGATGCCCTCTTGCTATAGCGC
ST_SOE_HR2_F	CTGTCATAATGGCCCTAACG
ST_SOE_HR2_R	CGACAACACCGTCCATGGCG
EM_SOE_HR1_F	CATCTGGAGGAGTGGAATTT
EM_SOE_HR1_R	CAACGTGTTGGTGTAGGAGGTCGGGCGGTGCACCAACGGC
EM_SOE_HR2_F	CCTCCTACACCAACACGTTG
Anom15-Orf3-F-1-gpda	ACCCCGCCACATAGACACATCTAAACAatgcagccccataatccctatg
Anom15-Orf3-F-2	gtttcccggaacagtcgtatg
Anom15-Orf3-R-1	gcgtcgcatggataacctatc
Anom15-Orf3-R-2	aagctcacatgtattcctggagcaaacggtatcaggggatggaagagac
Anom15-Orf4-F	CATACAGAACACTTCAAACAATCGCAAAAatggattgcgatccataccagc
Anom15-Orf4-R-pytu	CAGTGGAGGACATACCCGTAATTTTCTGcctggacccgccccatg
Anom15-Orf1-F-PE	TGATCTAACAACTTCTAGTAAACCGCAATCatgtctcttcagccaacggc
Anom15-Orf1-F-PYTR	cattaccccgccacatagacacatctaaacaatgtctcttcagccaacggc
Anom15-Orf1-R-pytr	TAAAGGGTATCATCGAAAGGGAGTCATCCAgatcggcttgtggaagtcgc
Anom15-Orf7-F-PO	tGCATACAGAACACTTCAAACAATCGCAAAAatgaaggccatcatcagcgtaag
Anom15-Orf7-F-PYTR	cattaccccgccacatagacacatctaaacaatgaaggccatcatcagcgtaag
Anom15-Orf7-R-PE	attccaaccttgggaagccctggacgaatccgacatgtatttagacggatgtcag
Anom15-Orf7-R-pYTR	TAAAGGGTATCATCGAAAGGGAGTCATCCAcgacatgtatttagacggatgtcag
Anom15-Orf8-F-gpda	cattaccccgccacatagacacatctaaacaatgtcacattccagccattattcc
Anom15-Orf8-R-PE	attccaaccttgggaagccctggacgaatccgccatacatggctataaatttggc
Anom15-Orf8-R-PO	cagtaagctcacatgtattcctggagcaaacgccatacatggctataaatttggc
Anom15-Orf2-F-gpda	TACCCCGCCACATAGACACATCTAAACAatgtcctggcctttgattggg
Anom15-Orf2-R-PO	agtaagctcacatgtattcctggagcaaactgtcgggccacggataaac
Anom15-Orf4-R-pytu	CAGTGGAGGACATACCCGTAATTTTCTGcctggacccgccccatg
Anom15-Orf5-F-PO	CATACAGAACACTTCAAACAATCGCAAAAatgcacccagaggcatgg
Anom15-Orf5-F-PYTR	TACCCCGCCACATAGACACATCTAAACAatgcacccagaggcatgg
Anom15-Orf5-R-PE	ttccaaccttgggaagccctggacgaatcctagcctttaccagggtgtataattcc
Anom15-Orf5-R-Pytr	GATGAGACCCAACAACCATGATACCAGGGGctagcctttaccagggtgtataattcc
Anom15-Orf6-F-PE	TCTAACAACTTCTAGTAAACCGCAATCatgtacgacgtgatagtcatcgg
Anom15-Orf6-F-PYTR	TACCCCGCCACATAGACACATCTAAACAatgtacgacgtgatagtcatcgg

Anom15-Orf6-R-pytp	TGATGAGACCCAACAACCATGATACCAGGGGgctcaggatgtggactactagac
Anom15-Orf7-R-PE	attccaaccttgggaagccctggacgaatccgacatgtatttagacggatgtcag
Anom15-Orf1-F-LIC	TACTTCCAATCCAATGCAatgtctcttcagccaacggc
Anom15-Orf1-R-LIC	TTATCCACTTCCAATGTTATTActaaaaatttgcagacaggccg
Anom15-Orf8-F-XW55	ataaaagataatattctacttttgctcccatgtcacattccagccattattcccc
Anom15-Orf8-R-XW55CHis	ttagtgatggtgatggtgatgcacgtgaaccatgtcttccgctccac
Anom15-Orf2-F-tef1p	TCTAATCTAAGTTTTAATTACAAAACTAGTatgtcctggcctttgattggg
Anom15-Orf2-R-cyc1t	AGCGTGACATAACTAATTACATGACTCGAGttatcccgattttgcagcccgac
cyp51-tef-R	TGACATAACTAATTACATGACTCGAGTTAGATCTTTTGTTCTGGATTTCTCTTTTCCCAG
cyp51-tef1-F	TCTAATCTAAGTTTTAATTACAAAACTAGTATGTCTGCTACCAAGTCAATCGTTG
ApM17-Orf1-F-1-mbfA	GGAGCCAGGCACACTGGTGGCCCTGCCACCATGCAATCAACACGTCCAATTCC
ApM17-Orf1-F-1-PO	caagtGCATACAGAACACTTCAAACAATCGCAAAAATGCAATCAACACGTCCAATTCC
ApM17-Orf1-F-2	CTTTTGCCTCAATAACCGACATCC
ApM17-Orf1-R-1	GAAACTCAACTGGTGGTGTCCG
ApM17-Orf1-R-2-PEgpda	ggtccccaatattccaaccttgggaagccctggacgaatcCTTATGGAAAGAGCCCAGGTGC
ApM17-Orf1-R-2-PYTP	GATGAGACCCAACAACCATGATACCAGGGGCTTATGGAAAGAGCCCAGGTGC
Apm17-Orf2-F-PEgdpa	ctcttatacaTGATCTAACAACTTCTAGTAAACCGCAATCATGGCATCCAACATTCTGGG
Apm17-Orf2-R-PYTP	TGAGTAGGAGTGATGAGACCCAACAACCATGATACCAGGGGGCTCCGTCCCAAAACAGTC
ApM17-Orf4-R-coxA	CCTCGAGGCCTGGGGGCTGGATCAGGCATTCCCCCCTTTCCTTATAAATCCACTATC
ApM17-Orf4-R-PYTU	CACAGTGGAGGACATACCCGTAATTTTCTGCCCCCCTTTCCTTATAAATCCACTATC
ApM17-Orf5-F-PYTP	CCATTACCCCGCCACATAGACACATCTAAACAATGGACGTGTACAAAGTTATTATCGTGG
ApM17-Orf5-R-coxA	CTCGAGGCCTGGGGGCTGGATCAGGCATTAGTATGTAGAGTCAAATCGAGTTTAGCTCAG
ApM17-Orf6-R-coxA	TCCTCGAGGCCTGGGGGCTGGATCAGGCATTATCGGATTGTTATTGTATTCCTGTTCTTGGTG
ApM17-Orf6-R-PYTR	TAAAGGGTATCATCGAAAGGGAGTCATCCATCGGATTGTTATTGTATTCCTGTTCTTGGTG
ApM17-Orf7-F-coxA	TGCCGTCATTGCAACCCACCCACCAGGACAATGCCTTTTCTTTGGAGACTAGTTTTTCC
ApM17-Orf7-R-mbfA	GGCTCCGGGTGATCAAAGACGAACGCTACAGCCAGCCACCATGGAACTTTTTC
ApM17-Orf7-R-PO	agaatcagtaagctcacatgtattcctggagcaaaGCCAGCCACCATGGAACTTTTTC
ApM17-Orf7-R-PYTP	GATGAGACCCAACAACCATGATACCAGGGGGCCAGCCACCATGGAACTTTTTC
ApM17-Orf3-F-PYTR	accattaccccgccacatagacacatctaaacaATGACTCCCCTTATCGAGCCC
ApM17-Orf3-R-mbfA	GGCTCCGGGTGATCAAAGACGAACGCTACATTTGAGCTTGTTTGGTCAAAGAAAACGG
ApM17-Orf3-R-PO	atcagtaagctcacatgtattcctggagcaaaTTTGAGCTTGTTTGGTCAAAGAAAACGG
ApM17-Orf6-F-mbfA	TGGAGCCAGGCACACTGGTGGCCCTGCCACCATGGCCTTCCTCCCCGAG
ApM17-Orf6-F-PO	caagtGCATACAGAACACTTCAAACAATCGCAAAAATGGCCTTCCTCCCCGAG
ApM17-Orf6-R-coxA	TCCTCGAGGCCTGGGGGCTGGATCAGGCATTATCGGATTGTTATTGTATTCCTGTTCTTGGTG
ApM17-Orf6-R-PE	tccaaccttgggaagccctggacgaatcATCGGATTGTTATTGTATTCCTGTTCTTGGTG
ApM17-Orf6-R-PYTR	TAAAGGGTATCATCGAAAGGGAGTCATCCATCGGATTGTTATTGTATTCCTGTTCTTGGTG
ApM17-Orf8-R-2-mbfA	GGCTCCGGGTGATCAAAGACGAACGCTACAACGCCGCAGCACTCGTAG
ApM17-Orf9-F-coxA	CTGCCGTCATTGCAACCCACCCACCAGGACAATGGAGGCACTGACAGCTTTC
ApM17-Orf9-F-PE	tcttatacaTGATCTAACAACTTCTAGTAAACCGCAATCATGGAGGCACTGACAGCTTTC
ApM17-Orf9-R-cox	TCCTCGAGGCCTGGGGGCTGGATCAGGCATTGTGGAGAAAGGGAAATTGAATAGGTATAG
ApM17-Orf9-R-mdhA	CATCCTCGGATGTAGGACCCCCATACACGCGTGGAGAAAGGGAAATTGAATAGGTATAG

ApM17-Orf9-R-PE	ccaatattccaaccttgggaagccctggacgaatcGTGGAGAAAGGGAAATTGAATAGGTATAG
ApM17-Orf9-R-pytr	CTAAAGGGTATCATCGAAAGGGAGTCATCCAGTGGAGAAAGGGAAATTGAATAGGTATAG
ApM17-Orf12-F-cox	TTGTCTGCCGTCATTGCAACCCACCCACCAGGACAATGAAAACACCTAACGGCATTGGAG
ApM17-Orf12-F-mdhA	GTATAGATTTGTCACAATCCCAAATTTCACCATGAAAACACCTAACGGCATTGGAG
ApM17-Orf12-F-PE	atacaTGATCTAACAACTTCTAGTAAACCGCAATCATGAAAACACCTAACGGCATTGGAG
ApM17-Orf12-R-PYTR	CTAAAGGGTATCATCGAAAGGGAGTCATCCACAGGTCATGCATGCAACGATTG
ApM17-ORf4-F-mbfA	GGAGCCAGGCACACTGGTGGCCCTGCCACCATGAAAGCCATTATCGTCACGGC
ApM17-ORf4-F-PO	caagtGCATACAGAACACTTCAAACAATCGCAAAAATGAAAGCCATTATCGTCACGGC
ApM17-Orf4-R-coxA	CCTCGAGGCCTGGGGGCTGGATCAGGCATTCCCCCCTTTCCTTATAAATCCACTATC
ApM17-Orf4-R-PYTU	CACAGTGGAGGACATACCCGTAATTTTCTGCCCCCCTTTCCTTATAAATCCACTATC
ApM17-Orf8-F-1	ATTACCCCGCCACATAGACACATCTAAACAATGGGTTCCATGGGCACCGATG
ApM17-Orf8-F-2	TGTAGACACGCAGCTAGACGAATCC
Apm17-Orf8-F-glaA-long	GAGAGCCTGAGCTTCATCCCCAGCATCATTACACCTCAGCAATGGGTTCCATGGGCACCG
Apm17-Orf8-F-gpda-long	GACTAACCATTACCCCGCCACATAGACACATCTAAACAATGGGTTCCATGGGCACCG
ApM17-Orf8-R-1	GTTGATGCTGGGCTTCCAGACG
ApM17-Orf8-R-2-mbfA	GGCTCCGGGTGATCAAAGACGAACGCTACAACGCCGCAGCACTCGTAG
ApM17-Orf8-R-2-PO	agaatcagtaagctcacatgtattcctggagcaaaACGCCGCAGCACTCGTAG
ApM17-Orf9-R-mdhA	CATCCTCGGATGTAGGACCCCCATACACGCGTGGAGAAAGGGAAATTGAATAGGTATAG
ApM17-Orf10-F-coxA	TGCCGTCATTGCAACCCACCCACCAGGACAATGAGTATGTTTGATGGTCAGGCG
ApM17-Orf10-R-mdhA	CATCCTCGGATGTAGGACCCCCATACACGCATCTAGATTGACATACAGAACCTCAAAGTG
ApM17-Orf10-R-PE	ccaatattccaaccttgggaagccctggacgaatcATCTAGATTGACATACAGAACCTCAAAGTG
ApM17-Orf13-F-mdhA	GTATAGATTTGTCACAATCCCAAATTTCACCATGGCGACTGATACCCCCC
ApM17-Orf13-F-PE	atacaTGATCTAACAACTTCTAGTAAACCGCAATCATGGCGACTGATACCCCCC
ApM17-Orf13-R-PYTU	CACAGTGGAGGACATACCCGTAATTTTCTGGGCTTTTTTTTTTTAGATAGGAAAGCGGC
Anom15-Orf3-F-1-gpda	ACCCCGCCACATAGACACATCTAAACAatgcagccccataatccctatg
Anom15-Orf3-F-2	gtttcccggaacagtcgtatg
Anom15-Orf3-R-1	gcgtcgcatggataacctatc
Anom15-Orf3-R-2	aagctcacatgtattcctggagcaaacggtatcaggggatggaagagac
Anom15-Orf4-F	CATACAGAACACTTCAAACAATCGCAAAAatggattgcgatccataccagc
Anom15-Orf4-R-pytu	CAGTGGAGGACATACCCGTAATTTTCTGcctggacccgccccatg
Anom15-Orf1-F-PE	TGATCTAACAACTTCTAGTAAACCGCAATCatgtctcttcagccaacggc
Anom15-Orf1-F-PYTR	cattaccccgccacatagacacatctaaacaatgtctcttcagccaacggc
Anom15-Orf1-R-pytr	TAAAGGGTATCATCGAAAGGGAGTCATCCAgatcggcttgtggaagtcgc
Anom15-Orf7-F-PO	tGCATACAGAACACTTCAAACAATCGCAAAAatgaaggccatcatcagcgtaag
Anom15-Orf7-F-PYTR	cattaccccgccacatagacacatctaaacaatgaaggccatcatcagcgtaag
Anom15-Orf7-R-PE	attccaaccttgggaagccctggacgaatccgacatgtatttagacggatgtcag
Anom15-Orf7-R-pYTR	TAAAGGGTATCATCGAAAGGGAGTCATCCAcgacatgtatttagacggatgtcag
Anom15-Orf8-F-gpda	cattaccccgccacatagacacatctaaacaatgtcacattccagccattattcc
Anom15-Orf8-R-PE	attccaaccttgggaagccctggacgaatccgccatacatggctataaatttggc
Anom15-Orf8-R-PO	cagtaagctcacatgtattcctggagcaaacgccatacatggctataaatttggc
Anom15-Orf2-F-gpda	TACCCCGCCACATAGACACATCTAAACAatgtcctggcctttgattggg

Anom15-Orf2-R-PO	agtaagctcacatgtattcctggagcaaactgtcgggccacggataaac
Anom15-Orf4-R-pytu	CAGTGGAGGACATACCCGTAATTTTCTGcctggacccgccccatg
Anom15-Orf5-F-PO	CATACAGAACACTTCAAACAATCGCAAAAatgcacccagaggcatgg
Anom15-Orf5-F-PYTR	TACCCCGCCACATAGACACATCTAAACAatgcacccagaggcatgg
Anom15-Orf5-R-PE	ttccaaccttgggaagccctggacgaatcctagcctttaccagggtgtataattcc
Anom15-Orf5-R-Pytr	GATGAGACCCAACAACCATGATACCAGGGGctagcctttaccagggtgtataattcc
Anom15-Orf6-F-PE	TCTAACAACTTCTAGTAAACCGCAATCatgtacgacgtgatagtcatcgg
Anom15-Orf6-F-PYTR	TACCCCGCCACATAGACACATCTAAACAatgtacgacgtgatagtcatcgg
Anom15-Orf6-R-pytp	TGATGAGACCCAACAACCATGATACCAGGGGgctcaggatgtggactactagac
Anom15-Orf7-R-PE	attccaaccttgggaagccctggacgaatccgacatgtatttagacggatgtcag
Anom15-Orf1-F-LIC	TACTTCCAATCCAATGCAatgtctcttcagccaacggc
Anom15-Orf1-R-LIC	TTATCCACTTCCAATGTTATTActaaaaatttgcagacaggccg
Anom15-Orf8-F-XW55	ataaaagataatattctacttttgctcccatgtcacattccagccattattcccc
Anom15-Orf8-R-XW55CHis	ttagtgatggtgatggtgatgcacgtgaaccatgtcttccgctccac
Anom15-Orf2-F-tef1p	TCTAATCTAAGTTTTAATTACAAAACTAGTatgtcctggcctttgattggg
Anom15-Orf2-R-cyc1t	AGCGTGACATAACTAATTACATGACTCGAGttatcccgattttgcagcccgac
cyp51-tef-R	TGACATAACTAATTACATGACTCGAGTTAGATCTTTTGTTCTGGATTTCTCTTTTCCCAG
cyp51-tef1-F	TCTAATCTAAGTTTTAATTACAAAACTAGTATGTCTGCTACCAAGTCAATCGTTG
ApM17-Orf1-F-1-mbfA	GGAGCCAGGCACACTGGTGGCCCTGCCACCATGCAATCAACACGTCCAATTCC
ApM17-Orf1-F-1-PO	caagtGCATACAGAACACTTCAAACAATCGCAAAAATGCAATCAACACGTCCAATTCC
ApM17-Orf1-F-2	CTTTTGCCTCAATAACCGACATCC
ApM17-Orf1-R-1	GAAACTCAACTGGTGGTGTCCG
ApM17-Orf1-R-2-PEgpda	ggtccccaatattccaaccttgggaagccctggacgaatcCTTATGGAAAGAGCCCAGGTGC
ApM17-Orf1-R-2-PYTP	GATGAGACCCAACAACCATGATACCAGGGGCTTATGGAAAGAGCCCAGGTGC
Apm17-Orf2-F-PEgdpa	ctcttatacaTGATCTAACAACTTCTAGTAAACCGCAATCATGGCATCCAACATTCTGGG
Apm17-Orf2-R-PYTP	TGAGTAGGAGTGATGAGACCCAACAACCATGATACCAGGGGGCTCCGTCCCAAAACAGTC
ApM17-Orf4-R-coxA	CCTCGAGGCCTGGGGGCTGGATCAGGCATTCCCCCCTTTCCTTATAAATCCACTATC

Table S2. Plasmids used in the study

Plasmid Name	**Plasmid Backbone**	**Description of Plasmid**
pNLU01	pYTU	*Aspergillus nidulans* expression vector containing genes Clz12 under the AmyB promoter and Clz7 under the glaA promoter.
pNLU02	pYTU	*Aspergillus nidulans* expression vector containing gene Clz12 under the glaA promoter
pNLU03	pYTU	*Aspergillus nidulans* expression vector containing gene Clz17 under the glaA promoter
pNLR01	pYTR	*Aspergillus nidulans* expression vector containing genes Clz10 under the pTrpC promoter and Clz14 under the gpdA promoter
pNLR02	pYTR	*Aspergillus nidulans* expression vector containing genes Clz14 under the gpdA promoter
pNLR03	pYTR	*Aspergillus nidulans* expression vector containing genes Clz10 under the gpdA promoter

pNLP01	pYTP	*Aspergillus nidulans* expression vector containing genes Clz11 under the amyB promoter
pNLP02	pYTP	*Aspergillus nidulans* expression vector containing genes Clz13 under the amyB promoter glaA promoter.
pNLU04	pYTU	Aspergillus nidulans expression vector containing rstn3+rstn4
pNLU05	pYTU	Aspergillus nidulans expression vector containing rstn3
pNLU06	pYTU	Aspergillus nidulans expression vector containing apm4+apm8+apm10+apm13
pNLR04	pYTR	Aspergillus nidulans expression vector containing rstn1+rstn7+rstn8
pNLR05	pYTR	Aspergillus nidulans expression vector containing rstn1+rstn7
pNLR06	pYTR	Aspergillus nidulans expression vector containing rstn7
pNLR07	pYTR	Aspergillus nidulans expression vector containing apm3+apm6+apm9+apm12
pNLP03	pYTP	Aspergillus nidulans expression vector containing rstn2+rstn5+rstn6
pNLP04	pYTP	Aspergillus nidulans expression vector containing rstn5+rstn6
pNLP05	pYTP	Aspergillus nidulans expression vector containing rstn5
pNLP06	pYTP	Aspergillus nidulans expression vector containing rstn6
pNLP07	pYTP	Aspergillus nidulans expression vector containing apm1+apm5+apm7
pYNL051	pxp318	S. cerevisae expression vector with tefp
pYNL052	pxp318	S. cerevisae expression vector with rstn2 under tefp
pYNL053	pxp318	S. cerevisae expression vector with yeast cyp51 under tefp
pYNL054	xw55	S. cerevisae expression vector with rstn3 under the adh2 promoter
pYNL055	xw06	S. cerevisae expression vector with rstn7 under the adh2 promoter
pENL01	pet28a	*E. coli* expression vector with rstn1 under T7 promoter
pENL02	pet28a	*E. coli* expression vector with rstn8 under T7 promoter
pNLU04	pYTU	Aspergillus nidulans expression vector containing rstn3+rstn4
pNLU05	pYTU	Aspergillus nidulans expression vector containing rstn3
pNLU06	pYTU	Aspergillus nidulans expression vector containing apm4+apm8+apm10+apm13
pNLR04	pYTR	Aspergillus nidulans expression vector containing rstn1+rstn7+rstn8
pNLR05	pYTR	Aspergillus nidulans expression vector containing rstn1+rstn7
pNLR06	pYTR	Aspergillus nidulans expression vector containing rstn7
pNLR07	pYTR	Aspergillus nidulans expression vector containing apm3+apm6+apm9+apm12

Table S3. Expression strains used in this study

Strain Name	**Organism**	**Description of Expression Strain**
A. nidulans-ClzA	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLUA and pNLRA
A. nidulans-ClzB	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU01, pNLR01, and pNLP01

A. nidulans-ClzC	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU01, pNLR01, and pNLP02
A. nidulans-ClzD	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU03, pNLR01, and pNLP01
A. nidulans-ClzE	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU01, pNLR02, and pNLP01
A. nidulans-ClzF	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU01, pNLR02, and pNLP01
A. nidulans-ClzG	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU02, pNLR01, and pNLP01
AN2001	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR04, and pNLP03
AN2002	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR06, and pNLP03
AN2003	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR05, and pNLP03
AN2004	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU05
AN2005	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04
AN2006	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU05+pNLR06
AN2007	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR06
AN2008	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR05, and pNLP05
AN2009	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR05, and pNLP06
AN2010	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR05, and pNLP04
AN2011	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU06, pNLR07, and pNLP07
SC001	*S. cerevisaie*	BY4743 expressing PYNL051
SC002	*S. cerevisaie*	BY4743 expressing PYNL052
SC003	*S. cerevisaie*	BY4743 expressing PYNL053
SC004	*S. cerevisaie*	BJ5464 expressing PYNL054
SC005	*S. cerevisaie*	BJ5464 expressing PYNL055
EC001	*E. coli*	BL21 expressing pENL01
EC002	*E. coli*	BL21 expressing pENL02
AN2001	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR04, and pNLP03
AN2002	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR06, and pNLP03
AN2003	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04, pNLR05, and pNLP03
AN2004	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU05
AN2005	*A. nid.*	*A. nid.* A1145ΔSTΔEM expressing pNLU04

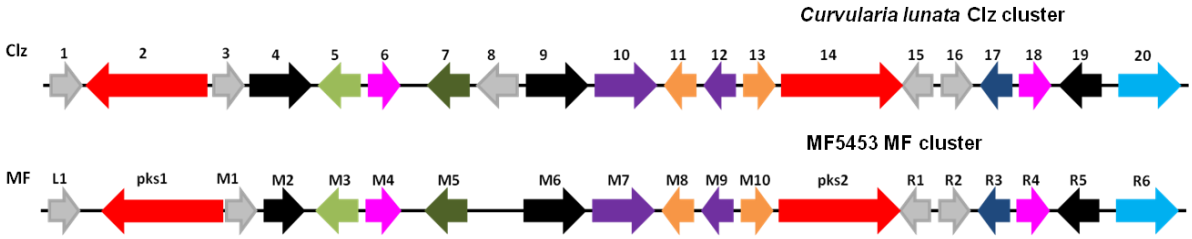


Figure S1. Comparison of the zaragozic acid A biosynthetic gene clusters

Table S4. Comparison of genes between ZA clusters of *C. lunata* and MF5453

C. lunata	MF5453	Putative Function	% Identity
Clz1	mfL1	Hypothetical Protein (HP)	
Clz2	Mfpks1	Polyketide Synthase (Squalestatin S1tetraketide synthase)	88
Clz3	mfM1	Hypothetical Protein (HP)	90
Clz4	mfM2	Transport Protein (MFS)	86
Clz5	mfM3	Short Chain Dehydrogenase (SDR)	86
Clz6	mfM4	Acyltransferase (AT)	88
Clz7	mfM5	Zinc Finger Transcription Factor (TF)	85
Clz8		Hypothetical Protein (HP)	
Clz9	mfM6	Transport Protein (MFS)	94
Clz10	mfM7	Phenylalanine Ammonia Lyase (PAL)	90
Clz11	mfM8	Alpha beta hydrolase	93
Clz12	mfM9	4-coumarate-CoA ligase	92
Clz13	mfM10	Beta lactamase	84
Clz14	Mfpks2	Polyketide Synthase (HRPKS)	91
Clz15	mfR1	Hypothetical Protein (HP): αKG dependent oxygenase (Phyre2)[205]	68
Clz16	mfR2	Hypothetical Protein (HP): αKG dependent oxygenase (Phyre2)[205]	93
Clz17	mfR3	Citrate synthase (CS)	88
Clz18	mfR4	Acyltransferase (AT)	76
Clz19	mfR5	Transport Protein (MFS)	74
Clz20	mfM6	Squalene Synthase (SS)	87

*Highlighted genes studied in this paper

Figure S2. Alignment of N-terminal sequences of various polyketide synthases.
Clz14: benzoyl-priming hexaketide from *C. lunata clz* cluster. Mfpks2: putative benzoyl-priming hexaketide from SQS1 gene cluster in the MF5453. EncA: PKS from enterocin pathway. LovB: nonaketide synthase in lovastatin pathway. CcsA: PKS-NRPS from cytochalasin biosynthesis. AurA: HRPKS in aurovertin E pathway that uses propionate starter unit. The conserved catalytic cysteine in KS domains is highlighted in turquoise. Clz14 and mfpks2 both have a unique ~90 residue N-terminal region upstream of where most KS domains begin. Alignment performed by the Clustal Omega program from Uniprot (online method).[206]

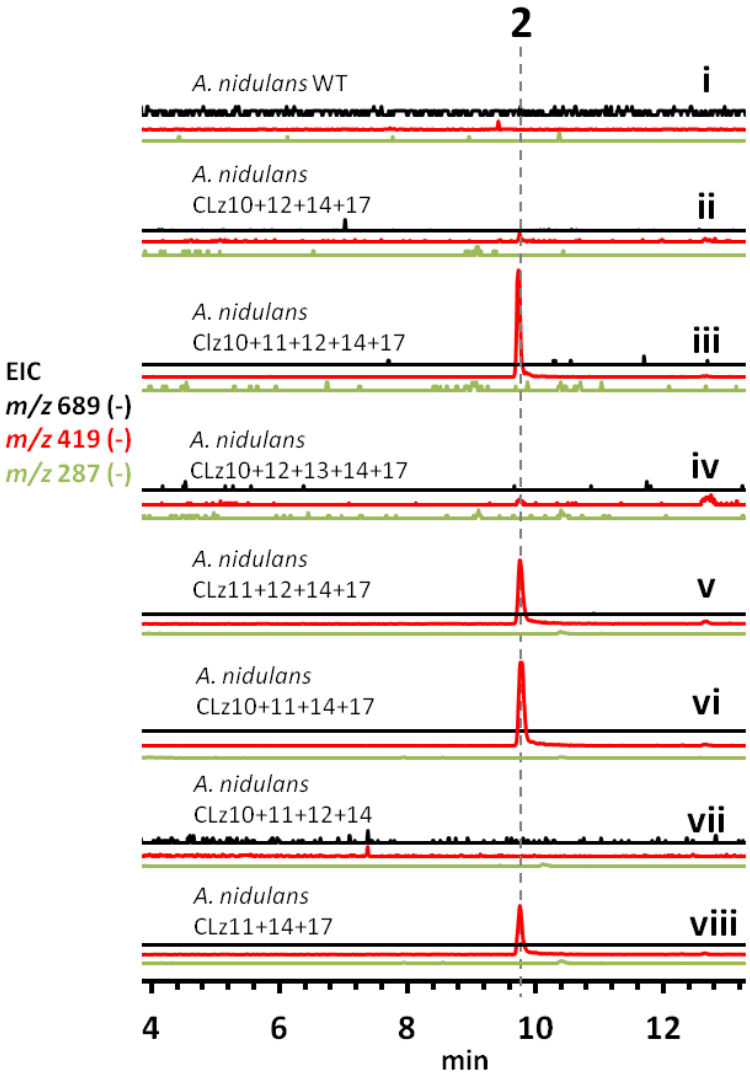


Figure S3. *A. nidulans* cassettes tested for production of 2.
[i] *A. nidulans* A1145 with empty vectors as a negative control. [ii-viii] Cassettes tested for production of zaragozic acid A products. EIC: extracted ion chromatography *m/z* values filtered for the masses of 1, 2, and 3

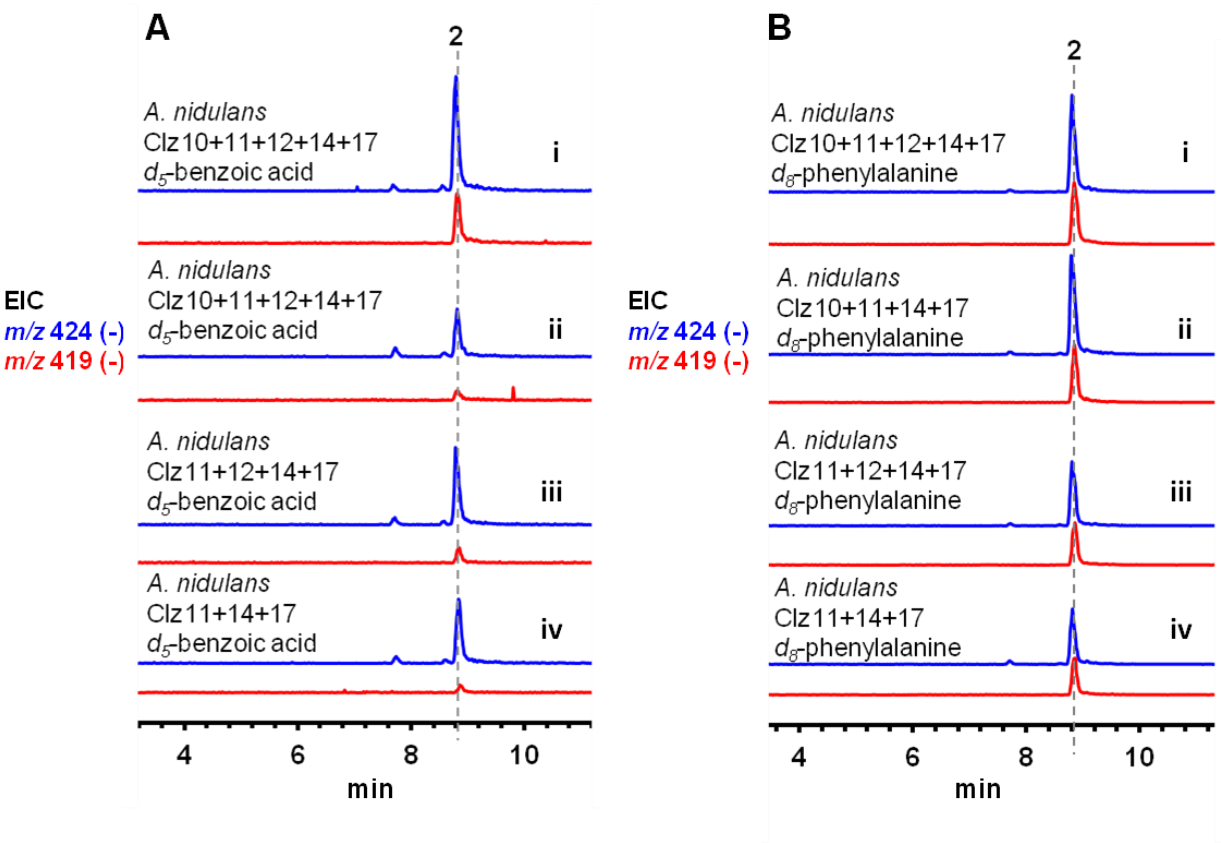
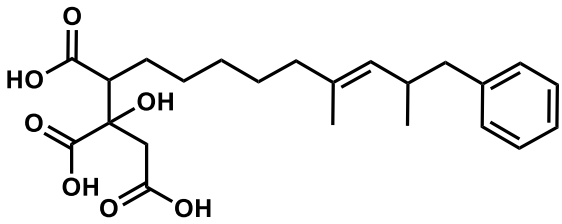
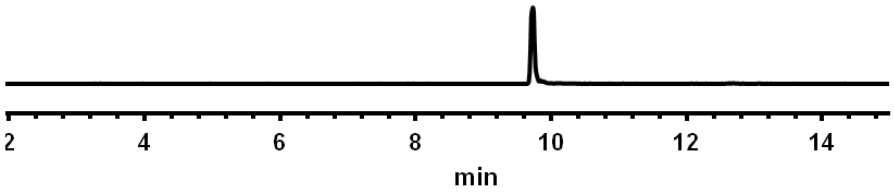
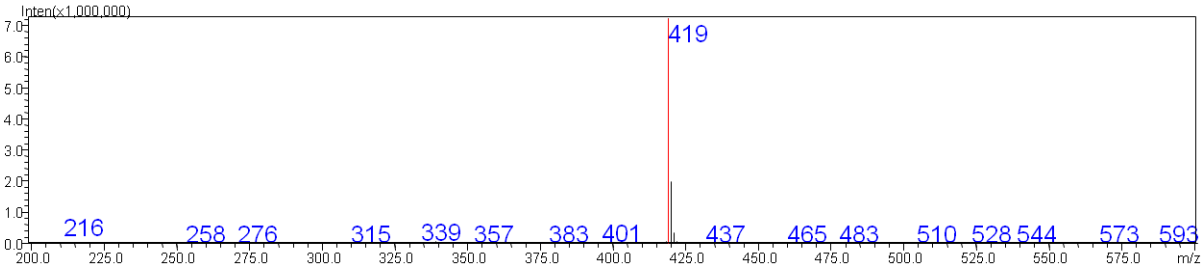


Figure S4. Labeled precursor feeding studies.
Deuterium labeled precursors were fed at 1 mg/mL to *A. nidulans strains* that produced **2**. A) **2** producing strains fed with d_5-benzoic acid. B) **2** producing strains fed with d_8-phenylalanine. The results show that the perdeuterated benzyl ring of both precursors is incorporated into **2**, increasing the mass of the product by 5.

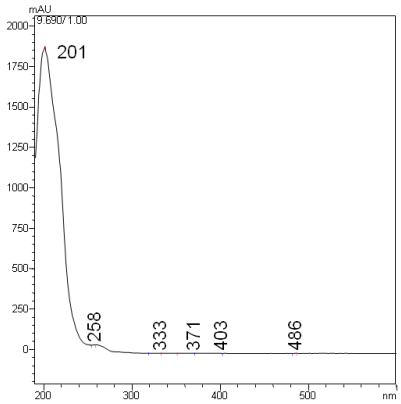


Exact Mass: 420.21





,

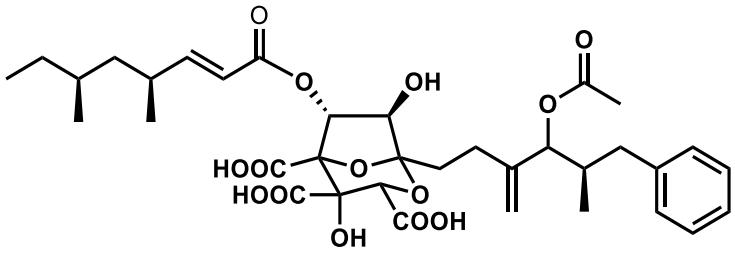


HR-MS for compound **2**: [M-H]- = 420.21805 calculated for $C_{23}H_{32}O_7$; found at 420.20743

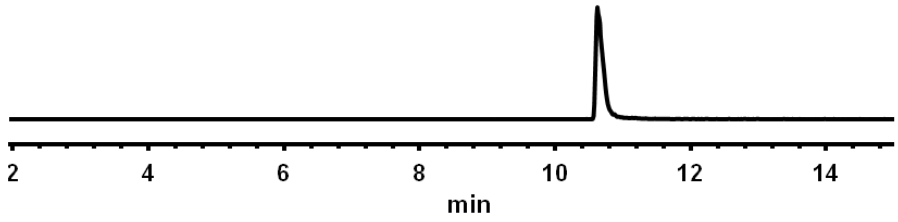
$[\alpha]^{27.8}{}_D$: -44 (c 0.3, CH_3OH). Previously reported $[\alpha]^{20}{}_D$: -48 (c 0.55, CH_3OH).[167]

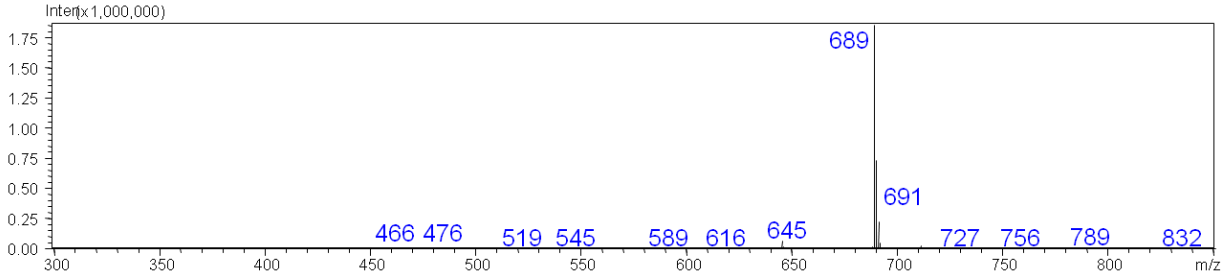
Figure S5. Characterization of 2

MS spectra, UV spectra, HR-MS, and specific optical rotation**.**



Exact Mass: 690.29





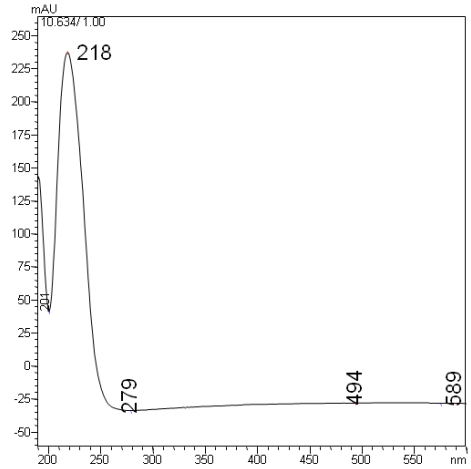


Figure S6. Characterization of 1
MS and UV spectra

Table S5. Metabolites targeted for deletion in *A. nidulans*

Metabolite	Mass	Gene targeted
sterigmatocystin	324	*stcA*
emericellamide A	609	*easA*
emericellamide C	595	*easA*
emericellamide D	595	*easA*
emericellamide E	623	*easA*
emericellamide F	623	*easA*

*Mass and genes adapted from Yaegashi et al.[207]

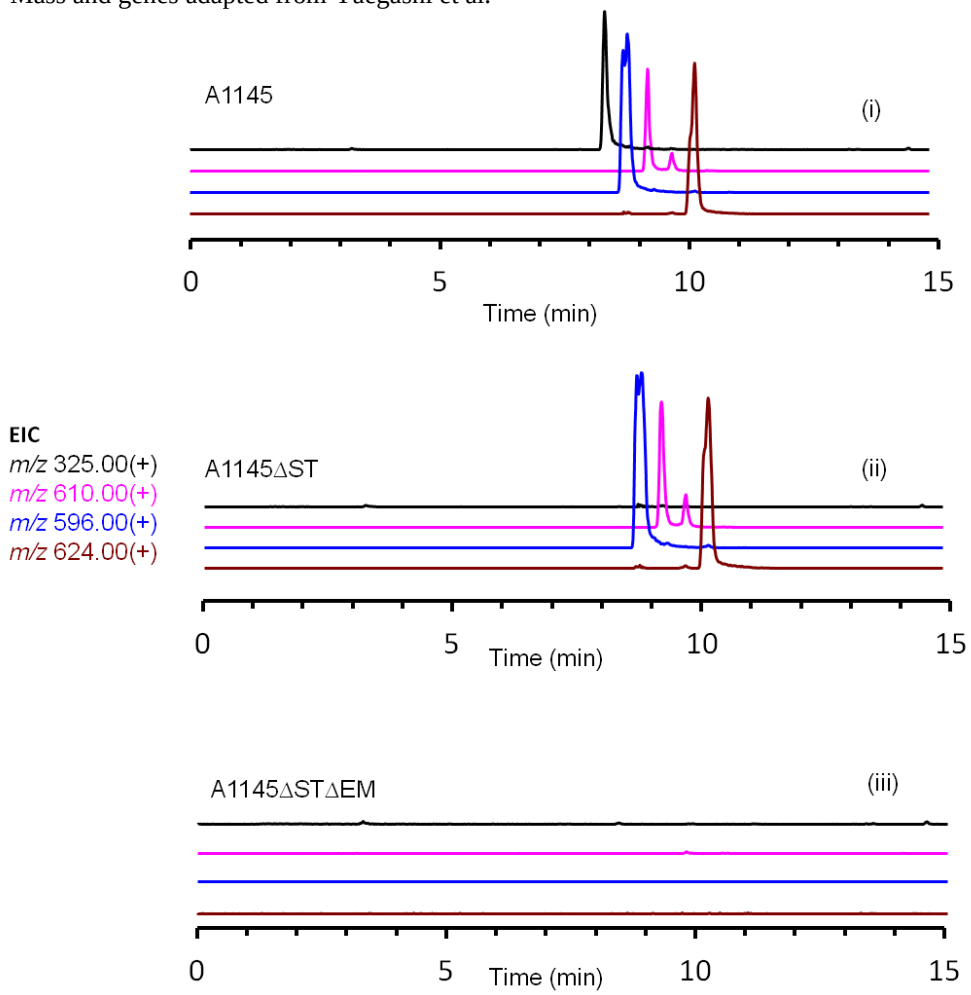


Figure S7. Removal of undesired metabolites in *A. nidulans*.
The production of certain metabolites were targeted and knocked out using the CRISPR/Cas9 system outlined in the methods. Traces show the successful deletion of these products. (i) *A. nidulans* A1145ΔST (ii) removal of sterigmatocystin (iii) removal of emericellamide products.

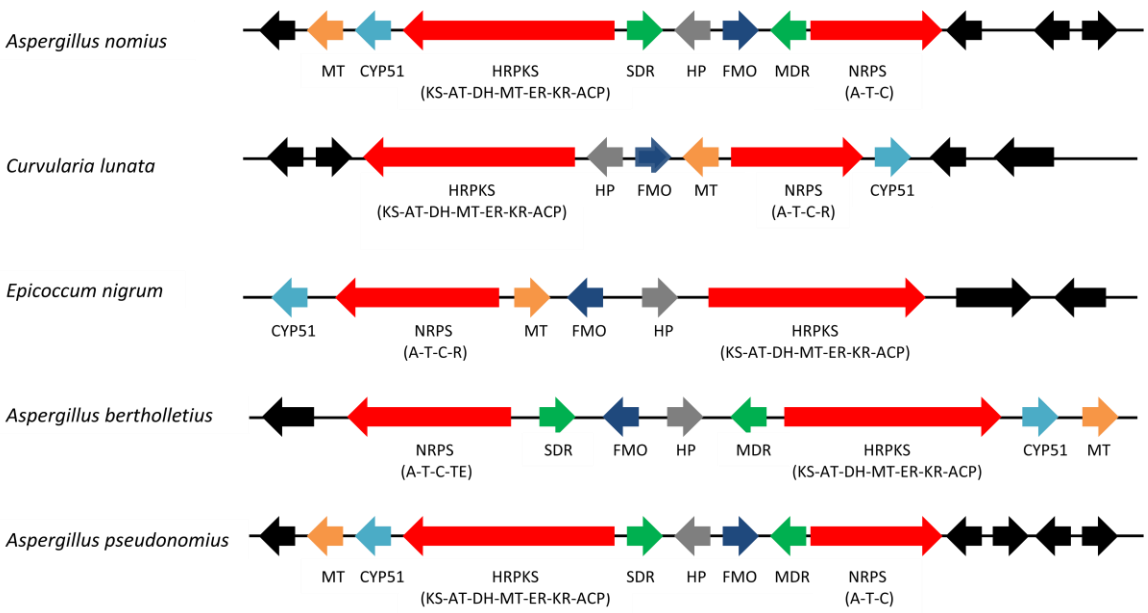


Figure S8. Homologous clusters of *rstn*
Homologous gene clusters from different strains of fungi. Homologous genes shown in same color

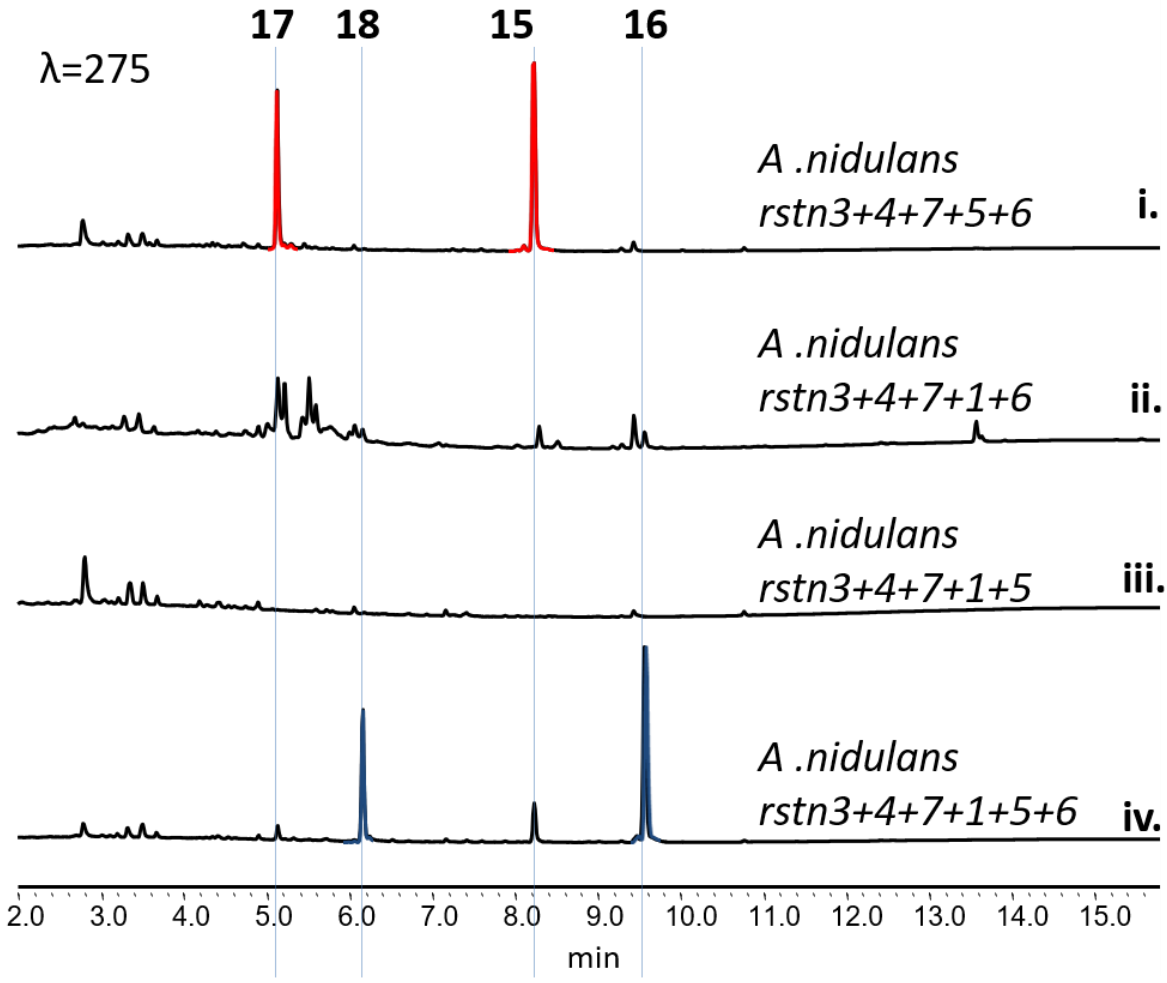


Figure S9. Expression of late *rstn* pathway intermediates
Combinations of genes expressing late pathway intermediates

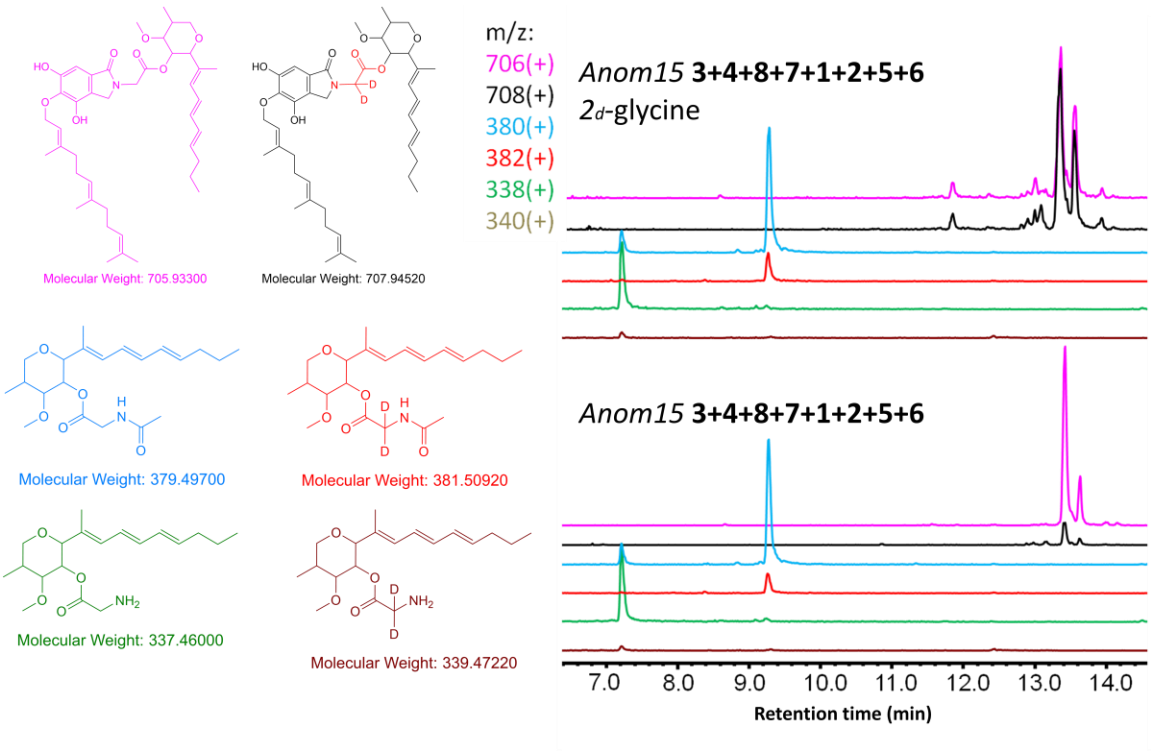


Figure S10. Labeled glycine feeding
Incorportaion of d2-glycine fed into constructs expressing the *rstn* cluster

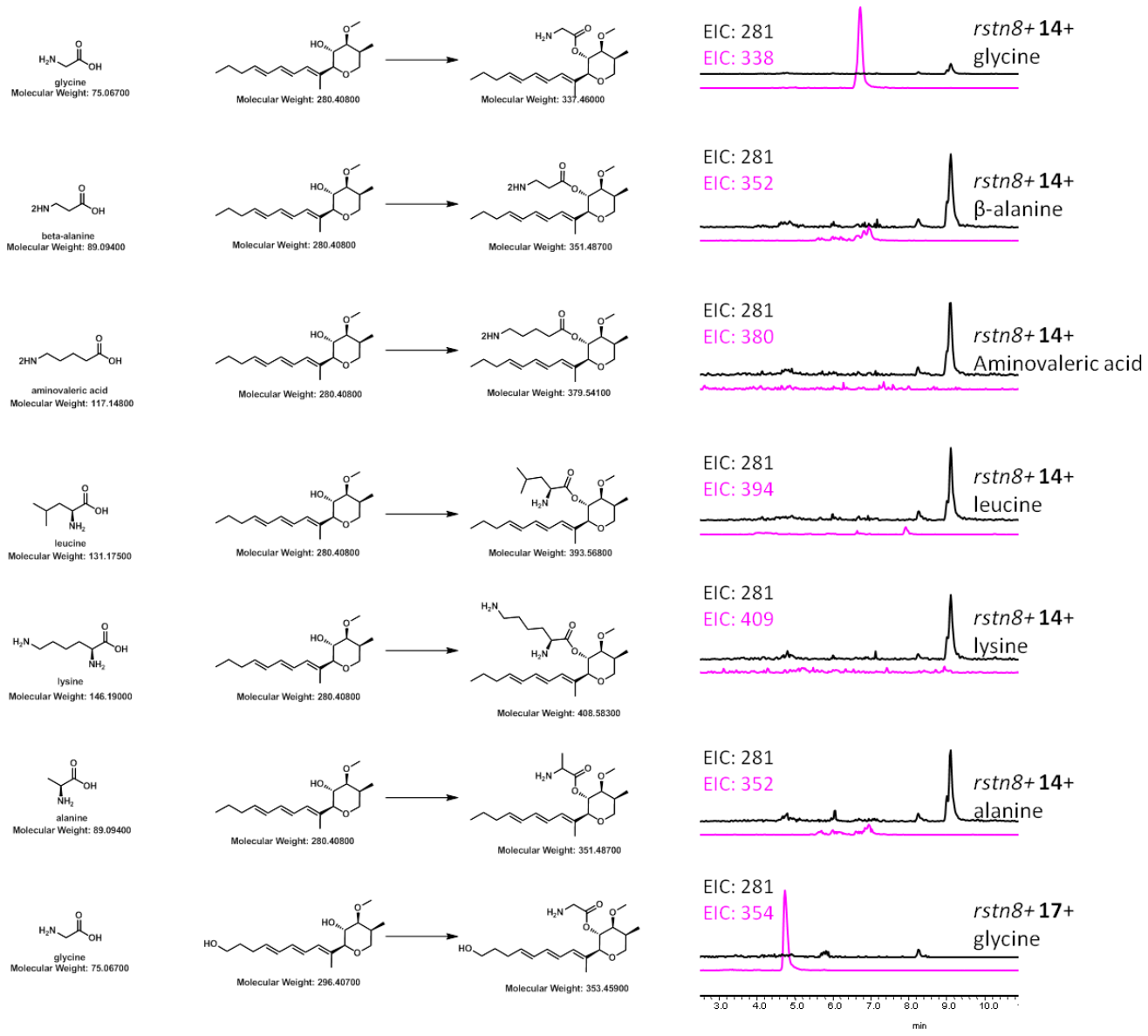


Figure S11. Substrate scope testing of *rstn8*
Different amino acid substrates and polyketide substrates tested with *rstn8*

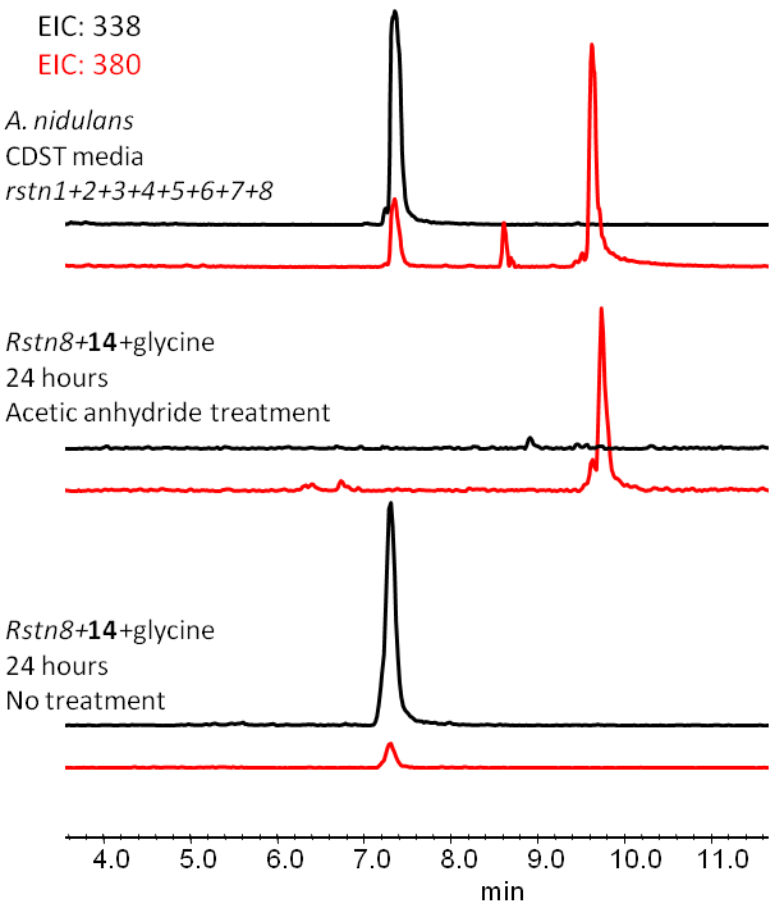
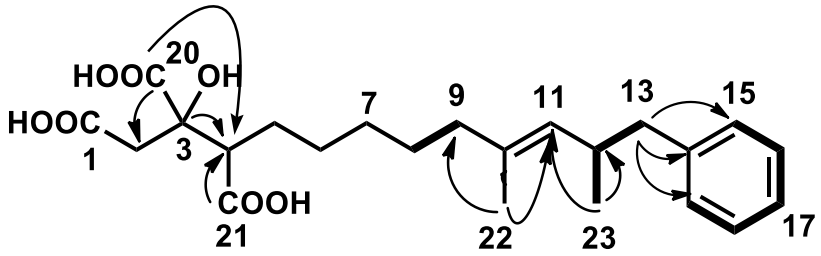


Figure S12. Derivatization of restricticin to confirm structure
Restricticin produced from *in vitro rstn8* reactions treated with acetic anhydride to yield compound **18**

Table S6. ^{1}H (500 Hz) and ^{13}C (125 Hz) of 2 in CD_3OD*



Key COSY correlations Key HMBC correlations

Position	δ_H (J in Hz)	δ_C
1	-	181.61/181.45
2	2.66 (overlapped)/2.97 (1H, d, 16.0)	43.71/42.51
	2.58 (overlapped)/2.88 (1H, d, 16.0)	
3	-	77.04
4	2.66/2.58 (1H, overlapped)	54.50/54.36
5/6/7/8	1.05-1.45 (8H, overlapped)	28.67/28.83/29.65/29.97
9	1.87, 2H	40.59/40.41
10	-	135.18/135.15
11	4.91 (overlapped)	131.56/131.49
12	2.62 (overlapped)	35.91
13	2.45 (dd, 8.11 and 13.03)	45.13/ 45.11
	2.55 (dd, 2.61 and 6.14)	
14	-	142.39
15	7.10 (t, 6.34)	130.31/130.28
16	7.21 (dd, 2.27 and 7.61)	129.01/128.98
17	7.10 (t, 6.34)	126.67/127.66
18	7.21 (dd, 2.27 and 7.61)	129.01/128.98
19	7.10 (t, 6.34)	130.31/130.28
20	-	178.19/177.57
21	-	174.89/174.79
22	1.34 (d, 4.38), 3H	16.11/16.00
23	0.93 (d, 6.42), 3H	21.48/21.44

* The presence of an additional set of closely matched signals in ^{13}C-NMR spectrum is attributed to the existence of a stereoisomer of **2**.

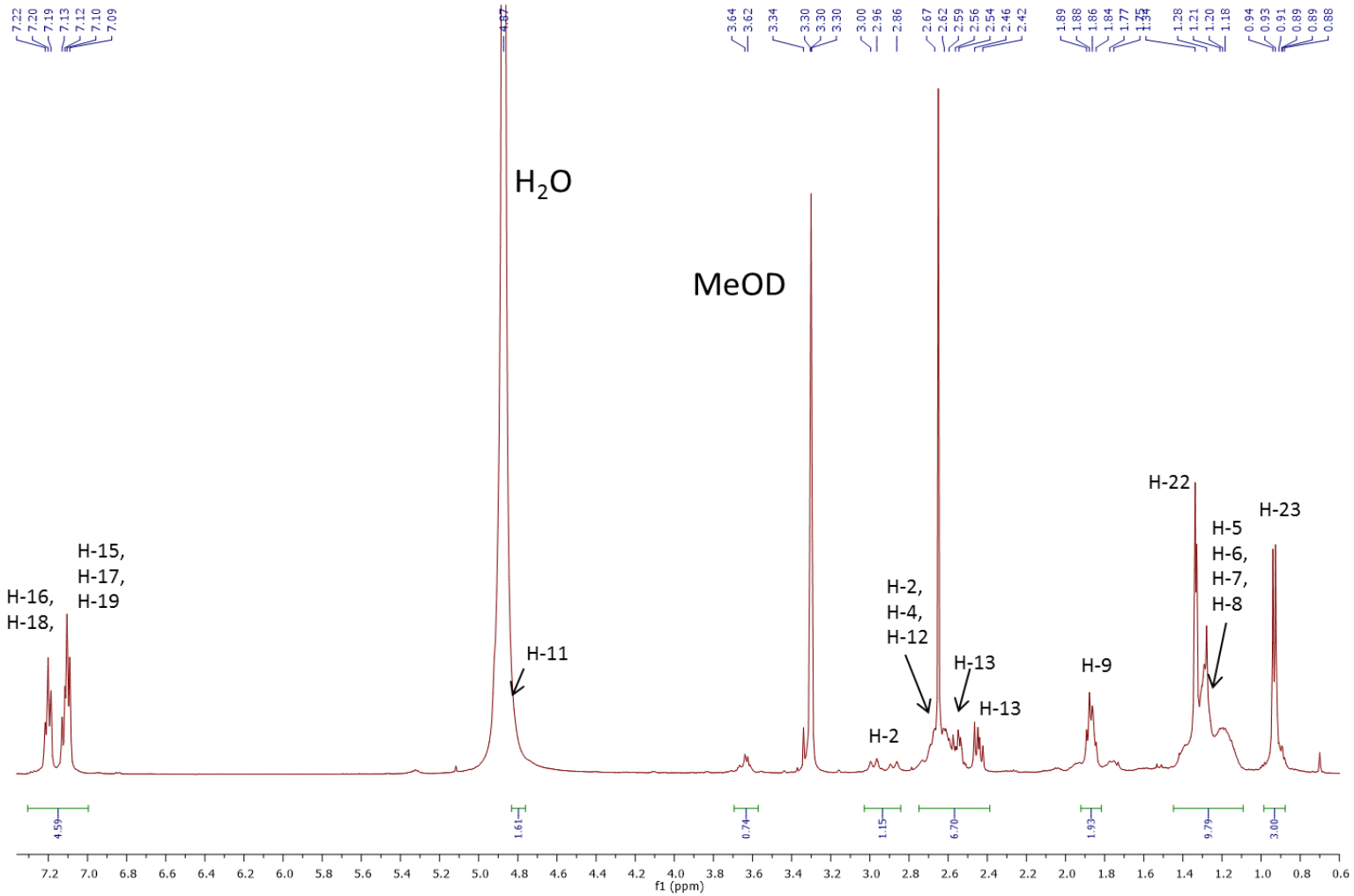


Figure S13. ^{1}H NMR spectrum of 2 in CD_3OD (500 MHz).

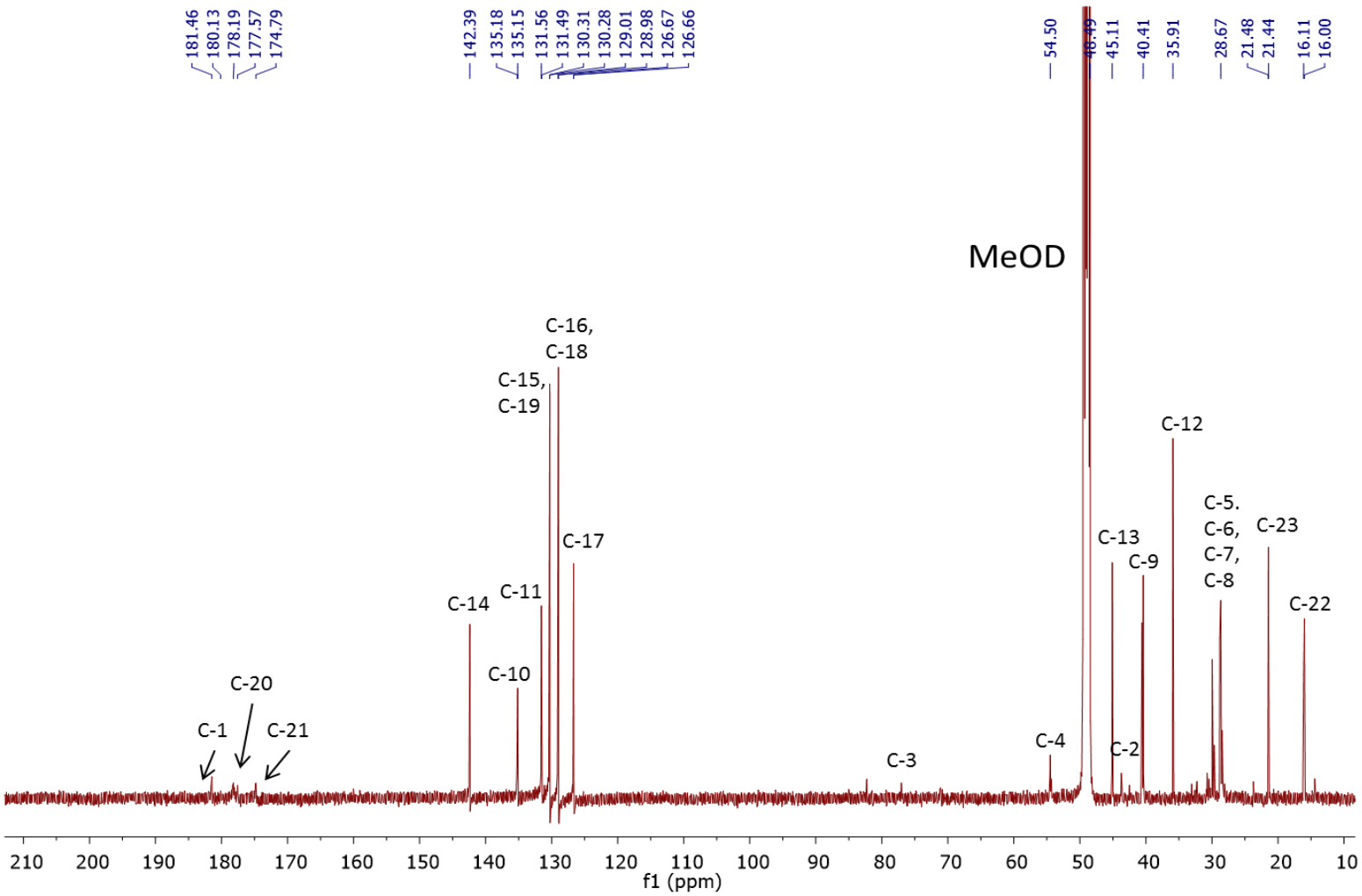


Figure S14. ^{13}C NMR spectrum of 2 in CD_3OD (125 MHz).

Figure S15. COSY NMR spectrum of 2 in CD_3OD

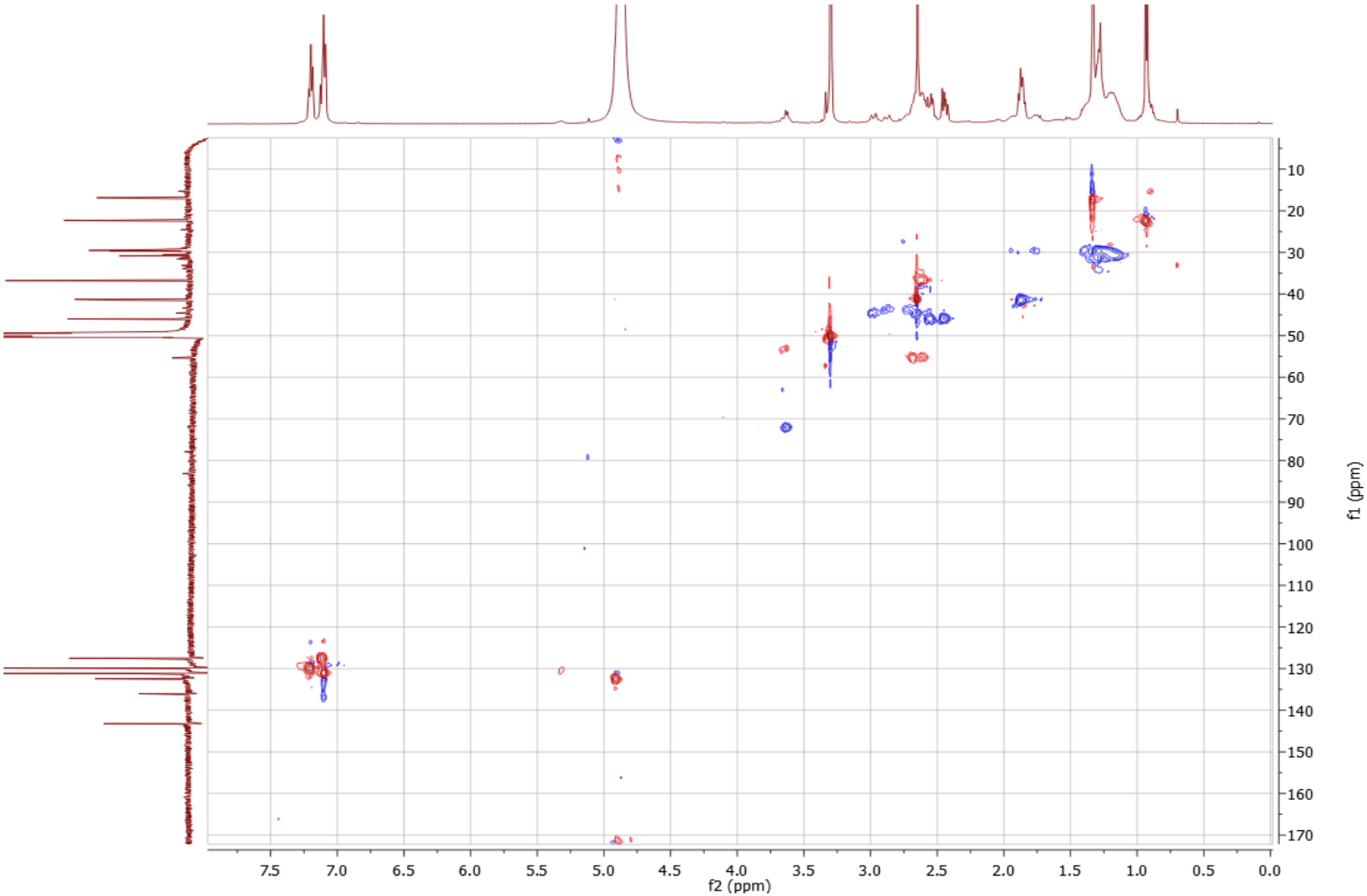


Figure S16. HSQC NMR spectrum of 2 in CD_3OD.

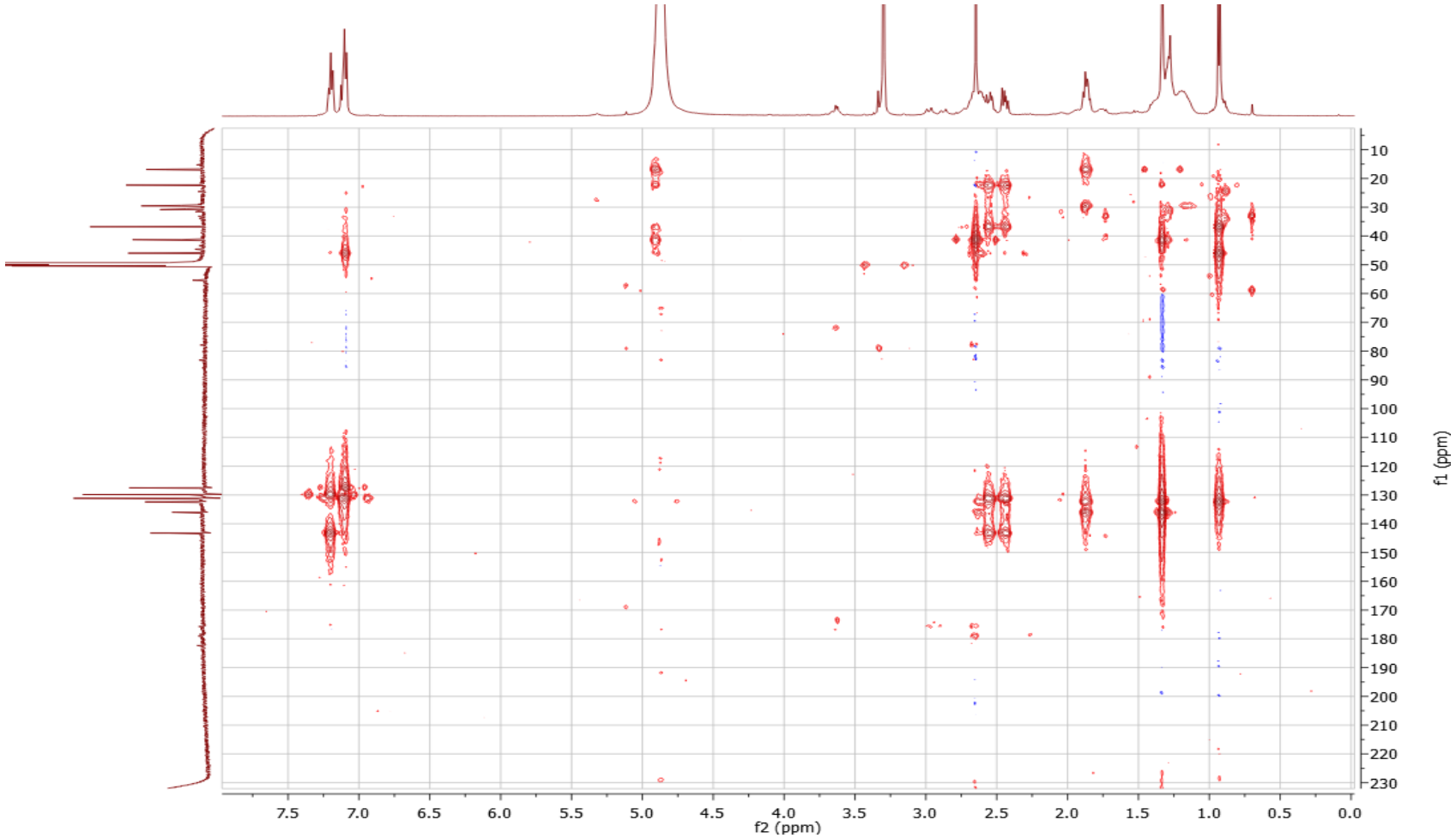


Figure S17. HMBC NMR spectrum of 2 in CD_3OD.

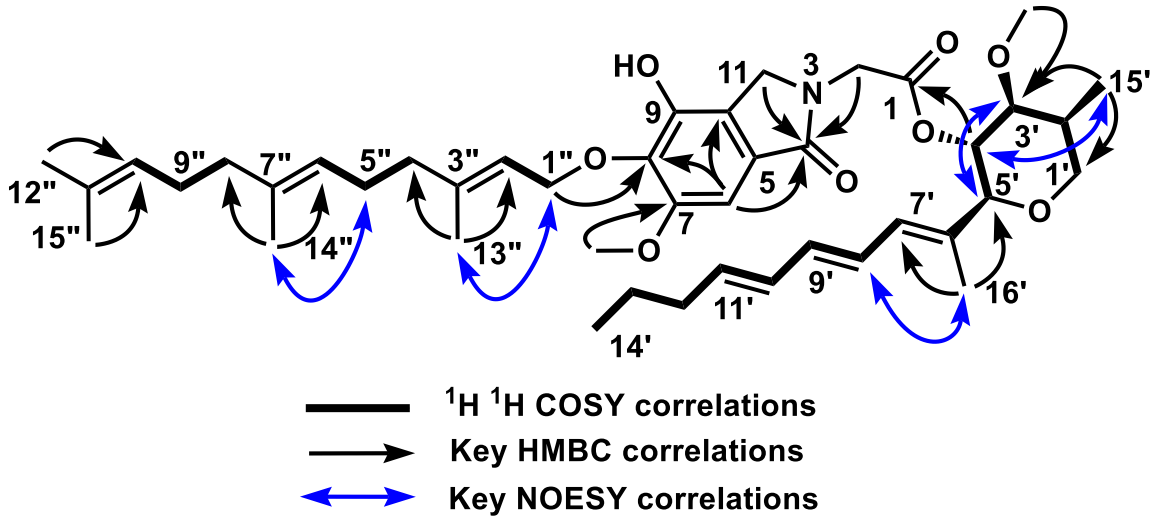


Table S7. ^{1}H (500 Hz) and ^{13}C (125 Hz) of **19-21**

No.	**19**		**20**		**21**	
	δc	*δH* (mult., J_{H-H} in Hz)	*δc*	*δH* (mult., J_{H-H} in Hz)	*δc*	*δH* (mult., J_{H-H} in Hz)
1	168.0		168.2		168.0	
2	44.2	4.48, d (17.4) 4.02, d (17.4)	44.7	4.35, d (17.4) 4.11, d (17.4)	44.1	4.52, d (17.4) 4.03, d (17.4)
4	169.4		169.6		168.7	
5	127.4		127.2		127.5	
6	102.8	6.93, s	98.8	6.96, s	98.8	6.97, s
7	150.7		153.5		153.3	
8	136.9		137.4		137.1	
9	144.7		144.7		144.7	
10	119.5		120.5		120.4	
11	47.9	4.31, d (16.5) 4.17, d (16.5)	48.5	4.38, d (16.5) 4.22, d (16.5)	47.6	4.34, d (16.5) 4.18, d (16.5)
1'	70.8	3.79, dd (11.8,1.3) 3.55, dd (11.8, 2.6)	71.3	3.76, d (11.5, 1.0) 3.58, dd (11.5, 2.0)	70.8	3.79, dd (11.8,1.3) 3.54, dd (11.8, 2.6)
2'	32.4	2.22, m	35.8	2.14, m	32.5	2.23, m
3'	81.4	3.33, dd (9.5, 5.2)	72.4	3.80, dd (9.5, 5.2)	81.4	3.33, dd (9.5, 5.2)
4'	70.0	5.00, t (9.5)	72.3	4.98, t (9.5)	70.0	5.01, t (9.5)
5'	85.2	3.51, d (9.5)	84.8	3.50, d (9.5)	85.2	3.51, d (9.5)
6'	132.7		133.0		132.7	

7'	129.7	5.96 d (10.5)	129.5	5.93 d (10.5)	129.7	5.97 d (10.5)
8'	125.8	6.28, dd (14.7, 10.5)	125.8	6.30, dd (14.7, 10.5)	125.8	6.29, dd (14.7, 10.5)
9'	134.4	6.20, dd (14.7, 10.5)	134.2	6.20, dd (14.7, 10.5)	134.4	6.21, dd (14.7, 10.5)
10'	130.7	6.10, dd (14.7, 10.5)	130.6	6.12, dd (14.7, 10.5)	130.7	6.11, dd (14.7, 10.5)
11'	135.9	5.72, dt (14.7, 7.2)	135.9	5.73, dt (14.7, 7.2)	135.8	5.72, dt (14.7, 7.2)
12'	34.9	2.05, m	34.9	2.09, m	34.9	2.08, m
13'	22.4	1.40, m	22.4	1.42, m	22.4	1.41, m
14'	13.7	0.88, t (7.5)	13.7	0.90, t (7.5)	13.7	0.89, t (7.5)
15'	10.7	1.06, d (7.2)	10.8	1.15, d (7.2)	10.8	1.07, d (7.2)
16'	11.7	1.76, s	11.7	1.78, s	11.7	1.77, s
1"	69.7	4.64, d (7.4)	69.7	4.66, d (7.4)	69.6	4.65, d (7.4)
2"	119.1	5.51, t (7.4)	119.1	5.50, t (7.4)	119.1	5.49, t (7.4)
3"	144.4		144.1		144.0	
4"	39.7	2.04, m 1.95, m	39.7	2.06, m 1.97, m	39.7	2.05, m 1.97, m
5"	26.2	2.07, m	26.3	2.09, m	26.3	2.08, m
6"	123.5	5.07, m	123.5	5.08, m	123.5	5.08, m
7"	135.6		135.6		135.6	
8"	39.6	2.04, m 1.95, m	39.6	2.06, m 1.97, m	39.6	2.05, m 1.97, m
9"	26.7	2.03, m	26.7	2.05, m	26.7	2.05, m
10"	124.2	5.06, m	124.2	5.07, m	124.2	5.07, m
11"	131.4		131.4		131.4	
12"	25.7	1.66, s	25.7	1.67, s	25.7	1.66, s
13"	16.4	1.65, s	16.4	1.66, s	16.4	1.65, s
14"	16.0	1.58, s	16.0	1.58, s	16.0	1.58, s
15"	17.7	1.58, s	17.7	1.58, s	17.7	1.58, s
3'-*OMe*	56.3	3.29, s			56.4	3.30, s
7-*OMe*			56.2	3.91, s	56.2	3.90, s

In $CDCl_3$, 500 MHz for ^1H and 125 MHz for ^{13}C NMR; Chemical shifts are reported in ppm. All signals are determined by 1H-1H COSY, HMBC and HSQC correlation.

Table S8. ^{1}H (500 Hz) and ^{13}C (125 Hz) of **14** and **18**

No.	**18**		**14**	
	δc	*δH* (mult., J_{H-H} in Hz)	*δc*	*δH* (mult., J_{H-H} in Hz)
1	70.8	3.81, dd (11.8,1.3) 3.57, dd (11.8, 2.6)	71.0	3.81, dd (11.8,1.3) 3.60, dd (11.8, 2.6)
2	32.4	2.26, m	31.7	2.21, m
3	81.4	3.37, dd (9.5, 5.2)	84.1	3.26, dd (9.5, 5.2)
4	70.2	5.02, t (9.5)	67.7	3.62, t (9.5)
5	85.1	3.53, d (9.5)	86.8	3.48, d (9.5)
6	132.6		133.4	
7	129.6	5.94 d (10.5)	129.8	6.14 d (10.5)
8	125.6	6.24, dd (14.7, 10.5)	125.8	6.34, dd (14.7, 10.5)
9	134.2	6.15, dd (14.7, 10.5)	134.3	6.21, dd (14.7, 10.5)
10	130.5	6.06, dd (14.7, 10.5)	130.7	6.10, dd (14.7, 10.5)
11	136.1	5.71, dt (14.7, 7.2)	135.7	5.71, dt (14.7, 7.2)
12	34.9	2.06, m	34.9	2.06, m
13	22.4	1.40, m	22.5	1.41, m
14	13.7	0.89, t (7.5)	13.8	0.89, t (7.5)
15	10.7	1.08, d (7.2)	10.8	1.05, d (7.2)
16	11.6	1.76, s	12.2	1.82, s
1'	169.4			
2'	41.4	4.10, dd (18.5, 5.5) 3.77, dd (18.5, 4.2)		
3'	170.0			
4'	22.9	1.97, s		
3-*OMe*	56.4	3.32, s	56.0	3.40, s

In $CDCl_3$, 500 MHz for ^{1}H and 125 MHz for ^{13}C NMR; Chemical shifts are reported in ppm. All signals are determined by 1H-1H COSY, HMBC and HSQC correlation.

Table S9. ^{1}H (500 Hz) and ^{13}C (125 Hz) of **13** and **17**

No.	**13**		**17**	
	δc	δH (mult., $J_{H\text{-}H}$ in Hz)	δc	δH (mult., $J_{H\text{-}H}$ in Hz)
1	71.3	3.76, dd (11.8,1.3) 3.62, dd (11.8, 2.6)	71.0	3.79, dd (11.8,1.3) 3.58, dd (11.8, 2.6)
2	35.4	2.11, m	31.7	2.19, m
3	74.8	3.71, dd (9.5, 5.2)	84.1	3.24, dd (9.5, 5.2)
4	69.0	3.53, t (9.5)	67.6	3.60, t (9.5)
5	86.8	3.43, d (9.5)	86.7	3.46, d (9.5)
6	133.0		133.8	
7	129.9	6.11 d (10.5)	129.7	6.11 d (10.5)
8	125.5	6.33, dd (14.7, 10.5)	126.3	6.33, dd (14.7, 10.5)
9	134.6	6.22, dd (14.7, 10.5)	133.9	6.18, dd (14.7, 10.5)
10	130.5	6.10, dd (14.7, 10.5)	131.1	6.11, dd (14.7, 10.5)
11	136.1	5.73, dt (14.7, 7.2)	134.5	5.69, dt (14.7, 7.2)
12	34.9	2.07, m	29.1	2.16, m
13	22.4	1.41, m	32.0	1.64, m
14	13.7	0.90, t (7.5)	62.2	3.62, t (6.5)
15	11.0	1.11, d (7.2)	10.8	1.02, d (7.2)
16	12.0	1.81, s	12.3	1.80, s

In $CDCl_3$, 500 MHz for ^{1}H and 125 MHz for ^{13}C NMR; Chemical shifts are reported in ppm. All signals are determined by 1H-1H COSY, HMBC and HSQC correlation.

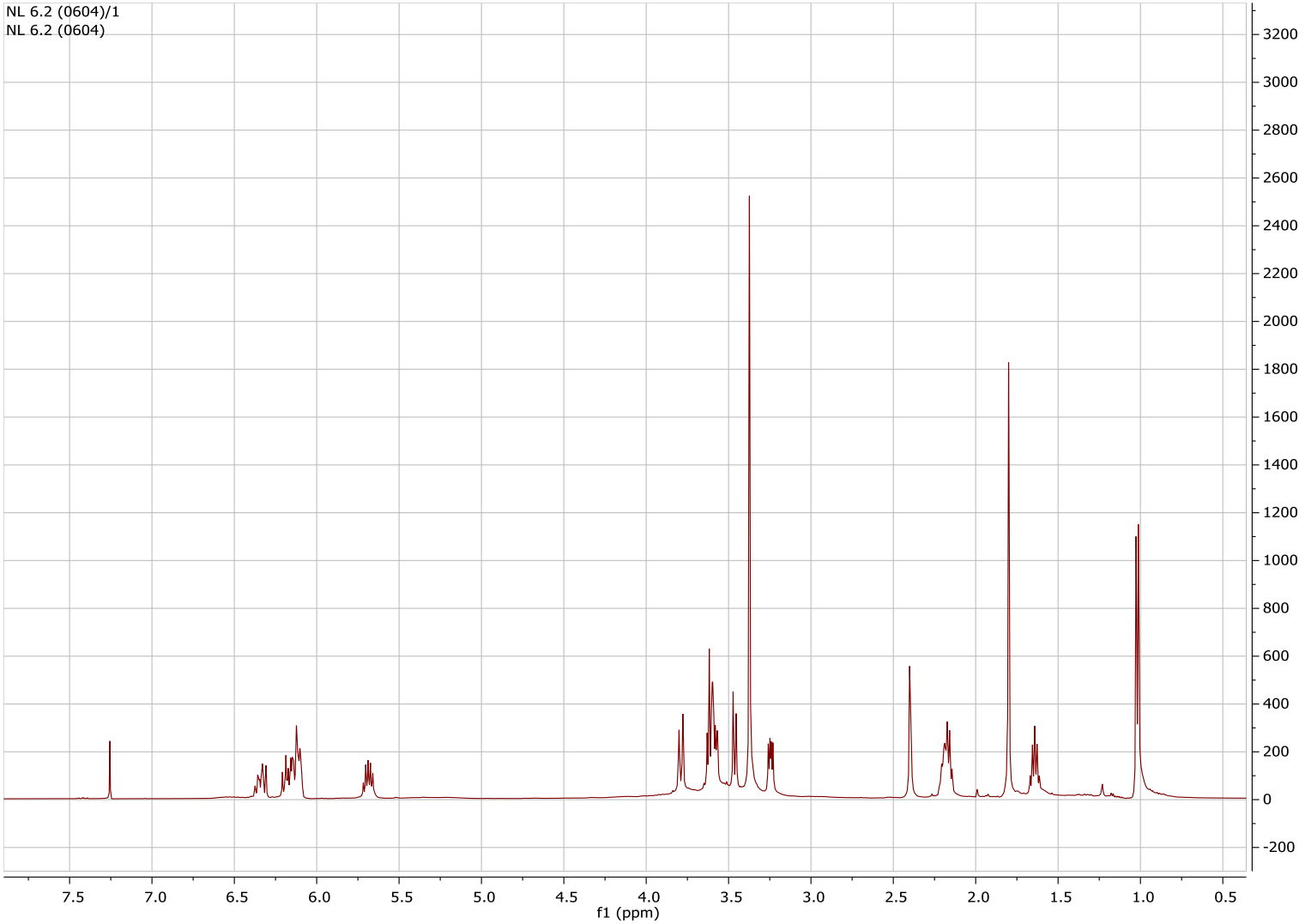


Figure S18. ^{1}H NMR spectrum of 17 in $CDCl_3$ (500 MHz).

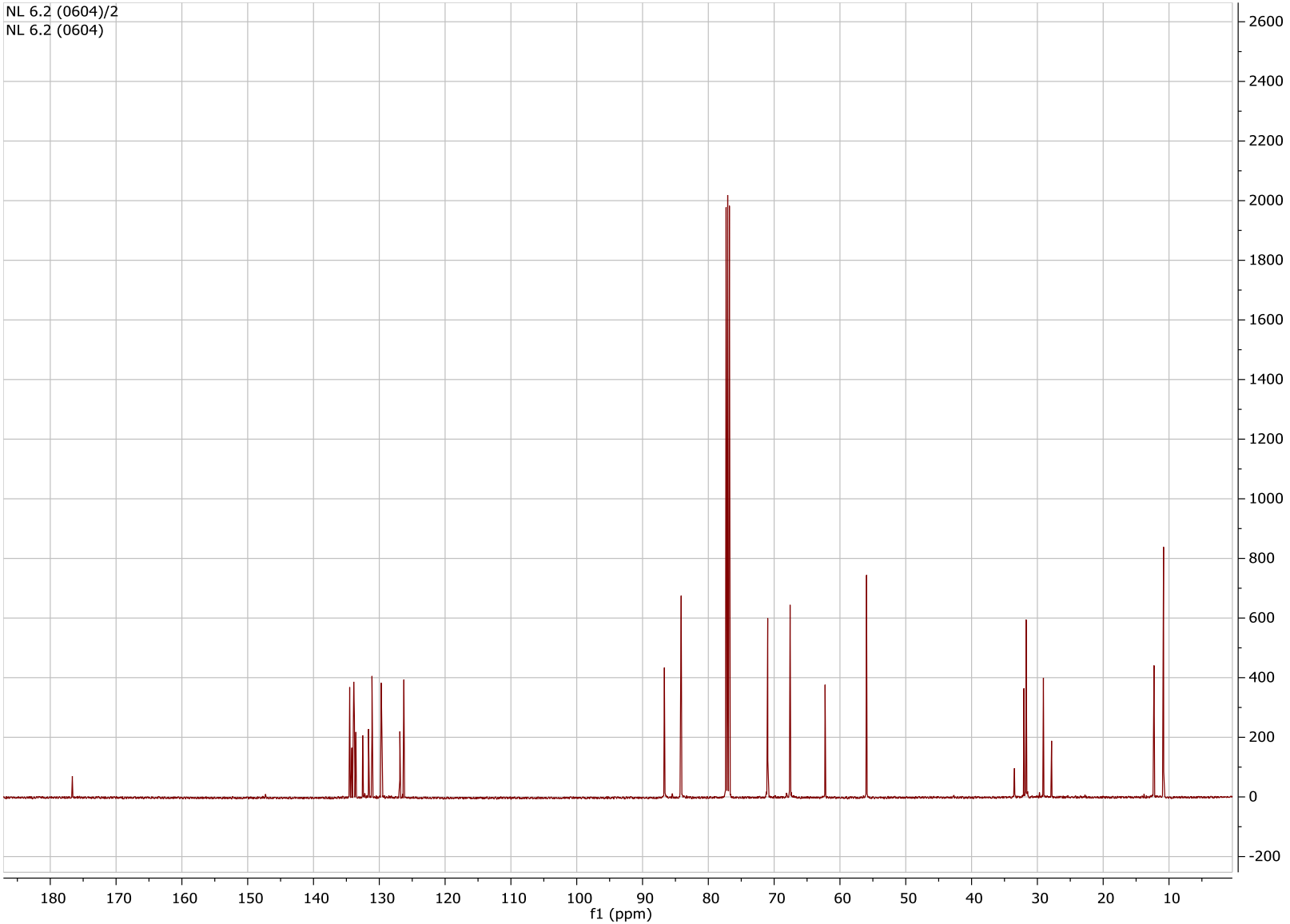


Figure S19. ^{13}C NMR spectrum of 17 in $CDCl_3$ (125 MHz).

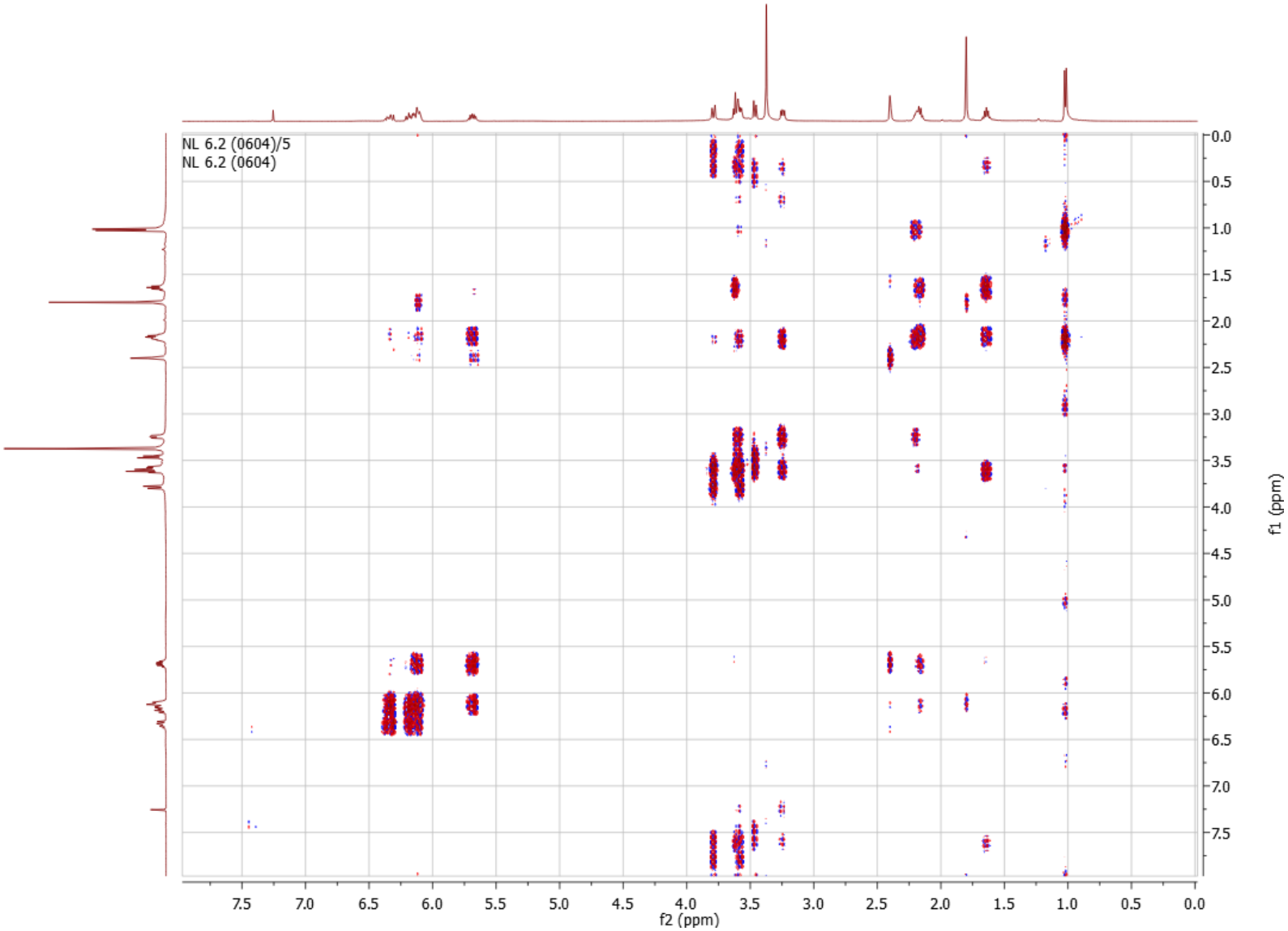


Figure S20. COSY NMR spectrum of 17 in $CDCl_3$

Figure S21. HSQC NMR spectrum of 17 in $CDCl_3$.

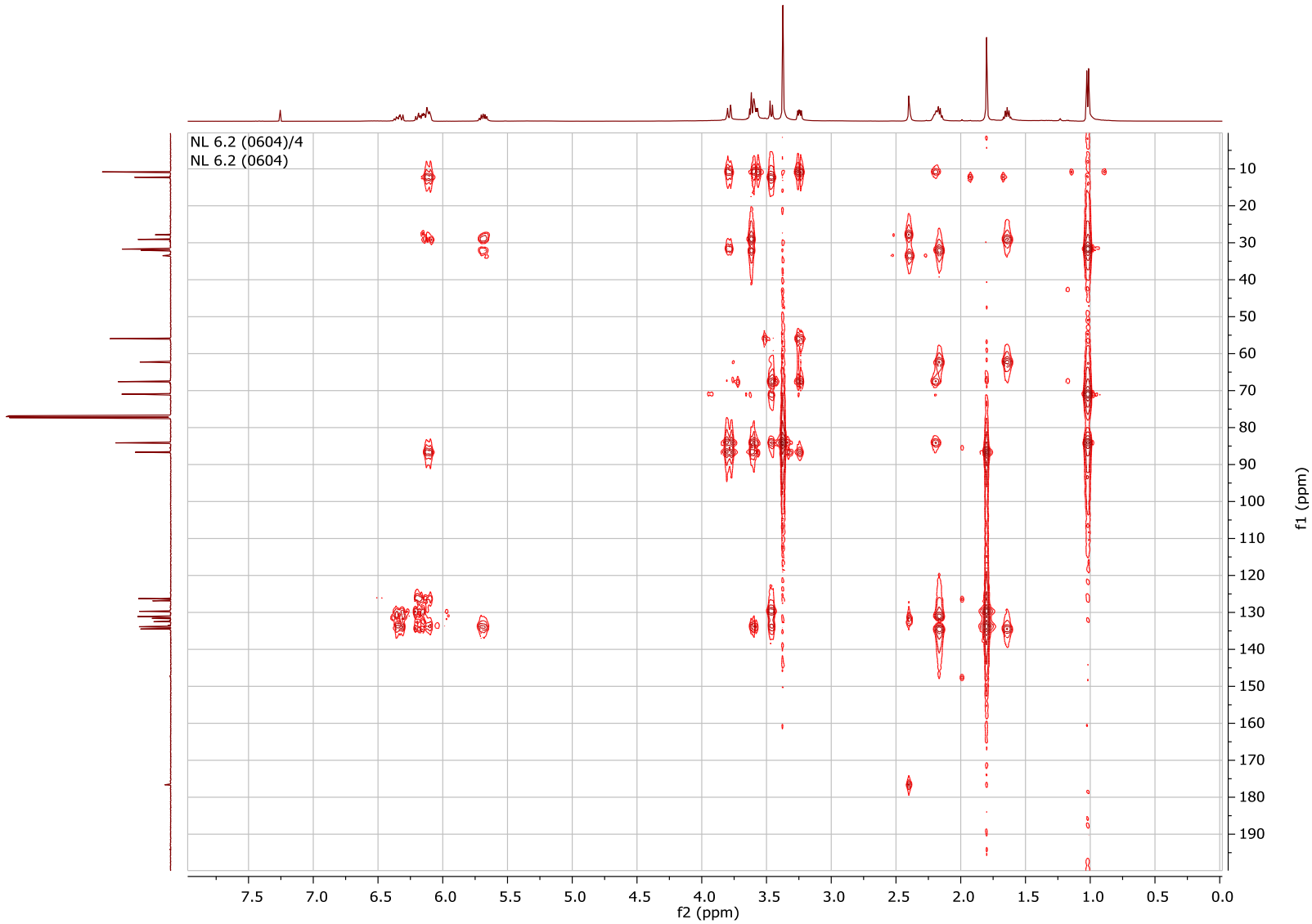


Figure S22. HMBC NMR spectrum of 17 in $CDCl_3$

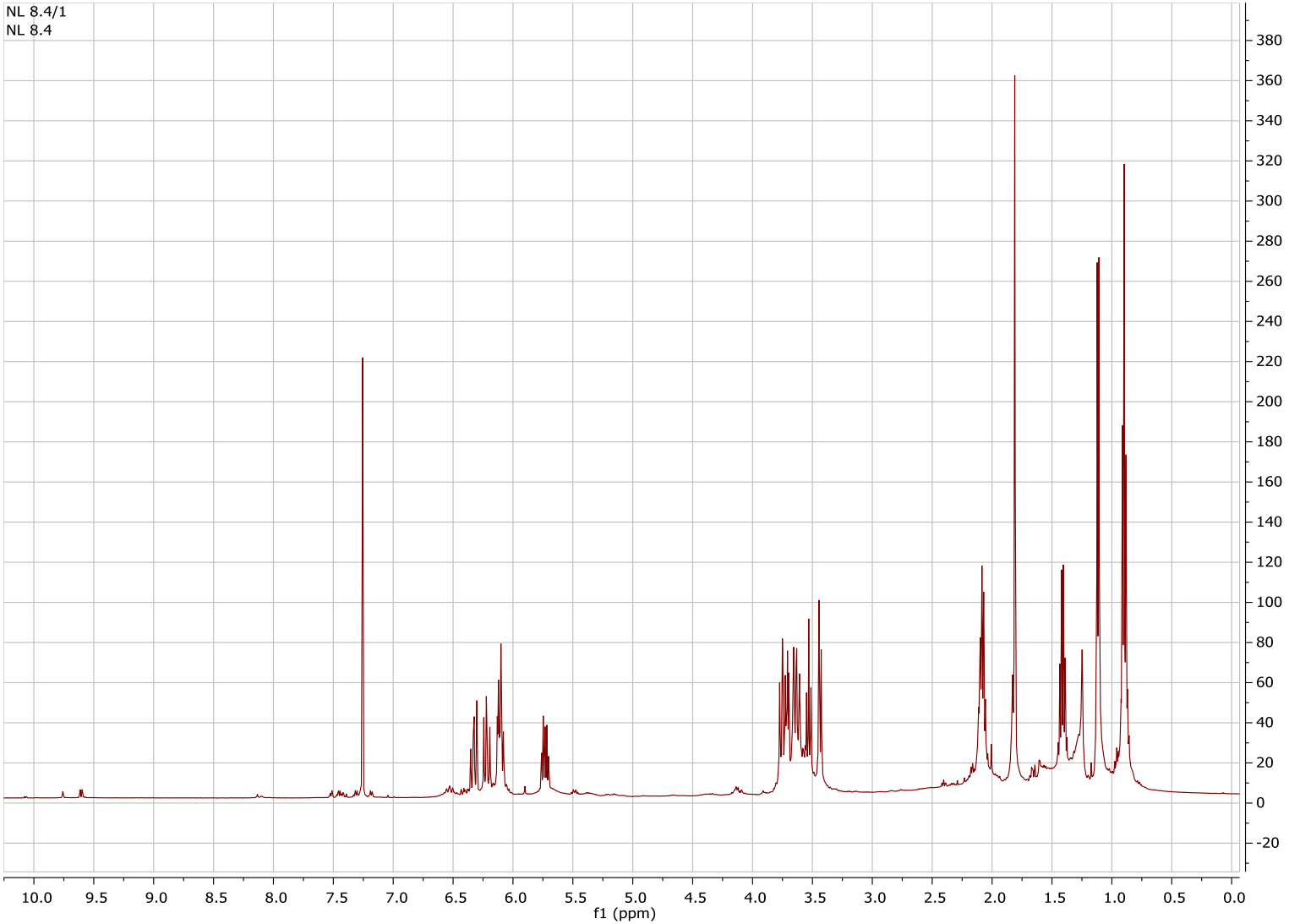


Figure S23. ^{1}H NMR spectrum of 14 in $CDCl_3$ (500 MHz).

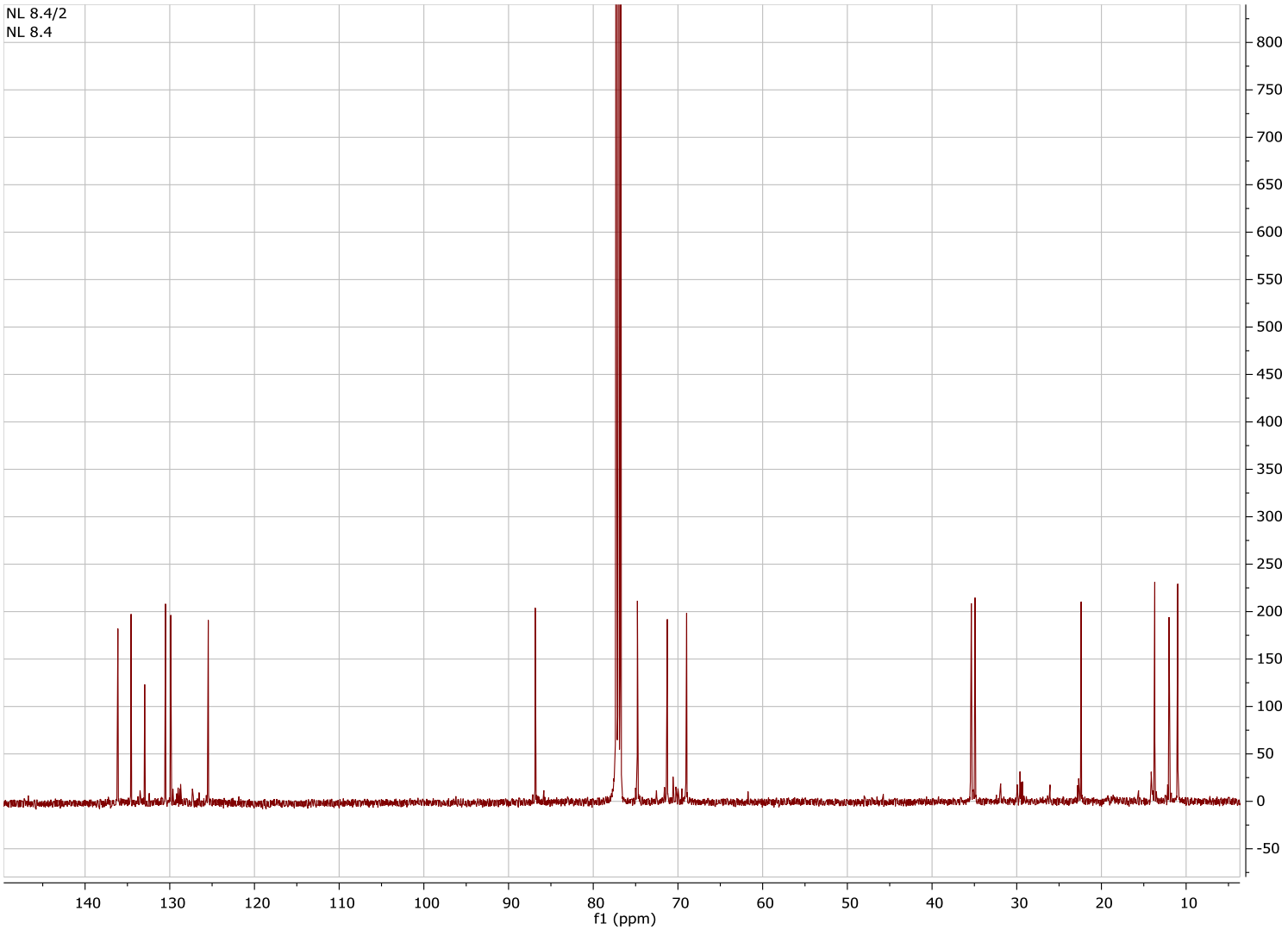


Figure S24. ^{13}C NMR spectrum of 14 in $CDCl_3$ (125 MHz).

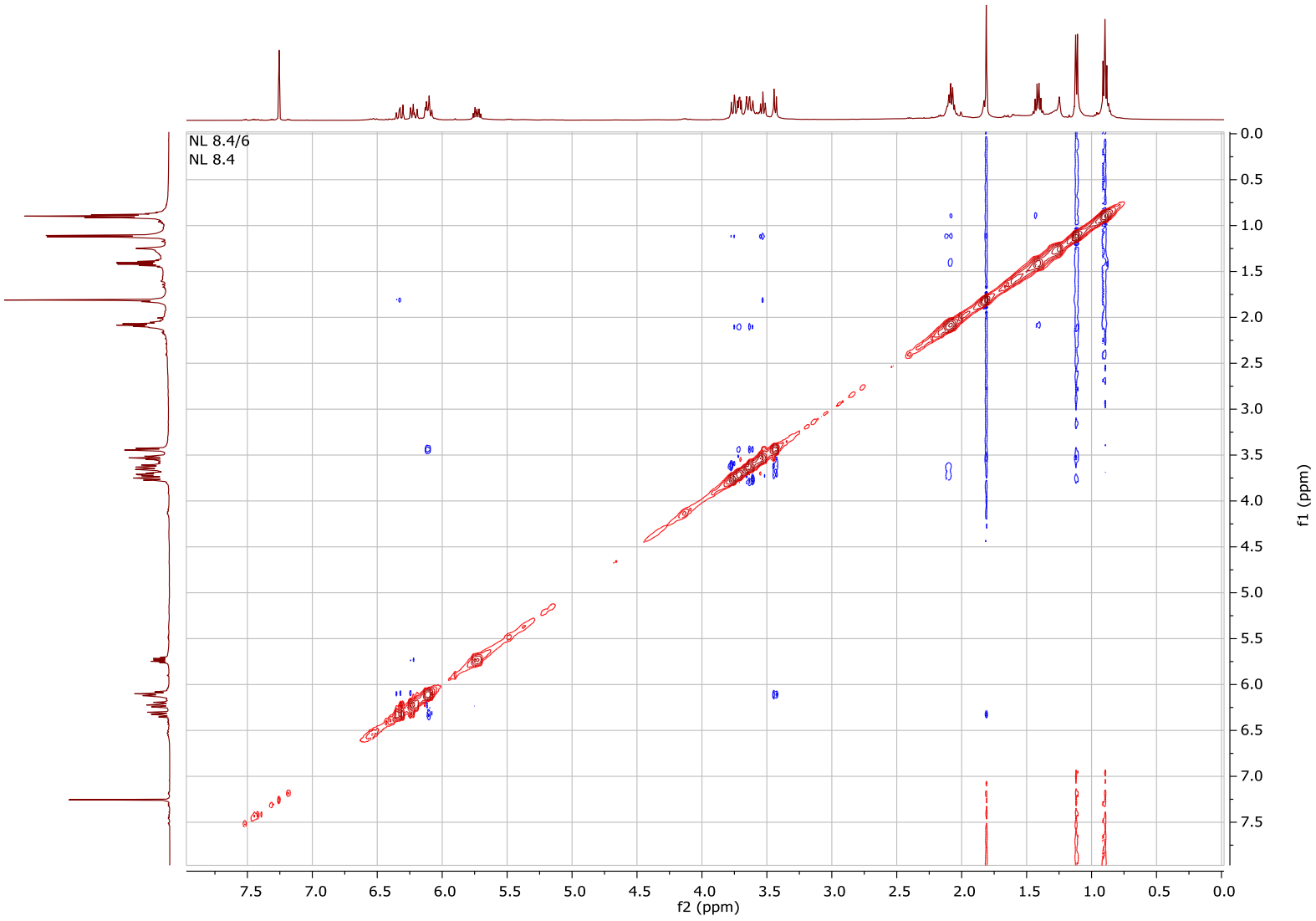


Figure S25. COSY NMR spectrum of 14 in $CDCl_3$

Figure S26. HSQC NMR spectrum of 14 in $CDCl_3$.

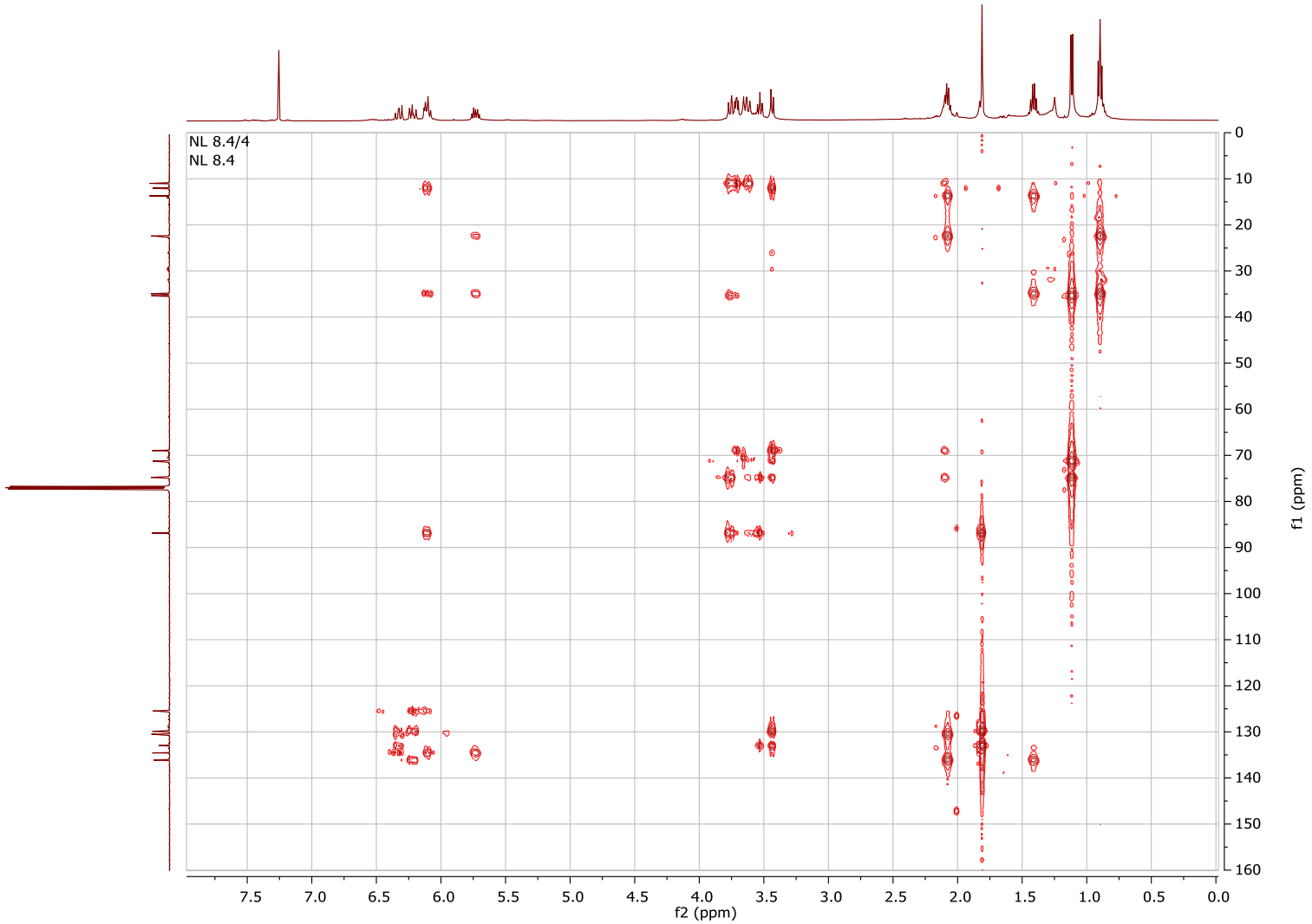


Figure S27. HMBC NMR spectrum of 14 in $CDCl_3$

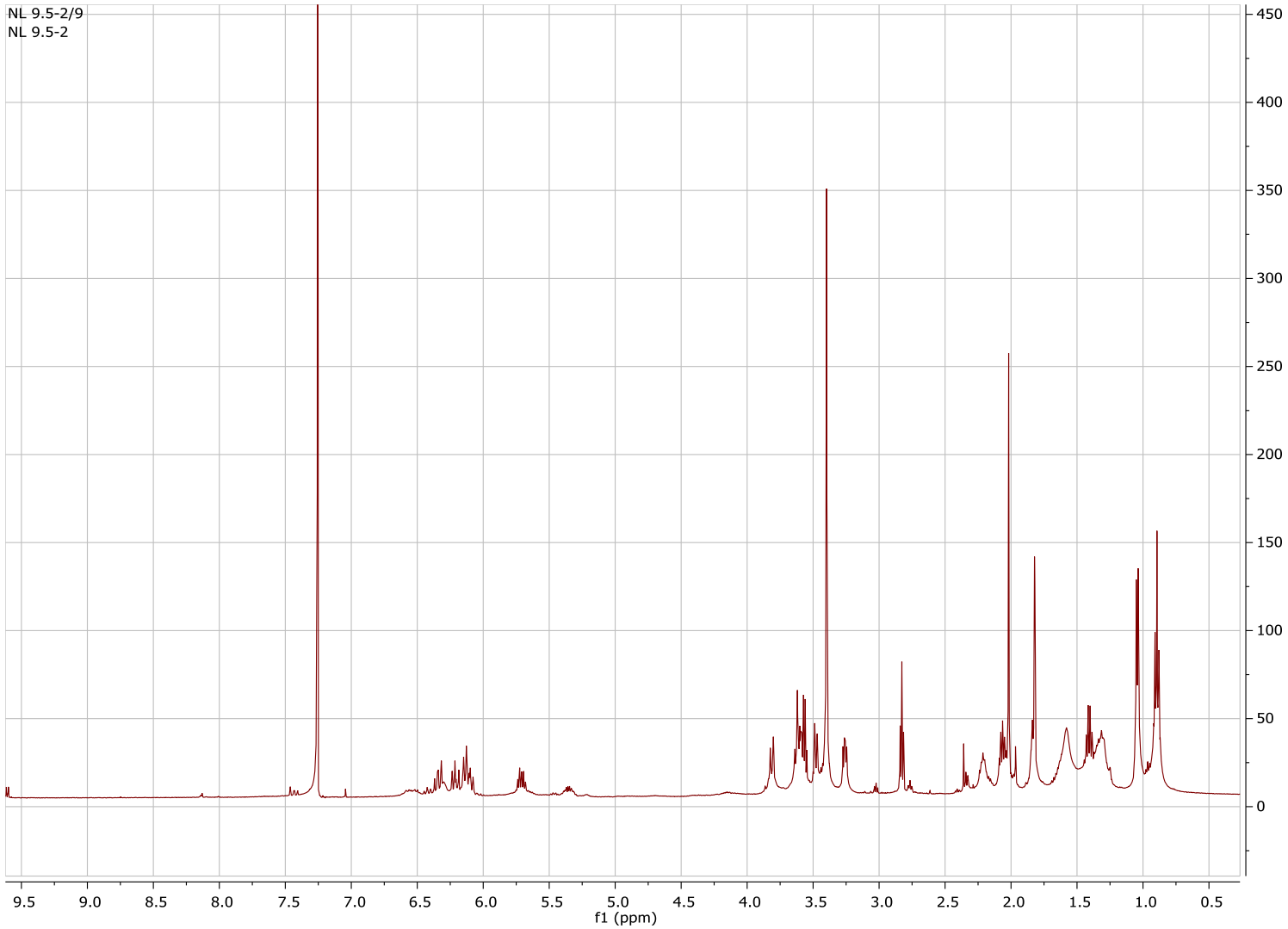


Figure S28. ^{1}H NMR spectrum of 15 in $CDCl_3$ (500 MHz).

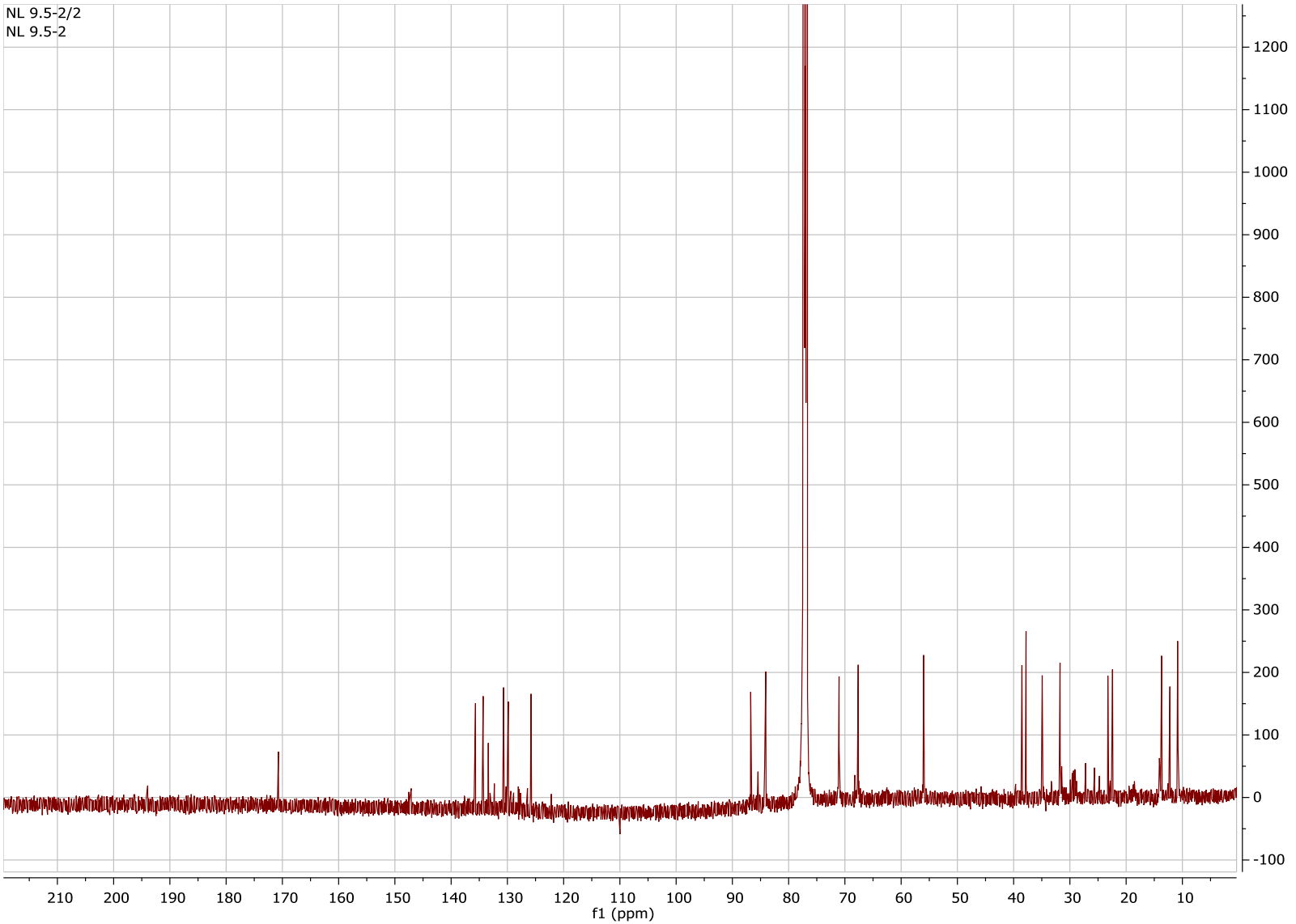


Figure S29. ^{13}C NMR spectrum of 15 in $CDCl_3$ (125 MHz).

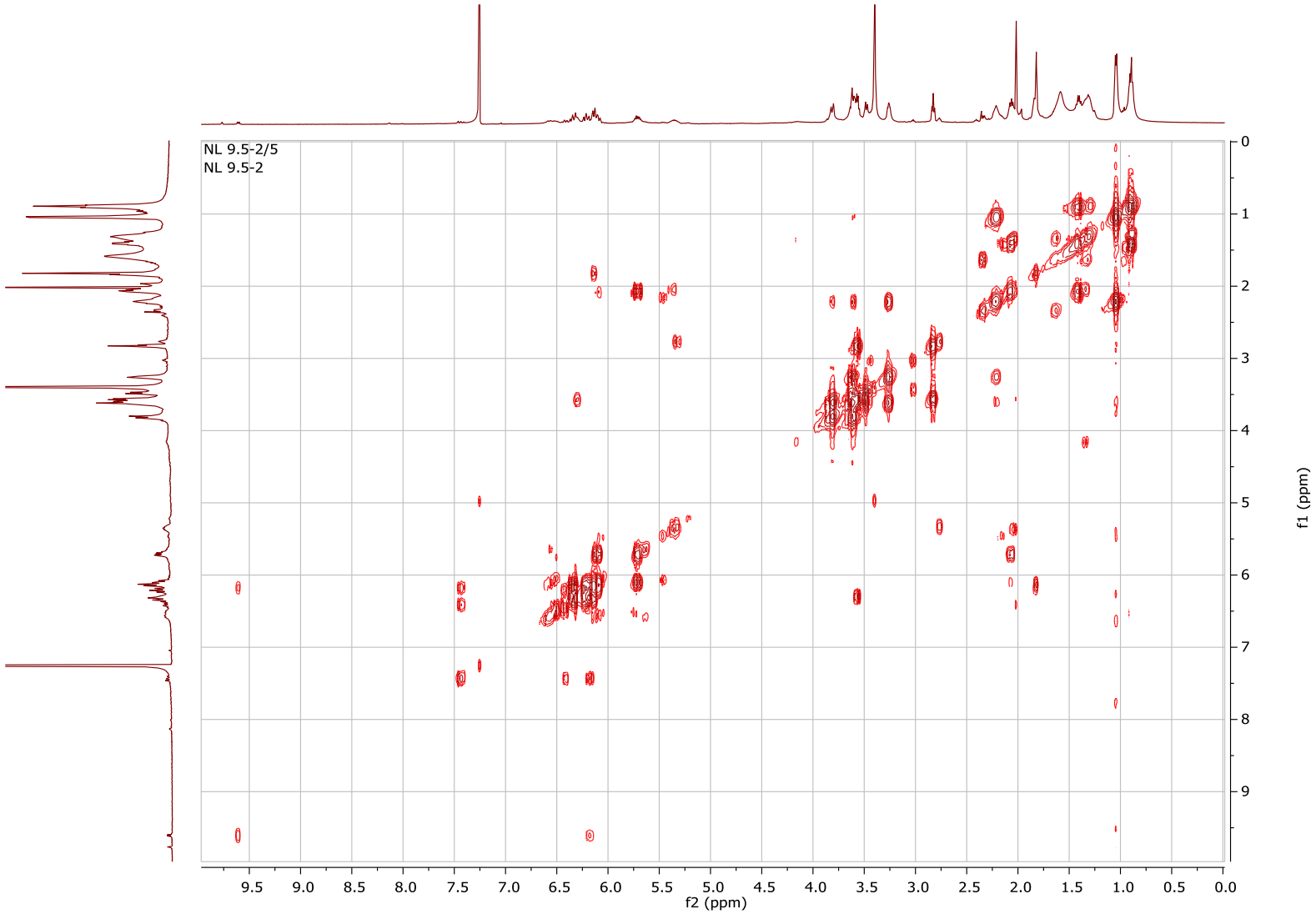


Figure S30. COSY NMR spectrum of 15 in $CDCl_3$

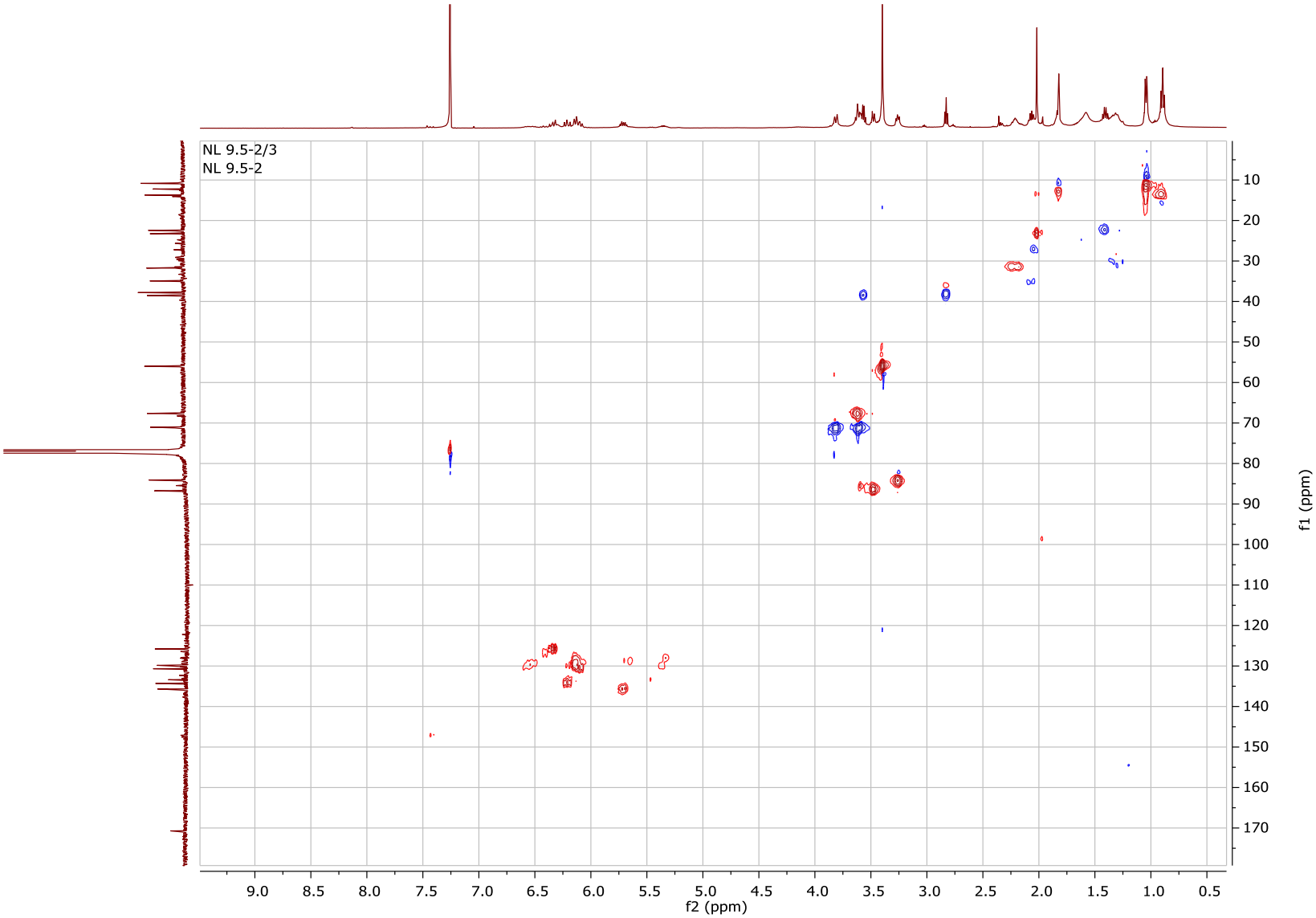


Figure S31. HSQC NMR spectrum of 15 in $CDCl_3$.

Figure S32. HMBC NMR spectrum of 15 in $CDCl_3$

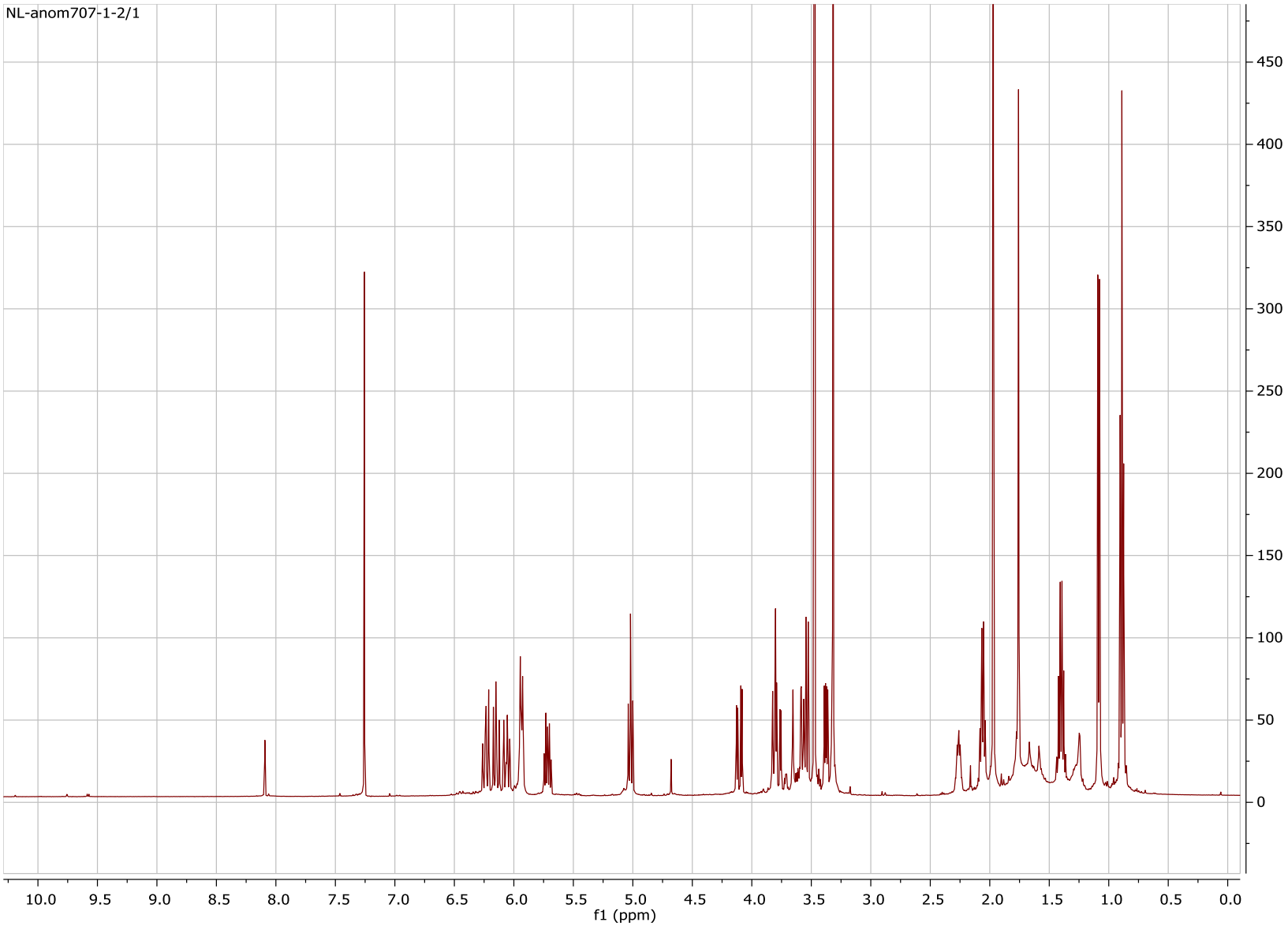


Figure S33. 1H NMR spectrum of 18 in CDCl3 (500 MHz).

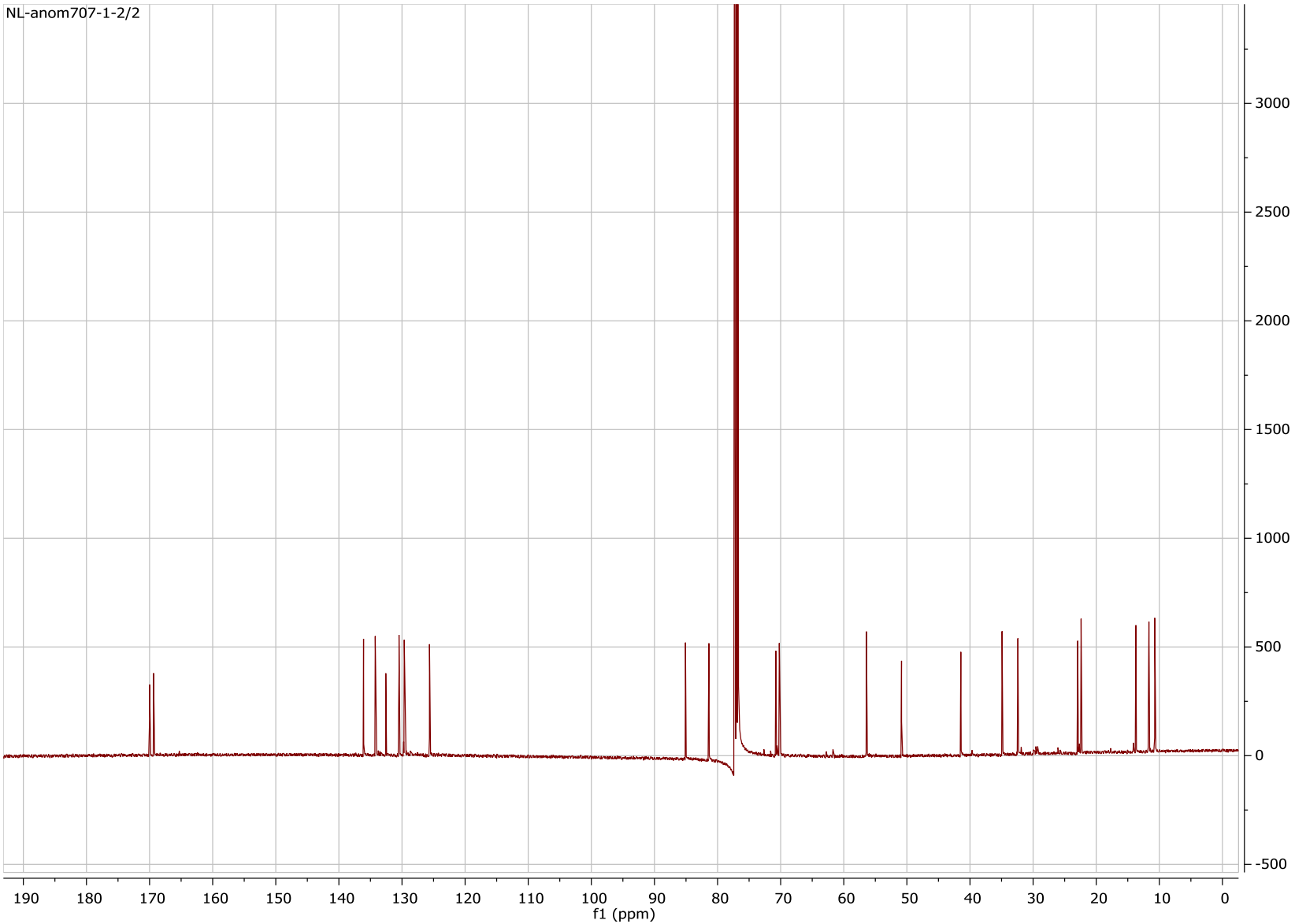


Figure S34. 13C NMR spectrum of 18 in CDCl3 (125 MHz).

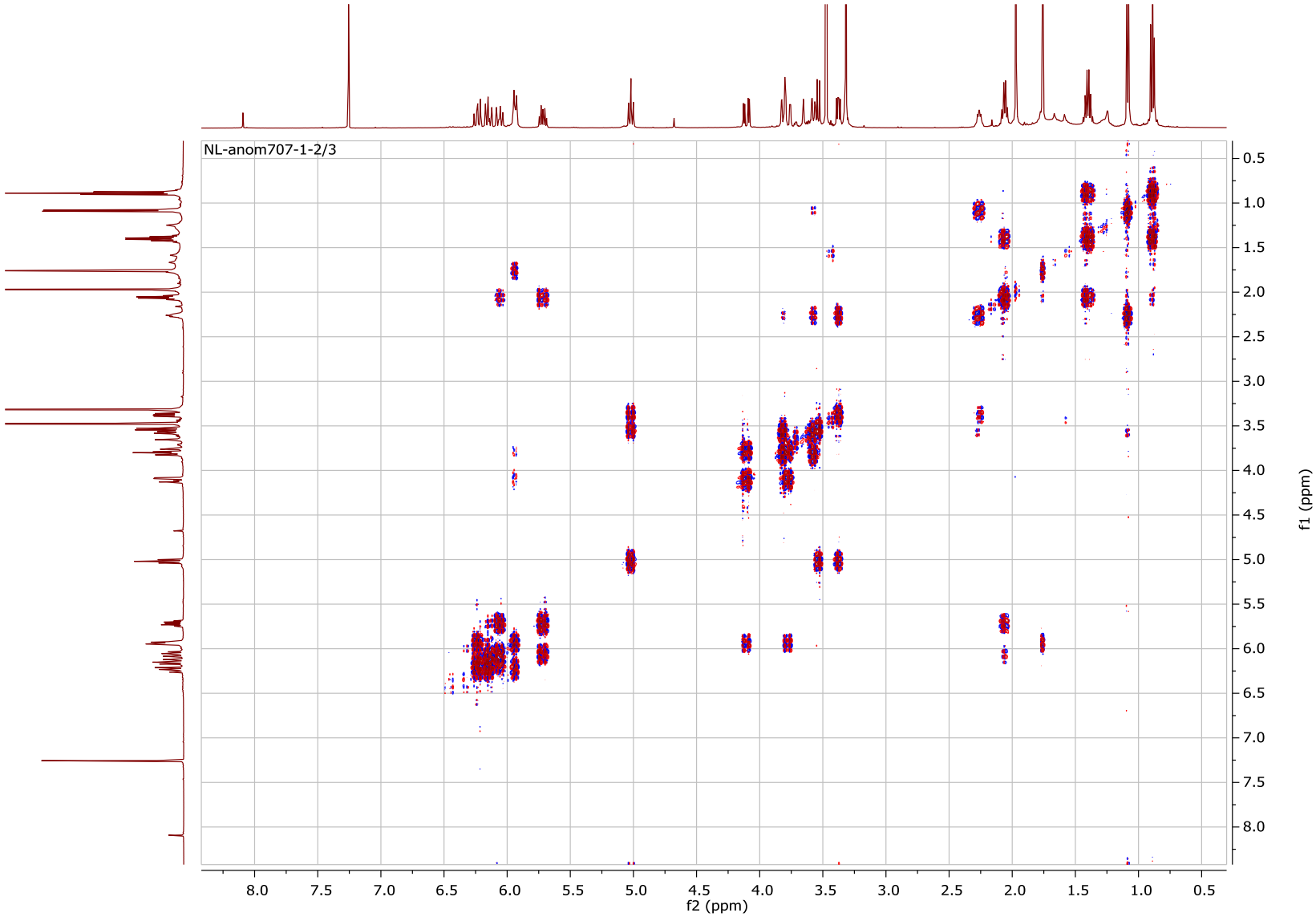


Figure S35. COSY NMR spectrum of 18 in CDCl3

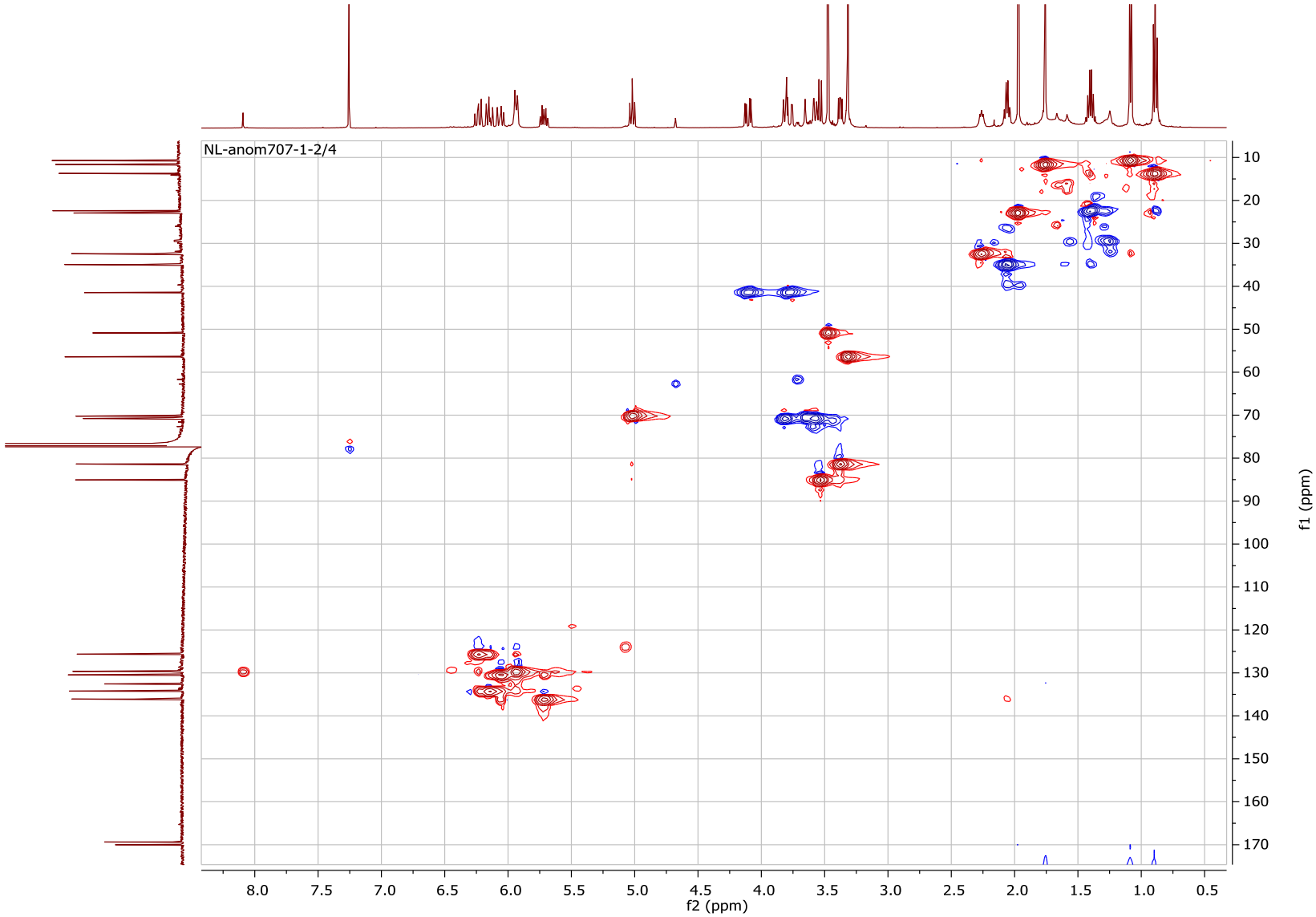


Figure S36. HSQC NMR spectrum of 18 in CDCl3.

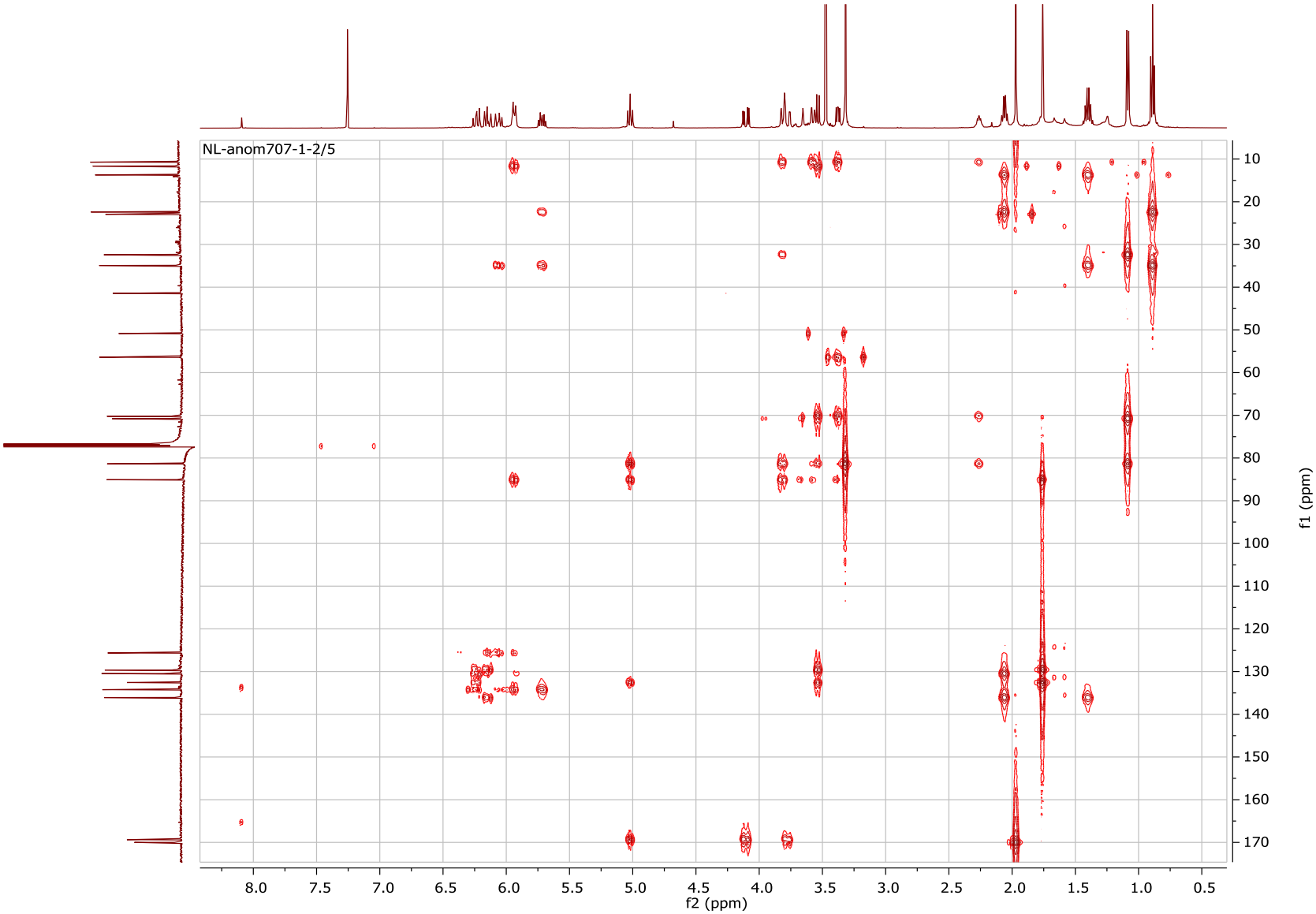


Figure S37. HMBC NMR spectrum of 18 in CDCl3

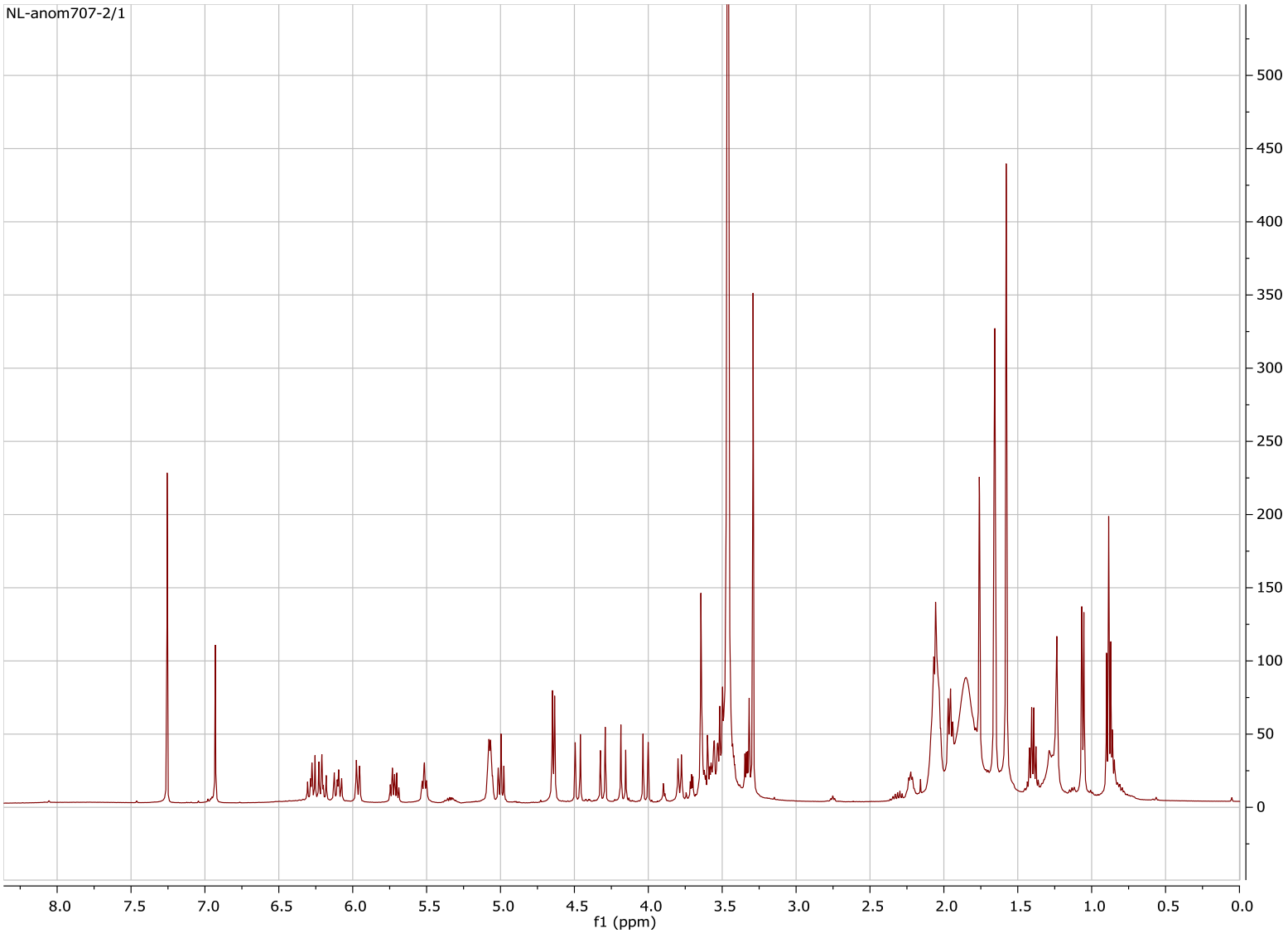


Figure S38. 1H NMR spectrum of 20 in CDCl3 (500 MHz).

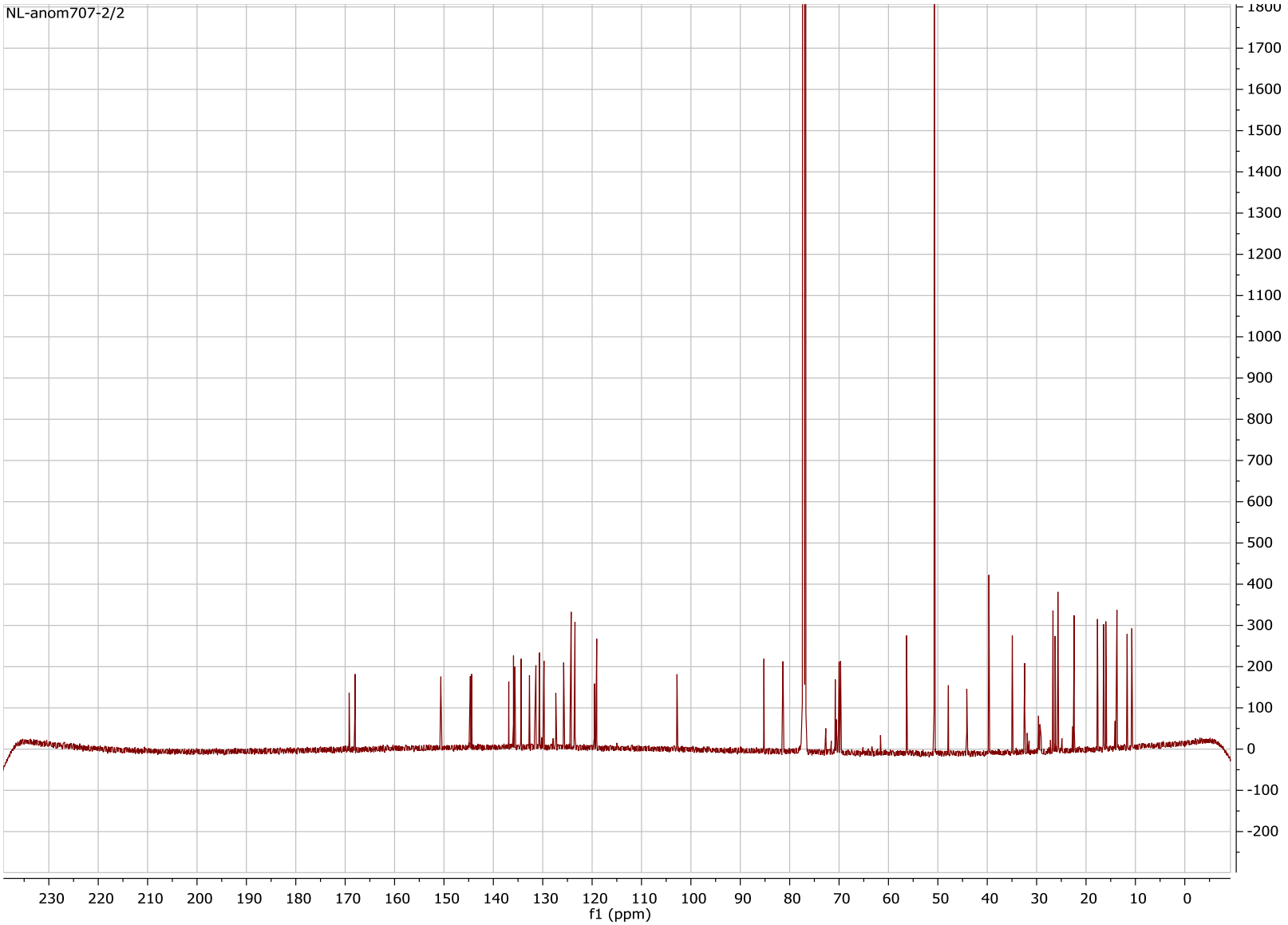


Figure S39. 13C NMR spectrum of 20 in CDCl3 (125 MHz).

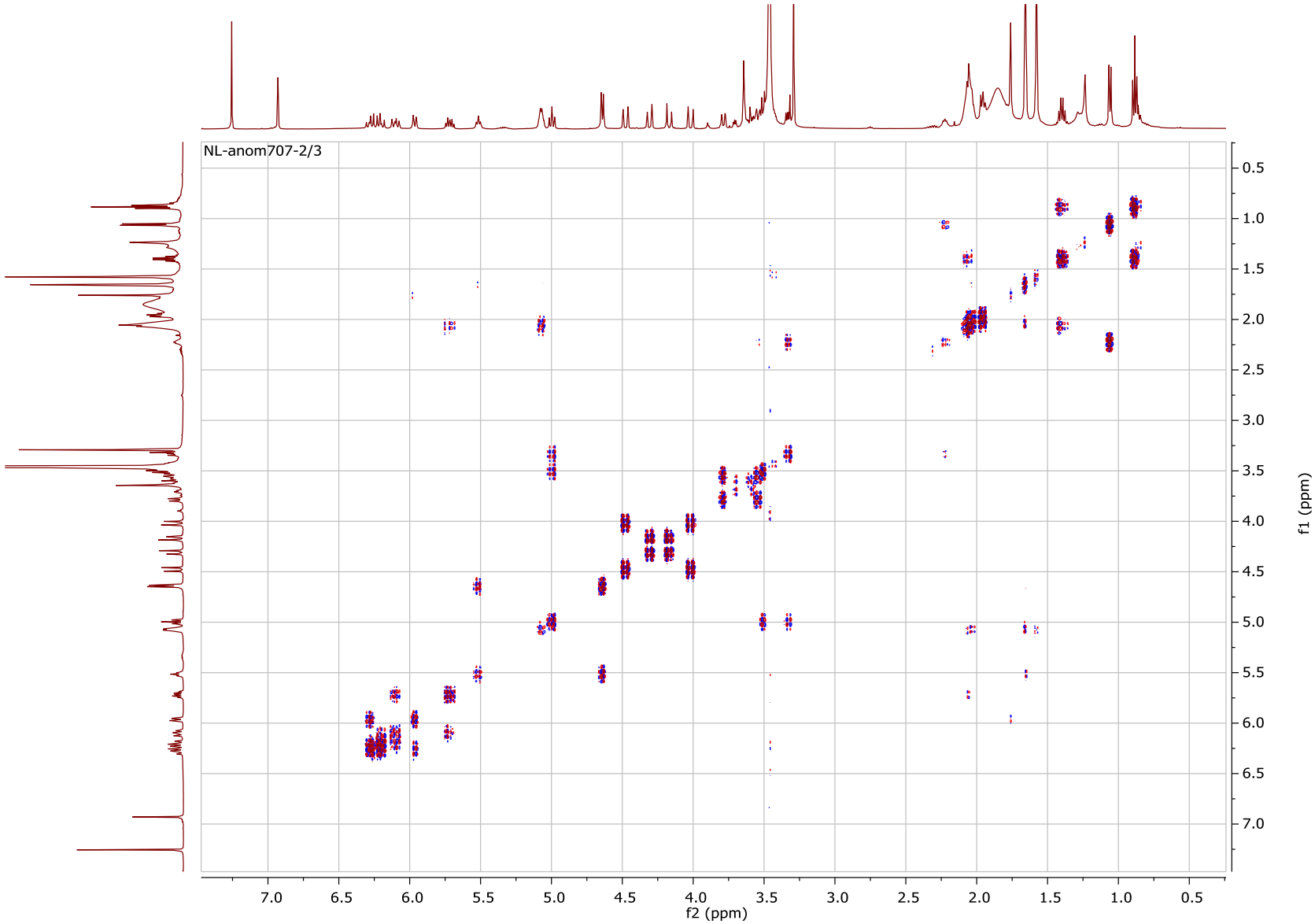


Figure S40. COSY NMR spectrum of 20 in CDCl3

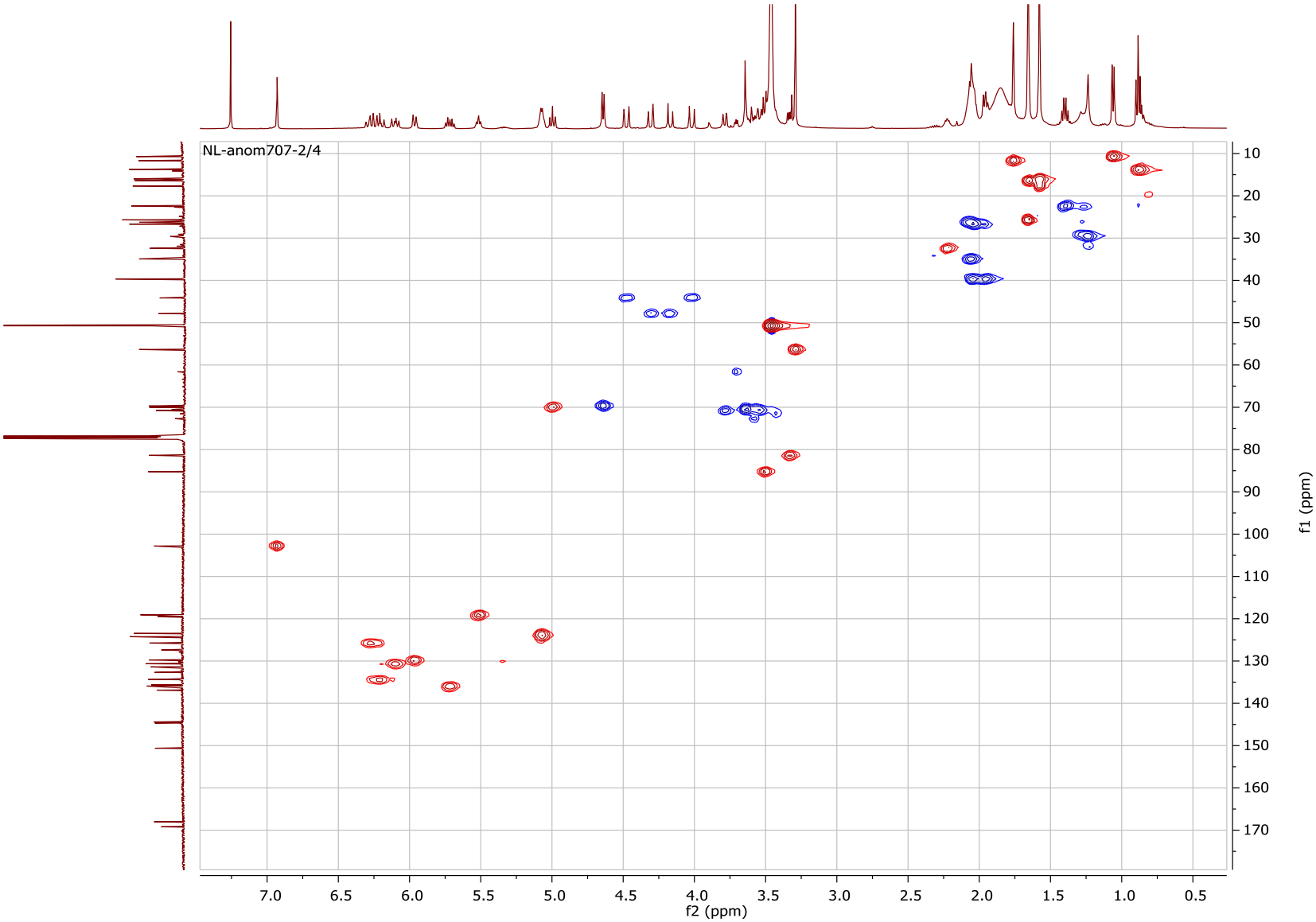


Figure S41. HSQC NMR spectrum of 20 in CDCl3

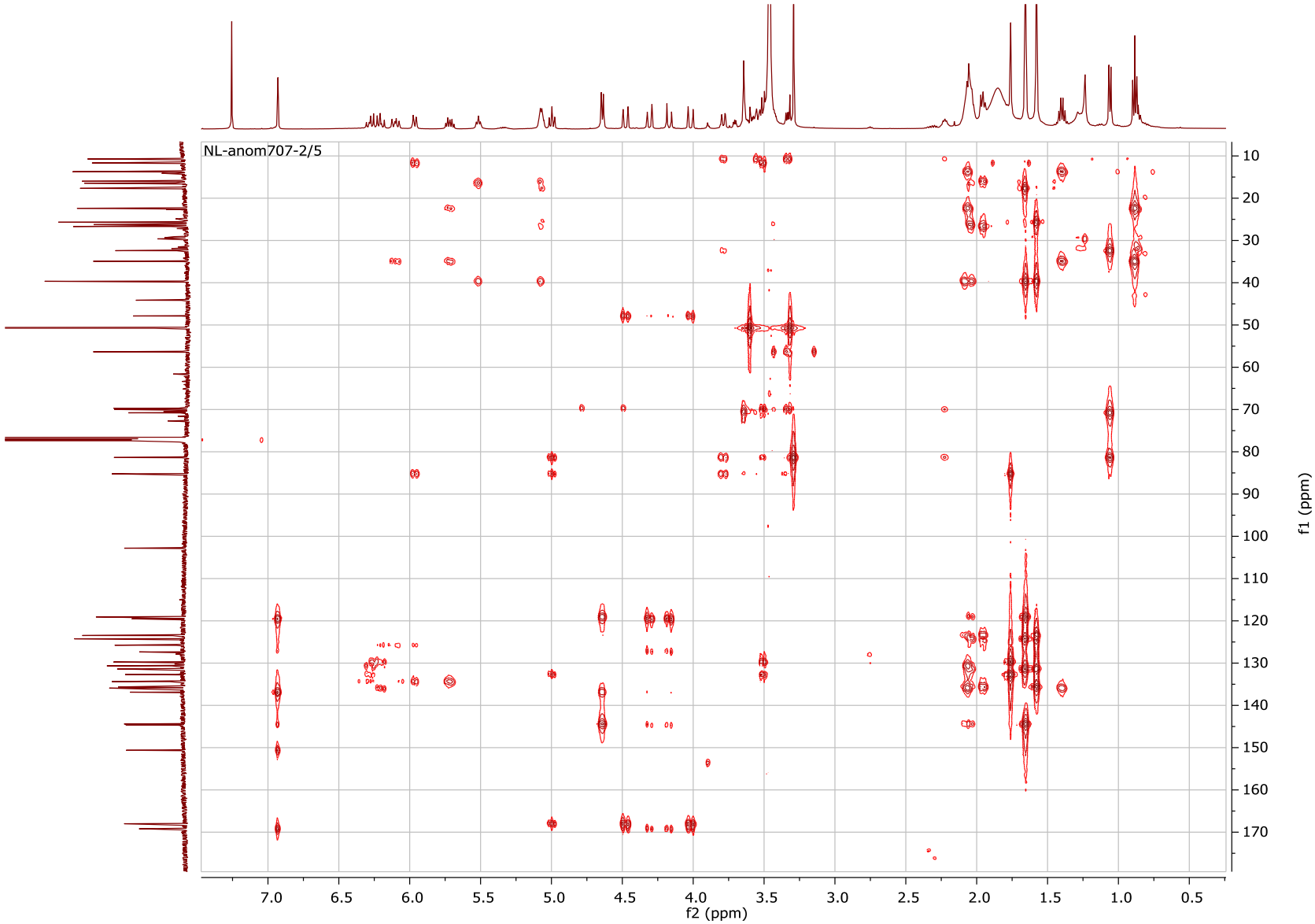


Figure S42. HMBC NMR spectrum of 20 in CDCl3

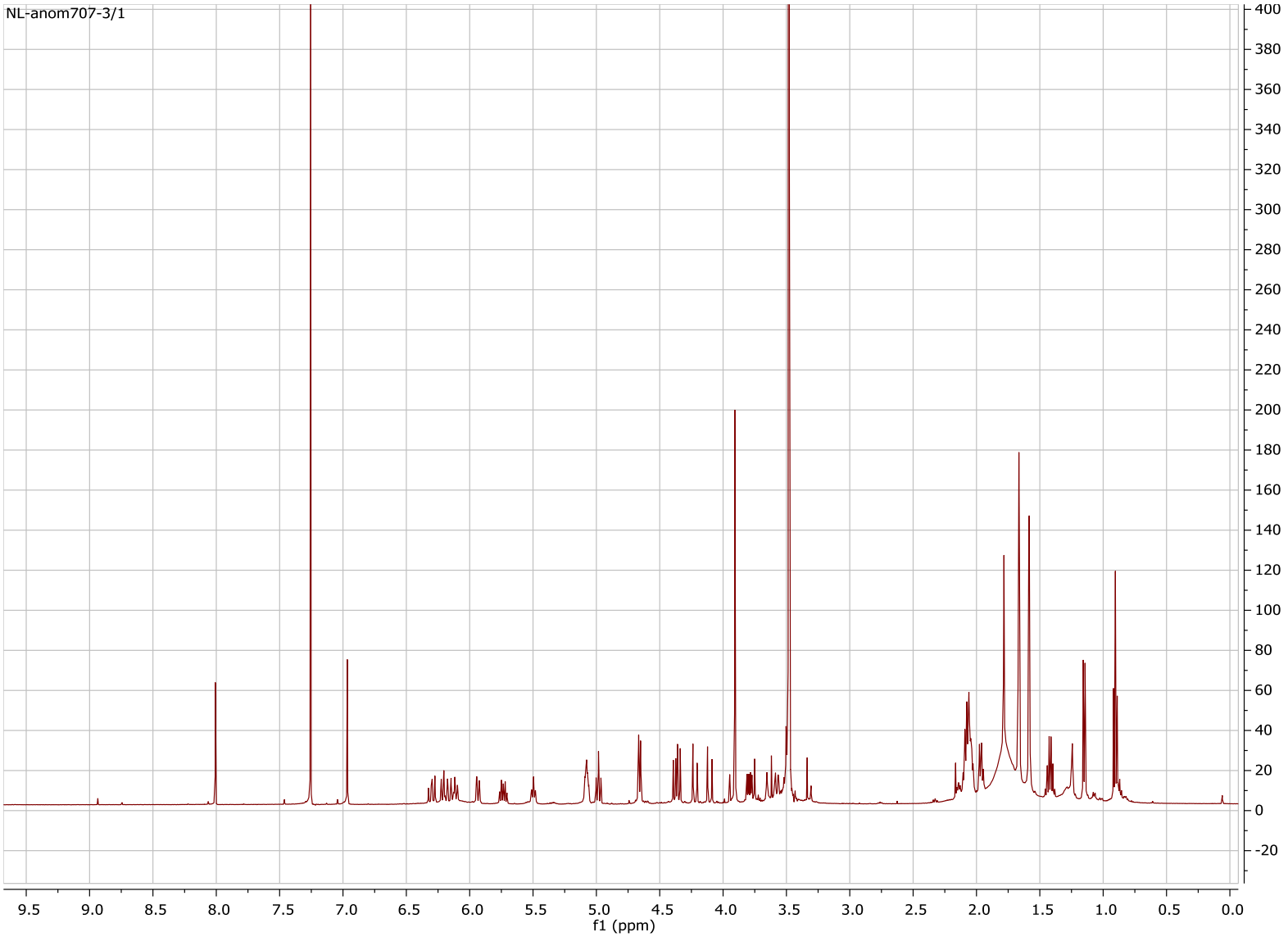


Figure S43. 1H NMR spectrum of 21 in CDCl3 (500 MHz).

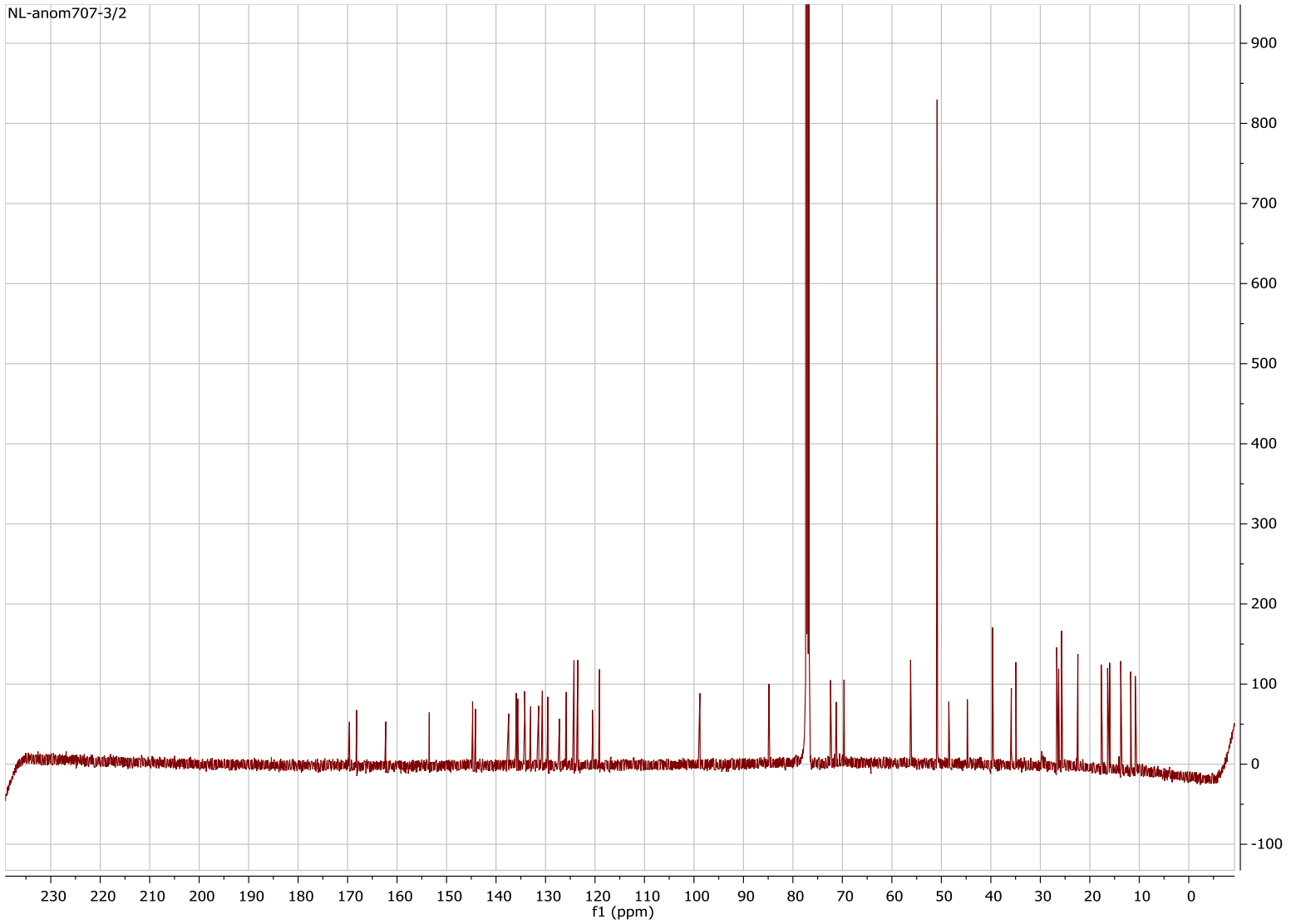


Figure S44. ^{13}C NMR spectrum of 21 in $CDCl_3$ (125 MHz).

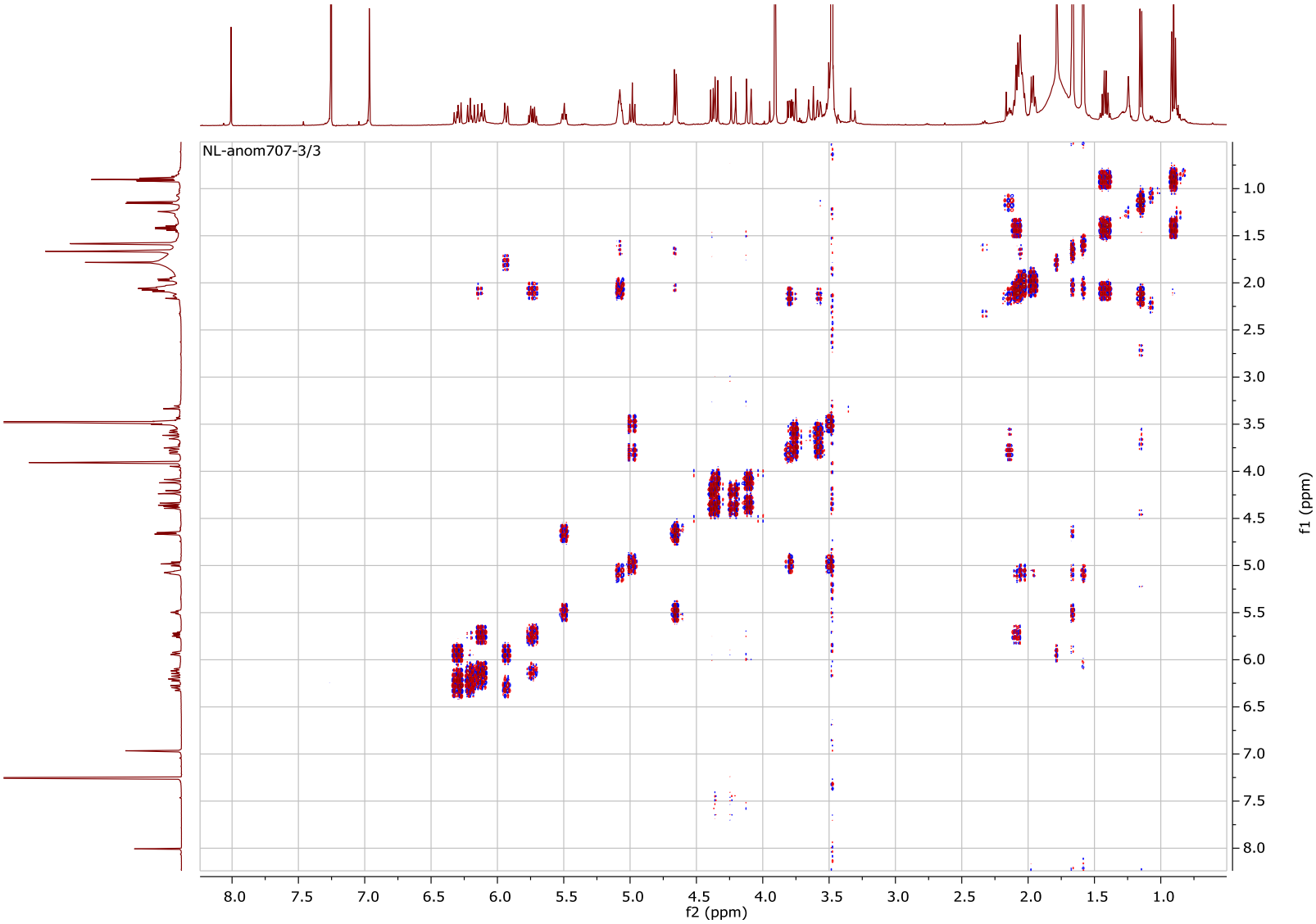


Figure S45. COSY NMR spectrum of 21 in $CDCl_3$

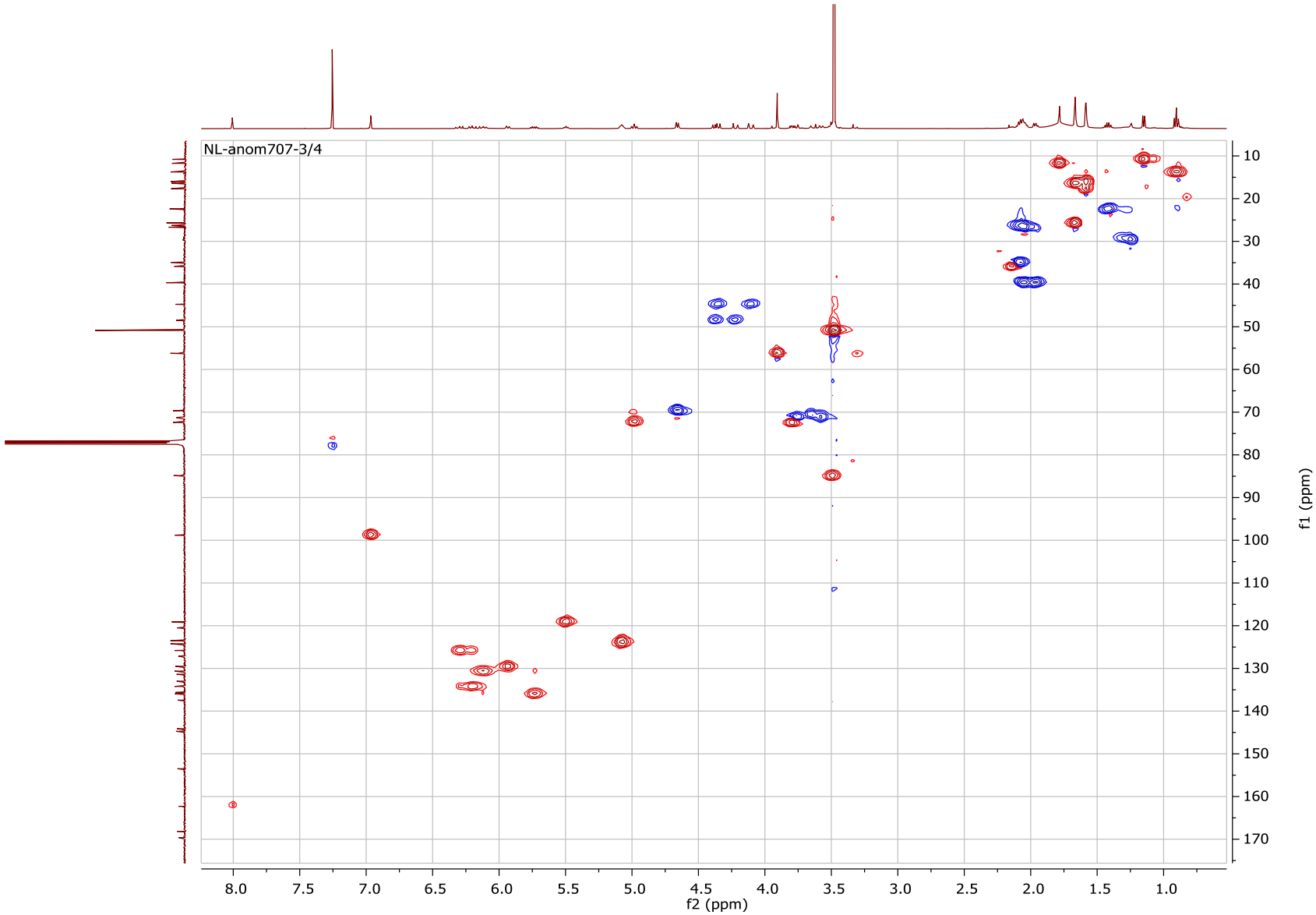


Figure S46. HSQC NMR spectrum of 21 in CDCl3

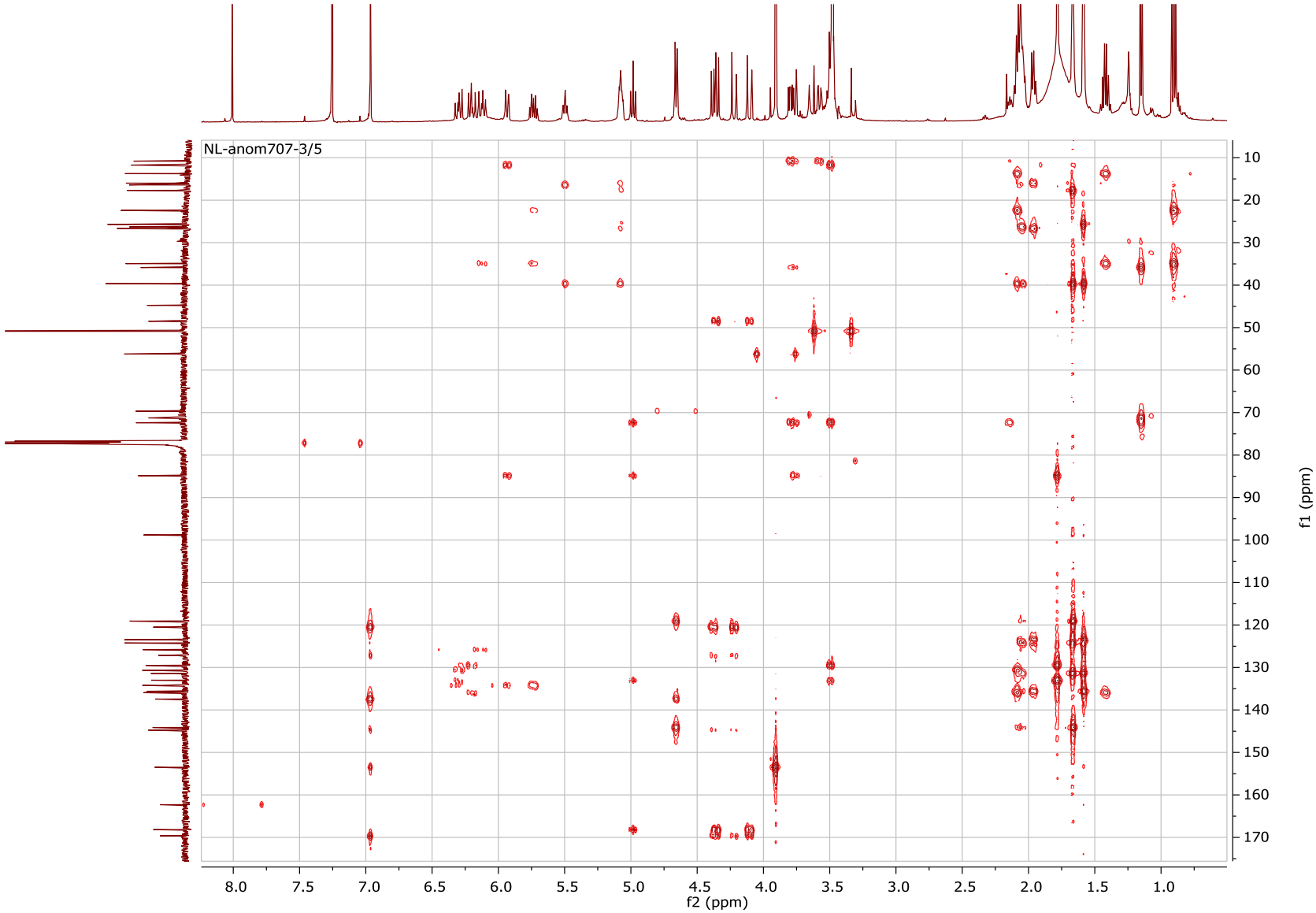


Figure S47. HSQC NMR spectrum of 21 in CDCl3

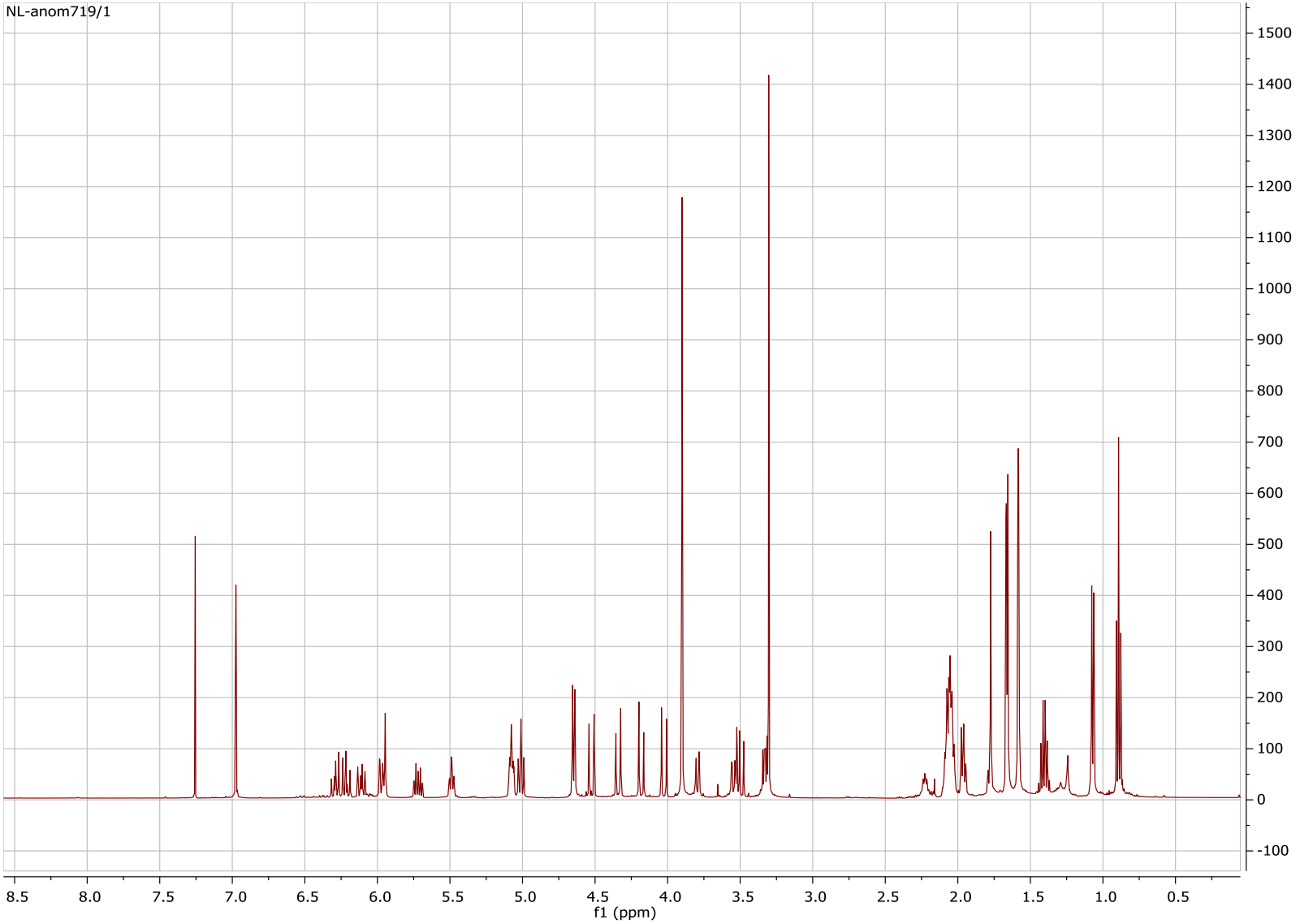


Figure S48. 1H NMR spectrum of 19 in CDCl3 (500 MHz).

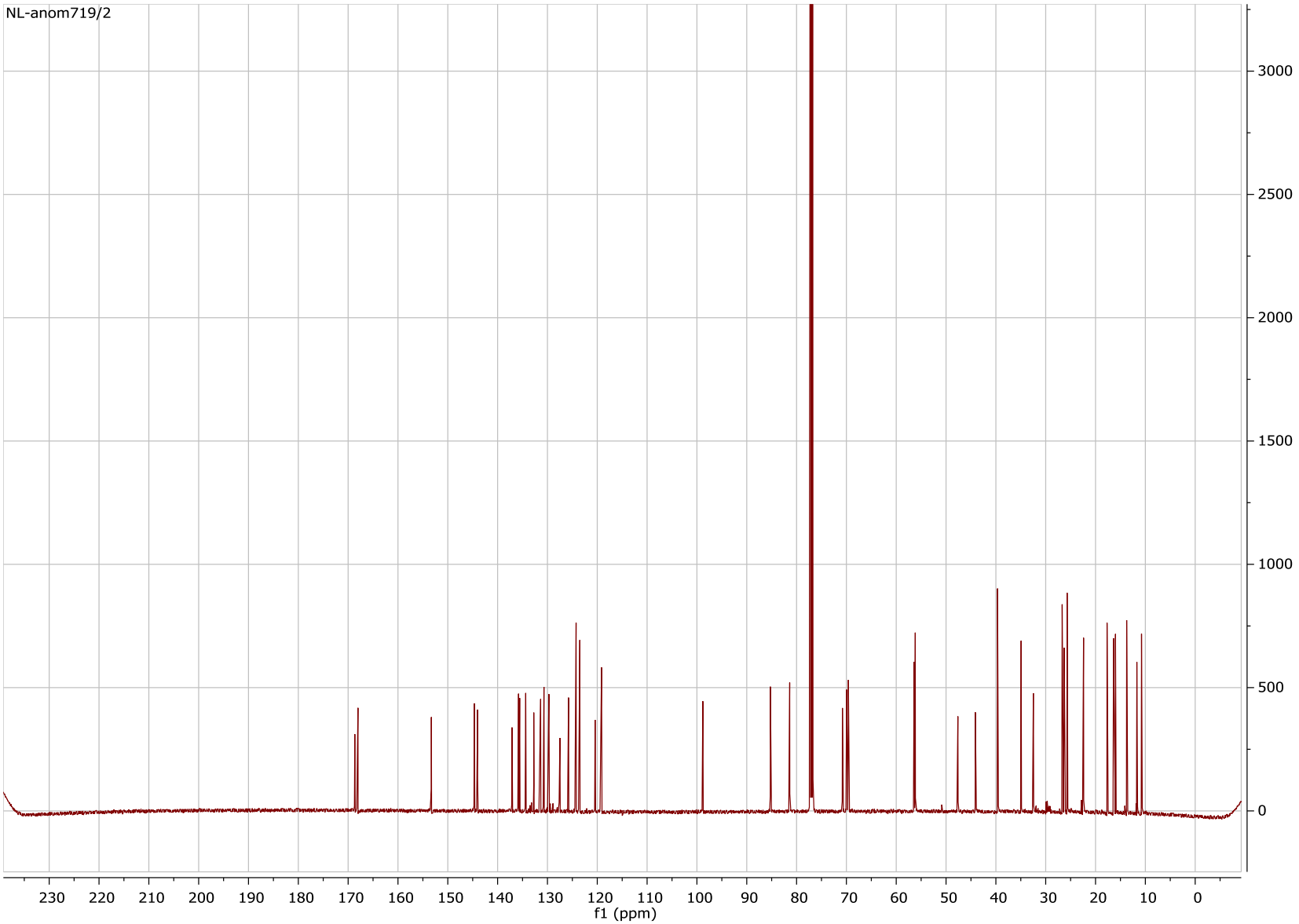


Figure S49. 13C NMR spectrum of 19 in CDCl3 (125 MHz).

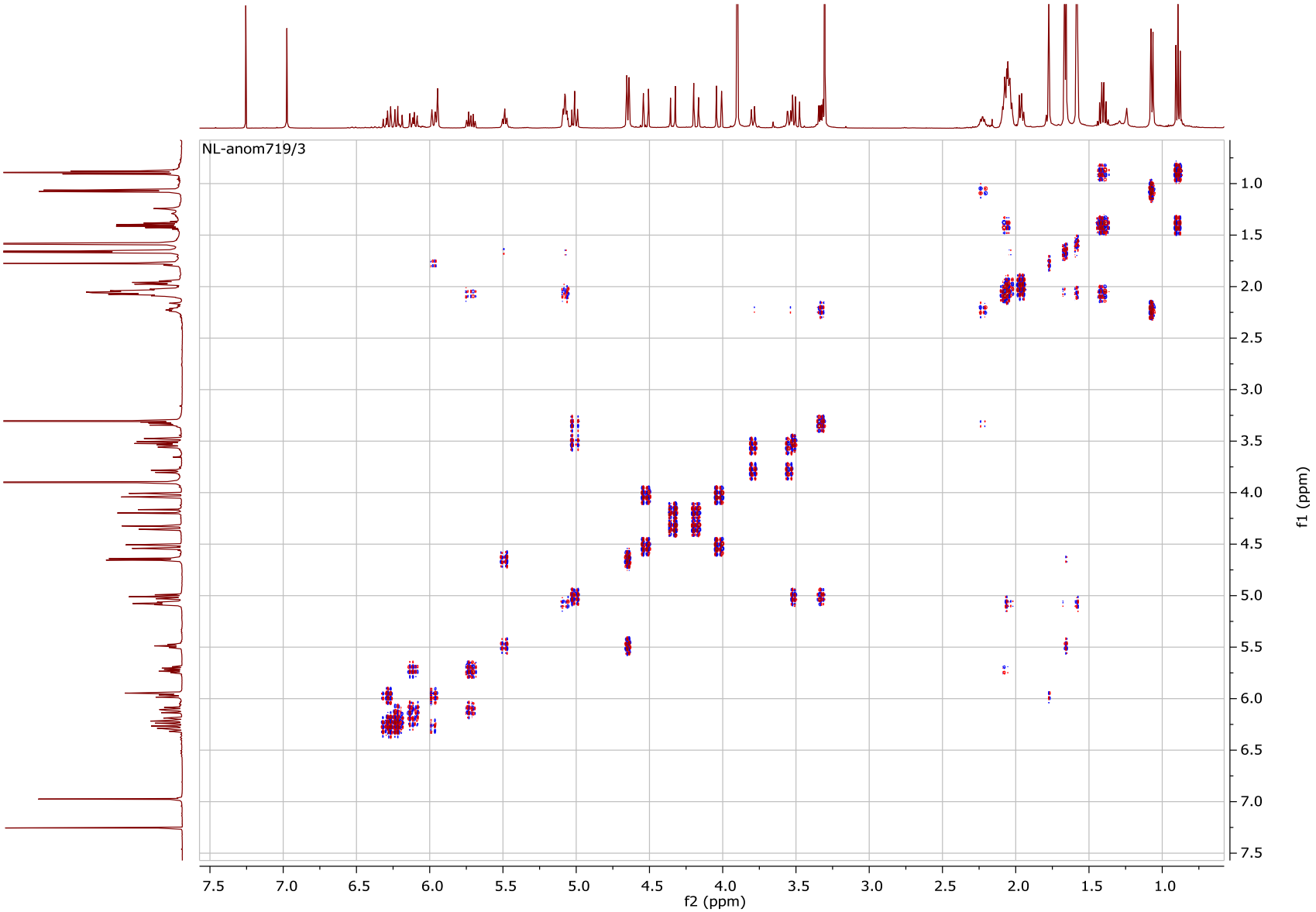


Figure S50. COSY NMR spectrum of 19 in CDCl3

Figure S51. HSQC NMR spectrum of 19 in CDCl3

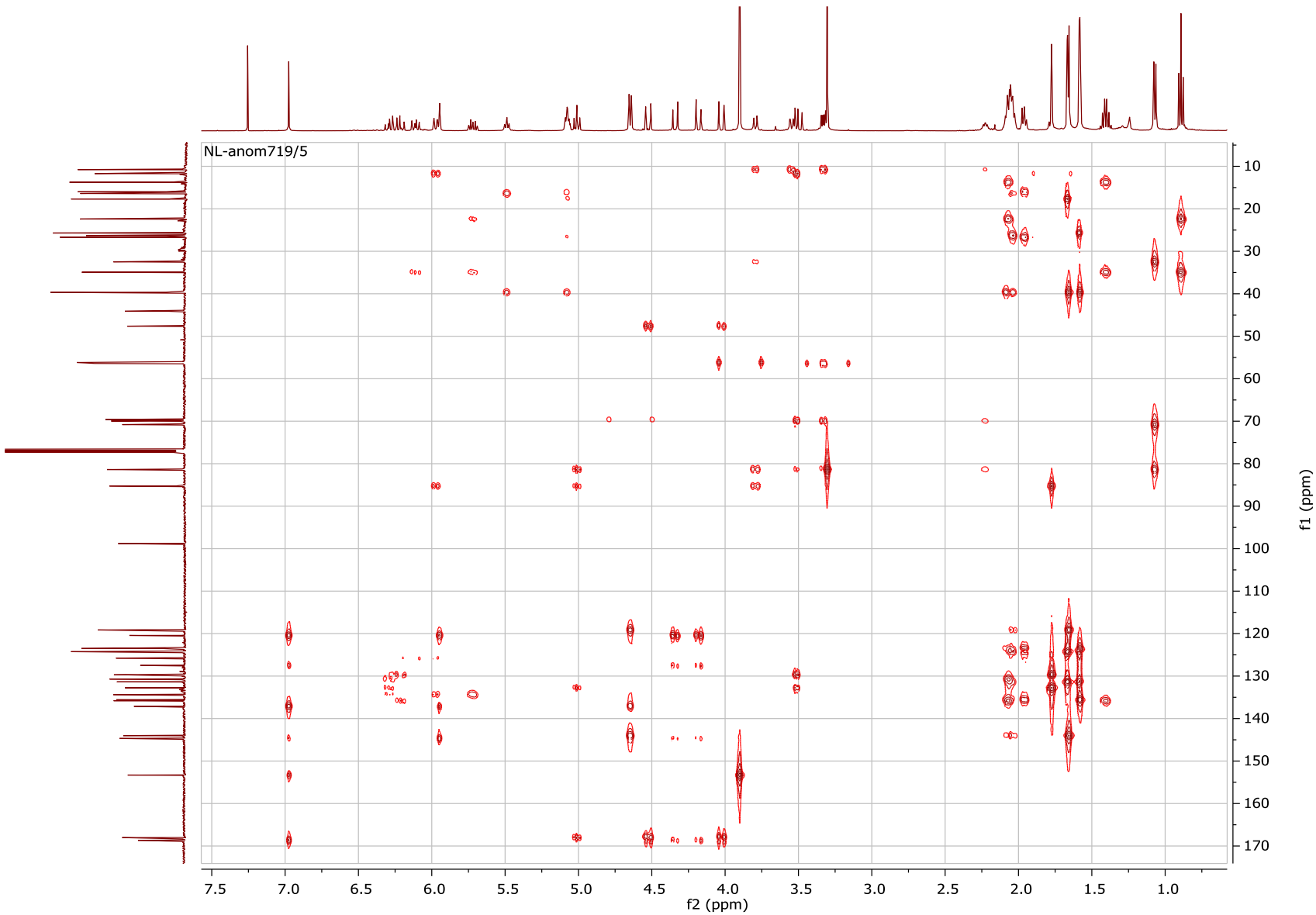


Figure S52. HMBC NMR spectrum of 19 in CDCl3

Table S10: List of Genome Sequences Analyzed with TGIF

List of Genome Sequences Analyzed with TGIF

1	A_versicolor_scaffolds.fasta	21	Eupen_K87.fasta	41	PenFKI.fasta	61	Talst1_2_AssemblyScaffolds.fasta
2	Acre_KY4917_FERM_K87_scafSeq.fasta	22	FERM_K87_scafSeq.fasta	42	Pen_citr_GCA_001950535.fasta	62	Tg_K87scafSeq.fasta
3	Acremoniumzaea-Kmer87.fasta	23	FND171-Aspergillus+terreus ATCC 20156.fasta	43	PenchWisc1_1_AssemblyScaffolds.fasta	63	Tolinf1_AssemblyScaffolds.fasta
4	Apimo1_AssemblyScaffolds.fasta	24	FND171-Aspergillus+terreus.fasta	44	Penex_AssemblyScaffolds.fasta	64	Trichoderma_virens_v2.fasta
5	Asp-alliaceous_Contigs.fasta	25	FND172-BP-5715.fasta	45	Penicillium funiculosum Thom funiculosin.fasta	65	Triha1_AssemblyScaffolds.fasta
6	Asp-fischeri_Contigs.fasta	26	FND173-Cladobotryum parnafungin.fasta	46	Penicillium herquei.fasta	66	Tw_K81_scafSeq.fasta
7	Aspbom1_AssemblyScaffolds.fasta	27	FND173-Cladobotryum.fasta	47	Penicillium oxalicum_K85_scaff.fasta	67	asp_nom.fasta
8	Aspca3_scaffolds.fasta	28	FNF033.fasta	48	Penicillium sp.HDN14-431.fasta	68	aspergillus_clavatus_1_contigs.fasta
9	Aspcand1.fasta	29	Fusco1_AssemblyScaffolds.fasta	49	Penla1_AssemblyScaffolds.fasta	69	az_kmer91_scafSeq.fasta
10	Aspfl1_AssemblyScaffolds.fasta	30	GCF_000149205.fasta	50	Pest_fici_GCA_000516985.1_PFICI_genomic.fasta	70	az_spades_contigs.fasta
11	Aspfu1_AssemblyScaffolds.fasta	31	Lenfl1_AssemblyScaffolds.fasta	51	Ph_K79_scafSeq.fasta	71	cladobotyrum_polished_assembly.fasta
12	Asplep1_AssemblyScaffolds.fasta	32	Microcera lavarum parnafungin 2.fasta	52	Po_K85_scaff.fasta	72	gib_zea_GCA_000240135.3_ASM24013v3_genomic.fasta
13	Aspte1_AssemblyScaffolds.fasta	33	Neucr2_AssemblyScaffolds.fasta	53	Pocchl1_AssemblyScaffolds.fasta	73	p_thy.fasta
14	Asptu1_AssemblyScaffolds.fasta	34	P. thomii R-89.fasta	54	Pt_K85_scafSeq.fasta	74	pcucu_spades_short_renamed_2.fasta
15	CA_K87_scafSeq.fasta	35	P.expansum.fasta	55	PvH1.fasta	75	penicillium FKI 1983 citridone.fasta
16	C_lunata.fasta	36	PExpansum_NRRL976.fasta	56	Saro_oryzae_GCA_001972265.1_JCM_12450_assembly_v001_genomic.fasta	76	penicillium_variabile.fasta
17	Canalb1_AssemblyScaffolds.fasta	37	PTLM2_K87_scafSeq.fasta	57	Sporormiela australis australianfungin.fasta	77	po_scaffolds.fasta
18	Cladobotryum sp. No. 11231.fasta	38	Pc_K85.scafSeq.fasta	58	Strep_nour_GCA_001704275.1_ASM170427v1_genomic.fasta	78	sar_ory_GCA_001605845.1_ASM160584v1_genomic.fasta
19	Cordy_ophio_GCA_001189435.1_Tophv1.fasta	39	Pcurcumernia_K85scafSeq.fasta	59	Strep_sp_mg1_GCA_000412265.2_ASM41226v2_genomic.fasta	79	sordaria_araneosa_contigs.fasta
20	Cs_K85_scaf.fasta	40	Pen.funiculosum.fasta	60	Tal_islan_GCA_000985935.1_PIS_genomic.fasta	80	zal_arb_GCA_000409485.1_GLAREA_genomic.fasta

List of MATLAB® scripts for TGIF

Script 1: mbdmadedb.m

```
%%
%
% <<mbmadedb.m>>
%
clear all
%%%%%%% Be sure to specify directory of where genomes are located
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
%                                   directory                                  =
'C:\Users\Lab\Desktop\Genome_Mining_Matlab\antifungal_tests\mining';
cd(directory);
% cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes
gbfiles = dir('*.fa');
%
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';

for i=1:length(gbfiles)

   [~,c]= fileparts(gbfiles(i).name);
   newnamegb = strcat(c,'.fasta');
   gbname = struct2cell(gbfiles(i));
   namegb = gbname{1};

   copyfile(namegb,newnamegb);
end
fasfiles = dir('*.fas');
for i=1:length(fasfiles)

   [~,c]= fileparts(fasfiles(i).name);
   newnamefas = strcat(c,'.fasta');
   fasname = struct2cell(fasfiles(i));
   namefas = fasname{1};

   copyfile(namefas,newnamefas);
end
fasfiles = dir('*.aa');
for i=1:length(fasfiles)

   [~,c]= fileparts(fasfiles(i).name);
   newnamefas = strcat(c,'.fasta');
   fasname = struct2cell(fasfiles(i));
   namefas = fasname{1};

   copyfile(namefas,newnamefas);
end

fastafiles = dir('*.fasta');
key = strings(length(fastafiles),2);
for i=1:length(fastafiles)
```

```
    key(i,1) = i;
end

% mkdir resultsfolder;
mkdir(resultsfolder);
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
%
copyfile('blastall.exe','C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_g
enomes\resultsfolder');
%
copyfile('formatdb.exe','C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_g
enomes\resultsfolder');
copyfile('blastall.exe',directory);
copyfile('formatdb.exe',directory);
cd(directory);
copyfile('blastall.exe',resultsfolder);
copyfile('formatdb.exe',resultsfolder);

for i=1:length(fastafiles)
    [~,f]=fileparts(fastafiles(i).name);
    newnamefasta = sprintf('%d.fasta',i);
    fastaname = struct2cell(fastafiles(i));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(i,2) = namefasta;

    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,resultsfolder);

end

file1 = sprintf('genome_key');
xlswrite(file1, key);

cd(directory);
cd(resultsfolder)

fastafiles = dir('*.fasta');
n = length(fastafiles);

for i=1:n
    dbname = sprintf('%d.fasta', i);
    outname = num2str(i);
    blastformat('Inputdb', dbname,'protein','false');
%     blastformat('-i dbname -t outname -p F');
end
% for i=1:n
%     %dbname = sprintf('%d.fasta', i);
%        dbname  =  sprintf('makeblastdb.exe  -in  %d.fasta  -dbtype  nucl  -out
%d',i,i);
%     system(dbname);
% end

%%%%%%%%%%%%%%organize targets%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
```

```
% cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\antifungal_tests
%cd TARGETS
% cd targets_custom
% cd targets_JB_DY
cd Cyp51

targetfiles = dir('*.fasta');
targetkey = strings(length(targetfiles),2);
for i=1:length(targetfiles)

   [~,c]= fileparts(targetfiles(i).name);
   newnametarget = sprintf('T%d.fa',i);
   targetname = struct2cell(targetfiles(i));
   nametarget = targetname{1};
   copyfile(nametarget,newnametarget);
   directory1 = strcat(directory,'/',resultsfolder);
   movefile(newnametarget,directory1);
   targetkey(i,2) = c;
   targetnum = sprintf('T%d',i);
   targetkey(i,1) = targetnum;
end


file2 = sprintf('target_key');
xlswrite(file2, targetkey);


% cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
% cd TARGETS
% targetfiles = dir('*.fasta');
% for i=1:length(targetfiles)
%        namefas = sprintf('T%d.fasta',i);
%    targetname = sprintf('T%d.fa',i);
%    copyfile(namefas,targetname);
%
%movefile(targetname,'C:\Users\Lab\Desktop\Genome_Mining_Matlab\test\resultsf
older');
%
%movefile(targetname,'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_stra
ins\resultsfolder');
%
movefile(targetname,'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genom
es\resultsfolder');
% end

cd(directory);
cd(resultsfolder)
save('genomes.mat','key','targetkey');
```

Script 2: mbblast.m

```
%%%%%%%%%%%%%%blast targets%%%%%%%%%%%%%%%%%%%%%%%%%
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
%                                 directory                                    =
'C:\Users\Lab\Desktop\Genome_Mining_Matlab\antifungal_tests\mining';
cd(directory);

% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)

targetfiles = dir('*.fa');
x = length(targetfiles);%% x = target number
gbfiles = dir('*.nhr');
n = length(gbfiles);


results = struct('gen_tar',cell(x,n));
%results = struct('gen_tar',[],'genome',[],'target',[],'output',[]);

idx = 0;
for a = 1:x
    for b =1:n
query = sprintf('T%d.fa',a);
outname = sprintf('T%dhits_in_%d.txt',a,b);
outname1 = sprintf('T%dhits_in_%d',a,b);
dbname = sprintf('%d.fasta',b);
idx = idx +1;
output  =  blastlocal('InputQuery',  query  ,'Database',  dbname  ,'Program',
'tblastn','ToFile',outname);
%output  =  blastlocal('InputQuery',  query  ,'Database',  dbname  ,'Program',
'tblastn');
results(a,b).gen_tar = outname1;
results(a,b).genome = b; %genome
results(a,b).target = a; %target
results(a,b).output = output;
    end
%results = blastlocal('-i T2.fasta -d 1 -p tblastn');
end
cd(directory);
cd(resultsfolder)
save('targetresults.mat','results','targetkey');
```

Script 3: colocalblast_mb.m

```
%%%%Colocalization of gene clusters
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);


% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
```

```
cd(resultsfolder)
targetfiles = dir('*.fa');
numtargets = length(targetfiles);%% x = target number
gbfiles = dir('*.nhr');
numgenomes = length(gbfiles);

%                                   blastfiles                                          =
'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes\resultsfolder';
%specify here the location of blast files
blastfiles                                                                              =
strcat('C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2\',results
folder);
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
cd 2nd_met_genes_core

%%%%%%%%%%%%%%%%%%%%copy and move files%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%copy secondmet domain files
for i = 1:5
    dom = sprintf('NRPS_A_%d.fasta',i);
    copyfile(dom,blastfiles);
end
for i = 1:5
    dom = sprintf('NRPS_C_%d.fasta',i);
    copyfile(dom,blastfiles);
end
for i = 1:5
    dom = sprintf('NRPS_T_%d.fasta',i);
    copyfile(dom,blastfiles);
end

%%%size of NRPS domain reference
NRPSdomnum1 = 3;
NRPSdomnum2 = 5;
NRPS = strings(NRPSdomnum1,NRPSdomnum2);

for i = 1:5
    dom = sprintf('NRPS_A_%d.fasta',i);
    NRPS(1,i) = dom;
end
for i = 1:5
    dom = sprintf('NRPS_C_%d.fasta',i);
    NRPS(2,i) = dom;
end
for i = 1:5
    dom = sprintf('NRPS_T_%d.fasta',i);
    NRPS(3,i) = dom;
end

%%copy secondmet PKS domain files
for i = 1:5
    dom = sprintf('PKS_KS_%d.fasta',i);
    copyfile(dom,blastfiles);
end
for i = 1:5
    dom = sprintf('PKS_AT_%d.fasta',i);
    copyfile(dom,blastfiles);
```

```
end
for i = 1:5
    dom = sprintf('PKS_ACP_%d.fasta',i);
    copyfile(dom,blastfiles);
end

%%%size of PKS domain reference
PKSdomnum1 = 3;
PKSdomnum2 = 5;
PKS = strings(PKSdomnum1,PKSdomnum2);

for i = 1:5
    dom = sprintf('PKS_KS_%d.fasta',i);
    PKS(1,i) = dom;
end
for i = 1:5
    dom = sprintf('PKS_AT_%d.fasta',i);
    PKS(2,i) = dom;
end
for i = 1:5
    dom = sprintf('PKS_ACP_%d.fasta',i);
    PKS(3,i) = dom;
end

%%%%%%%%%%copying terpene files
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
fastafiles = 29;
key = strings(length(fastafiles),3);
terpenecount = 0;

cd diterpene_1
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)

   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for a=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(a).name);
    newnamefasta = sprintf('diterpene_%d.fasta',a);
    fastaname = struct2cell(fastafiles(a));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(a,2) = namefasta;
    key(a,3) = 'diterpene';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end
```

```
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
cd meroterpenoid_2
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)

   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for b=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(b).name);
    newnamefasta = sprintf('meroterpenoid_%d.fasta',b);
    fastaname = struct2cell(fastafiles(b));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(b+a,2) = namefasta;
    key(b+a,3) = 'meroterpenoid';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end

cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
cd monoterpene_3
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)

   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for c=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(c).name);
    newnamefasta = sprintf('monoterpene_%d.fasta',c);
    fastaname = struct2cell(fastafiles(c));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(a+b+c,2) = namefasta;
    key(a+b+c,3) = 'monoterpene';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end

cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
cd sesquiterpernoid_4
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)
```

```
   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for d=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(d).name);
    newnamefasta = sprintf('sesquiterpenoid_%d.fasta',d);
    fastaname = struct2cell(fastafiles(d));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(a+b+c+d,2) = namefasta;
    key(a+b+c+d,3) = 'sesquiterpenoid';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end

cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
cd triterpenoid_5
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)

   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for e=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(e).name);
    newnamefasta = sprintf('triterpenoid_%d.fasta',e);
    fastaname = struct2cell(fastafiles(e));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(a+b+c+d+e,2) = namefasta;
    key(a+b+c+d+e,3) = 'triterpenoid';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end

cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
cd other_6
terpenefiles = dir('*.txt');
for i=1:length(terpenefiles)

   [~,z]= fileparts(terpenefiles(i).name);
   newnameterpene = strcat(z,'.fasta');
```

```
   terpenename = struct2cell(terpenefiles(i));
   nameterpene = terpenename{1};

   copyfile(nameterpene,newnameterpene);
end
fastafiles = dir('*.fasta');
for f=1:length(fastafiles)
    [~,fp]=fileparts(fastafiles(f).name);
    newnamefasta = sprintf('terpene_%d.fasta',f);
    fastaname = struct2cell(fastafiles(f));
    namefasta = fastaname{1};
    namefasta = convertCharsToStrings(namefasta);
    key(a+b+c+d+e+f,2) = namefasta;
    key(a+b+c+d+e+f,3) = 'terpene';
    terpenecount = terpenecount + 1;
    copyfile(namefasta,newnamefasta);
    movefile(newnamefasta,blastfiles);
end



for i=1:terpenecount
    key(i,1) = i;
end
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\terpenes
file1 = sprintf('terpene_key');
xlswrite(file1, key);

%%%size of terpene types reference
terpene1 = 7;
terpene2 = 4;
terpene3 = 1;
terpene4 = 11;
terpene5 = 1;
terpene6 = 5;
terpene = struct('type',[]);
type = struct('value',[]);

for i = 1:terpene1
    dom = sprintf('diterpene_%d.fasta',i);
    terpene(1).type(i).value = dom;
    terpene(1).name = 'diterpene';
end
for i = 1:terpene2
    dom = sprintf('meroterpenoid_%d.fasta',i);
    terpene(2).type(i).value = dom;
    terpene(2).name = 'meroterpenoid';
end
for i = 1:terpene3
    dom = sprintf('monoterpene_%d.fasta',i);
    terpene(3).type(i).value = dom;
    terpene(3).name = 'monoterpened';
end
for i = 1:terpene4
    dom = sprintf('sesquiterpenoid_%d.fasta',i);
    terpene(4).type(i).value = dom;
```

```
    terpene(4).name = 'sesquiterpenoid';
end
for i = 1:terpene5
    dom = sprintf('triterpenoid_%d.fasta',i);
    terpene(5).type(i).value = dom;
    terpene(5).name = 'triterpenoid';
end
for i = 1:terpene6
    dom = sprintf('terpene_%d.fasta',i);
    terpene(6).type(i).value = dom;
    terpene(6).name = 'terpene';
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%BLASTING%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%[x,~]= size(targetidx);
x = numgenomes;

secondmet = struct('NRPS',[],'PKS',[],'terpene',[]); % 6 is number of second
metabolite genes tested

%%blasting NRPS domains
cd(directory);
cd(resultsfolder);
for a = 1:x
    z=1;
    for b =1:size(NRPS,1)
        for c = 1:size(NRPS,2)
% genomenum = targetidx(a,2);
genomenum = a;
query = NRPS(b,c);
query = convertStringsToChars(query);
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(NRPS(b,c),'.'),outA);
outname = strcat(strtok(NRPS(b,c),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
secondmet.NRPS(a,z).gen_tar = outname1;
secondmet.NRPS(a,z).genome = genomenum; %genome
secondmet.NRPS(a,z).domain = NRPS(b,c); %target
secondmet.NRPS(a,z).output = output;
z=z+1;
        end
   end
end

%%%blasting PKS
```

```
cd(directory);
cd(resultsfolder);
for a = 1:x
    z=1;
    for b =1:size(PKS,1)
        for c = 1:size(PKS,2)
% genomenum = targetidx(a,2);
genomenum = a;
query = PKS(b,c);
query = convertStringsToChars(query);
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(PKS(b,c),'.'),outA);
outname = strcat(strtok(PKS(b,c),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
secondmet.PKS(a,z).gen_tar = outname1;
secondmet.PKS(a,z).genome = genomenum; %genome
secondmet.PKS(a,z).domain = PKS(b,c); %target
secondmet.PKS(a,z).output = output;
z=z+1;
       end
   end
end
%%%blasting terpene
cd(directory);
cd(resultsfolder);
for a = 1:x
    z=1;
    for b =1:length(terpene)
        for c = 1:length(terpene(b).type)
% genomenum = targetidx(a,2);
genomenum = a;
query = terpene(b).type(c).value;
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(terpene(b).type(c).value,'.'),outA);
outname = strcat(strtok(terpene(b).type(c).value,'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
secondmet.terpene(a,z).gen_tar = outname1;
secondmet.terpene(a,z).genome = genomenum; %genome
secondmet.terpene(a,z).type = terpene(b).name; %target
secondmet.terpene(a,z).output = output;
```

```
z=z+1;
       end
   end
end
cd(directory);
cd(resultsfolder);

save('secondmet.mat','secondmet','PKS','NRPS','terpene');
```

Script 4: secondmetcheck_NRPS.m

```
%%%%%%%%%%check for NRPS clusters%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;

% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)
load('secondmet.mat');
idenval = 10;

cd(directory);
cd(resultsfolder)


genomenum = size(secondmet.NRPS,1);
NRPShitsidx = struct('output',cell(genomenum,size(NRPS,1)));
output                                                                  =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);

%%%count number hits of NRPShits%%%%

%%reduce hits by identity%%
%count A domain and index
for a = 1:length(secondmet.NRPS(:,1))
    hitcount = 1;
    for b = 1:(length(secondmet.NRPS(1,:))/3)
        for c = 1:length(secondmet.NRPS(a,b).output.Hits)
            for d = 1:length(secondmet.NRPS(a,b).output.Hits(c).HSPs)
                if
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                    NRPShitsidx(a,1).output(hitcount).scaffold                =
secondmet.NRPS(a,b).output.Hits(c).Name;
                    NRPShitsidx(a,1).output(hitcount).genome                  =
secondmet.NRPS(a,b).genome;
                    NRPShitsidx(a,1).output(hitcount).domain                  =
secondmet.NRPS(a,b).domain;
                    NRPShitsidx(a,1).output(hitcount).indexleft               =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    NRPShitsidx(a,1).output(hitcount).indexright              =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
```

```
            end
        end
    end
end
%count C domain and index
for a = 1:length(secondmet.NRPS(:,1))
    hitcount = 1;
    for b = 1+size(NRPS,2):size(NRPS,2)+(length(secondmet.NRPS(1,:))/3)
        for c = 1:length(secondmet.NRPS(a,b).output.Hits)
            for d = 1:length(secondmet.NRPS(a,b).output.Hits(c).HSPs)
                if
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                    NRPShitsidx(a,2).output(hitcount).scaffold               =
secondmet.NRPS(a,b).output.Hits(c).Name;
                    NRPShitsidx(a,2).output(hitcount).genome                 =
secondmet.NRPS(a,b).genome;
                    NRPShitsidx(a,2).output(hitcount).domain                 =
secondmet.NRPS(a,b).domain;
                    NRPShitsidx(a,2).output(hitcount).indexleft              =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    NRPShitsidx(a,2).output(hitcount).indexright             =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end
%count T domain and index
hitcount = 1;
for a = 1:length(secondmet.NRPS(:,1)) %%iterate through genomes
    hitcount = 1;
    for                                    b                                 =
1+size(NRPS,2)+size(NRPS,2):size(NRPS,2)+size(NRPS,2)+(length(secondmet.NRPS(
1,:))/3) %%iterate through domains
        for c = 1:length(secondmet.NRPS(a,b).output.Hits) %%iterate through
genome hits
            for    d    =    1:length(secondmet.NRPS(a,b).output.Hits(c).HSPs)
%%iterate through scaffold hits
                if
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                    NRPShitsidx(a,3).output(hitcount).scaffold               =
secondmet.NRPS(a,b).output.Hits(c).Name;
                    NRPShitsidx(a,3).output(hitcount).genome                 =
secondmet.NRPS(a,b).genome;
                    NRPShitsidx(a,3).output(hitcount).domain                 =
secondmet.NRPS(a,b).domain;
                    NRPShitsidx(a,3).output(hitcount).indexleft              =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    NRPShitsidx(a,3).output(hitcount).indexright             =
secondmet.NRPS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end
```

```
%%%%reduce hits by checking colocalization of domains%%
NRPScluster = struct('output',[]);
output = struct('scaffold',[],'genome',[],'indexright',[],'indexleft',[]);
check1 = length(NRPShitsidx);
clusterdist = 10000;

for a = 1:check1 %% iterate through num of genomes

            numhits1 = length(NRPShitsidx(a,1).output);
            numhits2 = length(NRPShitsidx(a,2).output);
            numhits3 = length(NRPShitsidx(a,3).output);

            clustercount = 1;

        for c = 1:numhits1 %%going through NRPS domain hits A domain

            for d = 1:numhits2 %checks against C domain
                    distcheck1 = clusterdist + 100;
                    distcheck2 = clusterdist + 100;
                    distcheck3 = clusterdist + 100;
                    distcheck4 = clusterdist + 100;
                tf1                                                                 =
strcmp(NRPShitsidx(a,1).output(c).scaffold,NRPShitsidx(a,2).output(d).scaffol
d);
                if tf1 ==1
                    distcheck1  =  abs(NRPShitsidx(a,1).output(c).indexright  -
NRPShitsidx(a,2).output(d).indexright);
                    distcheck2  =  abs(NRPShitsidx(a,1).output(c).indexright  -
NRPShitsidx(a,2).output(d).indexleft);
                    distcheck3  =  abs(NRPShitsidx(a,1).output(c).indexleft  -
NRPShitsidx(a,2).output(d).indexright);
                    distcheck4  =  abs(NRPShitsidx(a,1).output(c).indexleft  -
NRPShitsidx(a,2).output(d).indexleft);

                end


                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    tfcheck2 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                    for e = 1:numhits3 %checks against T domain if C and A
are together
                        distcheck5 = clusterdist + 100;
                        distcheck6 = clusterdist + 100;
                        distcheck7 = clusterdist + 100;
                        distcheck8 = clusterdist + 100;
```

```
                    tf2                                                     =
strcmp(NRPShitsidx(a,1).output(c).scaffold,NRPShitsidx(a,3).output(e).scaffol
d);
                        if tf2 == 1
                        distcheck5                                          =
abs(NRPShitsidx(a,1).output(c).indexright                                   -
NRPShitsidx(a,3).output(e).indexright);
                        distcheck6                                          =
abs(NRPShitsidx(a,1).output(c).indexright                                   -
NRPShitsidx(a,3).output(e).indexleft);
                        distcheck7  =  abs(NRPShitsidx(a,1).output(c).indexleft
- NRPShitsidx(a,3).output(e).indexright);
                        distcheck8  =  abs(NRPShitsidx(a,1).output(c).indexleft
- NRPShitsidx(a,3).output(e).indexleft);
                        end

                    tfcheck2 = 0;
                    boolean2 = [distcheck5,distcheck6,distcheck7,distcheck8];
                    for i=1:length(boolean2)
                        if boolean2(i) <= clusterdist
                            tfcheck2 = 1;
                            break
                        end
                    end
                    if tfcheck2 == 1
                        arrayidx                                            =
[NRPShitsidx(a,1).output(c).indexright,NRPShitsidx(a,1).output(c).indexleft,N
RPShitsidx(a,2).output(d).indexright,NRPShitsidx(a,2).output(d).indexleft,NRP
Shitsidx(a,3).output(e).indexleft,NRPShitsidx(a,3).output(e).indexright];
                        arrayidx = sort(arrayidx);
                        NRPScluster(a).output(clustercount).scaffold        =
NRPShitsidx(a,1).output(c).scaffold;
                        NRPScluster(a).output(clustercount).genome = a;
                        NRPScluster(a).output(clustercount).indexright      =
arrayidx(6);
                        NRPScluster(a).output(clustercount).indexleft       =
arrayidx(1);
                        clustercount = clustercount + 1;
                        break
                    end
                    end

                end
                if tfcheck2 ==1
                    break;
                end

            end
        end
end

%%%%sort NRPScluster struct to eliminate duplicates

NRPSclustersort = struct('output',[]);
for i = 1:length(NRPScluster)
if isempty(NRPScluster(i).output)
```

```
    continue
end
NRPSfields = fieldnames(NRPScluster(i).output);
outputsort = struct2cell(NRPScluster(i).output);
sz = size(outputsort);
outputsort = reshape(outputsort, sz(1), []);
outputsort = outputsort';
NRPSfields = NRPSfields';
outputsort = sortrows(outputsort, [1 4 3]);
outputsort = cell2struct(outputsort, NRPSfields, 2);
NRPSclustersort(i).output = outputsort;

end

%%%%eliminate duplicates
NRPSclean = struct('output',[]);
distcheck = 10000;

for i = 1:length(NRPSclustersort)
    b = 1;
    hitslen = length(NRPSclustersort(i).output)-1;
        if length(NRPSclustersort(i).output) == 1
            hitslen = 1;
        end
  for a = 1:hitslen

     if a == 1
         NRPSclean(i).output(b).scaffold                                    =
NRPSclustersort(i).output(a).scaffold;
         NRPSclean(i).output(b).genome = NRPSclustersort(i).output(a).genome;
         NRPSclean(i).output(b).indexright                                  =
NRPSclustersort(i).output(a).indexright;
         NRPSclean(i).output(b).indexleft                                   =
NRPSclustersort(i).output(a).indexleft;
     end
%                                       NRPSclean(i).output(b).scaffold      =
NRPSclustersort(i).output(a).scaffold;
%      NRPSclean(i).output(b).genome = NRPSclustersort(i).output(a).genome;
     distcheck9 = clusterdist + 100;
     distcheck10 = clusterdist + 100;
     distcheck11 = clusterdist + 100;
     distcheck12 = clusterdist + 100;
     tf3                                                                    =
strcmp(NRPSclustersort(i).output(a).scaffold,NRPSclustersort(i).output(a+1).s
caffold);
          if tf3 == 1
          distcheck9    =    abs(NRPSclustersort(i).output(a).indexright    -
NRPSclustersort(i).output(a+1).indexright);
          distcheck10    =    abs(NRPSclustersort(i).output(a).indexright    -
NRPSclustersort(i).output(a+1).indexleft);
          distcheck11    =    abs(NRPSclustersort(i).output(a).indexleft    -
NRPSclustersort(i).output(a+1).indexright);
          distcheck12    =    abs(NRPSclustersort(i).output(a).indexleft    -
NRPSclustersort(i).output(a+1).indexleft);
          end
```

```
                    tfcheck3 = 0;
                    boolean3                                                =
[distcheck9,distcheck10,distcheck11,distcheck12];
                    for x=1:length(boolean3)
                        if boolean3(x) <= distcheck
                            tfcheck3 = 1;
                            break
                        end
                    end
%                     if tfcheck3 == 1
%                                                                 arrayidx   =
[NRPSclustersort(i).output(a).indexright,NRPSclustersort(i).output(a).indexle
ft,NRPSclustersort(i).output(a+1).indexright,NRPSclustersort(i).output(a+1).i
ndexleft];
%                         arrayidx = sort(arrayidx);
%                         NRPSclean(i).output(b).indexright = arrayidx(4);
%                         NRPSclean(i).output(b).indexleft = arrayidx(1);
%                     end

                    if tf3 ~= 1 || tfcheck3 ~= 1
                        b = b+1;
                        NRPSclean(i).output(b).scaffold                     =
NRPSclustersort(i).output(a+1).scaffold;
                        NRPSclean(i).output(b).genome                       =
NRPSclustersort(i).output(a+1).genome;
                        NRPSclean(i).output(b).indexright                   =
NRPSclustersort(i).output(a+1).indexright;
                        NRPSclean(i).output(b).indexleft                    =
NRPSclustersort(i).output(a+1).indexleft;
                    end
                    if tf3 ==1 && tfcheck3 ==1 && a~=1
                        arrayidx                                            =
[NRPSclustersort(i).output(a).indexright,NRPSclustersort(i).output(a).indexle
ft,NRPSclustersort(i).output(a+1).indexright,NRPSclustersort(i).output(a+1).i
ndexleft];
                        arrayidx = sort(arrayidx);
                        NRPSclean(i).output(b).indexright = arrayidx(4);
                        NRPSclean(i).output(b).indexleft = arrayidx(1);
                    end
    end

end

cd(directory);
cd(resultsfolder);
save('NRPSresults.mat','NRPSclean');
```

Script 5: secondmetcheck_terpene.m

```
%%%%%%%%%%check for terpene clusters%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
```

```
cd(resultsfolder)

load('secondmet.mat');
idenval = 30;

cd(directory);
cd(resultsfolder)


genomenum = size(secondmet.terpene,1);
terpenecluster = struct('output',cell(genomenum,size(terpene,1)));
output                                                                  =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[],'
type',[]);

%%%count number hits of terpenehits%%%%

%%reduce hits by identity%%
%count terpene hits and index
for a = 1:length(secondmet.terpene(:,1))
    hitcount = 1;
    for b = 1:(length(secondmet.terpene(1,:)))
        for c = 1:length(secondmet.terpene(a,b).output.Hits)
            for d = 1:length(secondmet.terpene(a,b).output.Hits(c).HSPs)
                if
secondmet.terpene(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                terpenecluster(a,1).output(hitcount).scaffold           =
secondmet.terpene(a,b).output.Hits(c).Name;
                terpenecluster(a,1).output(hitcount).genome             =
secondmet.terpene(a,b).genome;
                terpenecluster(a,1).output(hitcount).indexleft          =
secondmet.terpene(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                terpenecluster(a,1).output(hitcount).indexright         =
secondmet.terpene(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                terpenecluster(a,1).output(hitcount).type               =
secondmet.terpene(a,b).type;
                hitcount = hitcount + 1;
                end
            end
        end
    end
end


%%%%sort terpenecluster struct to eliminate duplicates

terpeneclustersort = struct('output',[]);
for i = 1:length(terpenecluster)
if isempty(terpenecluster(i).output)
    continue
end
terpenefields = fieldnames(terpenecluster(i).output);
outputsort = struct2cell(terpenecluster(i).output);
sz = size(outputsort);
outputsort = reshape(outputsort, sz(1), []);
```

```
outputsort = outputsort';
terpenefields = terpenefields';
outputsort = sortrows(outputsort, [1 4 3]);
outputsort = cell2struct(outputsort, terpenefields, 2);
terpeneclustersort(i).output = outputsort;

end

%%%%eliminate duplicates
terpeneclean = struct('output',[]);
distcheck = 10000;
clusterdist = 10000;

for i = 1:length(terpeneclustersort)
    b = 1;
    hitslen = length(terpeneclustersort(i).output)-1;
         if length(terpeneclustersort(i).output) == 1
             hitslen = 1;
         end
  for a = 1:hitslen

     if a == 1
         terpeneclean(i).output(b).scaffold                                    =
terpeneclustersort(i).output(a).scaffold;
         terpeneclean(i).output(b).genome                                      =
terpeneclustersort(i).output(a).genome;
         terpeneclean(i).output(b).indexright                                  =
terpeneclustersort(i).output(a).indexright;
         terpeneclean(i).output(b).indexleft                                   =
terpeneclustersort(i).output(a).indexleft;
         terpeneclean(i).output(b).indexleft                                   =
terpeneclustersort(i).output(a).indexleft;
         terpeneclean(i).output(b).type                                        =
terpeneclustersort(i).output(a).type;
     end
%                                    terpeneclean(i).output(b).scaffold      =
terpeneclustersort(i).output(a).scaffold;
%                                      terpeneclean(i).output(b).genome      =
terpeneclustersort(i).output(a).genome;
     distcheck9 = clusterdist + 100;
     distcheck10 = clusterdist + 100;
     distcheck11 = clusterdist + 100;
     distcheck12 = clusterdist + 100;
     tf3                                                                     =
strcmp(terpeneclustersort(i).output(a).scaffold,terpeneclustersort(i).output(
a+1).scaffold);
          if tf3 == 1
          distcheck9   =   abs(terpeneclustersort(i).output(a).indexright   -
terpeneclustersort(i).output(a+1).indexright);
          distcheck10   =   abs(terpeneclustersort(i).output(a).indexright   -
terpeneclustersort(i).output(a+1).indexleft);
          distcheck11   =   abs(terpeneclustersort(i).output(a).indexleft   -
terpeneclustersort(i).output(a+1).indexright);
          distcheck12   =   abs(terpeneclustersort(i).output(a).indexleft   -
terpeneclustersort(i).output(a+1).indexleft);
          end
```

```
                    tfcheck3 = 0;
                    boolean3                                                =
[distcheck9,distcheck10,distcheck11,distcheck12];
                    for x=1:length(boolean3)
                        if boolean3(x) <= distcheck
                            tfcheck3 = 1;
                            break
                        end
                    end
%                     if tfcheck3 == 1
%                                                                 arrayidx   =
[terpeneclustersort(i).output(a).indexright,terpeneclustersort(i).output(a).i
ndexleft,terpeneclustersort(i).output(a+1).indexright,terpeneclustersort(i).o
utput(a+1).indexleft];
%                         arrayidx = sort(arrayidx);
%                         terpeneclean(i).output(b).indexright = arrayidx(4);
%                         terpeneclean(i).output(b).indexleft = arrayidx(1);
%                       end

                    if tf3 ~= 1 || tfcheck3 ~= 1
                        b = b+1;
                        terpeneclean(i).output(b).scaffold                  =
terpeneclustersort(i).output(a+1).scaffold;
                        terpeneclean(i).output(b).genome                    =
terpeneclustersort(i).output(a+1).genome;
                        terpeneclean(i).output(b).indexright                =
terpeneclustersort(i).output(a+1).indexright;
                        terpeneclean(i).output(b).indexleft                 =
terpeneclustersort(i).output(a+1).indexleft;
                        terpeneclean(i).output(b).indexleft                 =
terpeneclustersort(i).output(a+1).indexleft;
                        terpeneclean(i).output(b).type                      =
terpeneclustersort(i).output(a).type;
                    end
                    if tf3 ==1 && tfcheck3 ==1 && a~=1
                        arrayidx                                            =
[terpeneclustersort(i).output(a).indexright,terpeneclustersort(i).output(a).i
ndexleft,terpeneclustersort(i).output(a+1).indexright,terpeneclustersort(i).o
utput(a+1).indexleft];
                        arrayidx = sort(arrayidx);
                        terpeneclean(i).output(b).indexright = arrayidx(4);
                        terpeneclean(i).output(b).indexleft = arrayidx(1);
                    end
    end

end

cd(directory);
cd(resultsfolder);
save('terpeneresults.mat','terpeneclean');
```

Script 6: secondmetcheck_PKS_test.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%check for PKS clusters%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
```

```
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)


load('secondmet.mat');

cd(directory);
cd(resultsfolder)

idenval = 5;
genomenum = size(secondmet.PKS,1);
PKShitsidx = struct('output',cell(genomenum,size(PKS,1)));
output                                                                  =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);

%%%count number hits of NRPShits%%%%

%%reduce hits by identity%%
%count KS domain and index
for a = 1:length(secondmet.PKS(:,1))
    hitcount = 1;
    for b = 1:(length(secondmet.PKS(1,:))/3)
        for c = 1:length(secondmet.PKS(a,b).output.Hits)
            for d = 1:length(secondmet.PKS(a,b).output.Hits(c).HSPs)
                if
secondmet.PKS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                    PKShitsidx(a,1).output(hitcount).scaffold                =
secondmet.PKS(a,b).output.Hits(c).Name;
                    PKShitsidx(a,1).output(hitcount).genome                  =
secondmet.PKS(a,b).genome;
                    PKShitsidx(a,1).output(hitcount).domain                  =
secondmet.PKS(a,b).domain;
                    PKShitsidx(a,1).output(hitcount).indexleft               =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    PKShitsidx(a,1).output(hitcount).indexright              =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end
%count  AT domain and index
for a = 1:length(secondmet.PKS(:,1))
    hitcount = 1;
    for b = 1+size(PKS,2):size(PKS,2)+(length(secondmet.PKS(1,:))/3)
        for c = 1:length(secondmet.PKS(a,b).output.Hits)
            for d = 1:length(secondmet.PKS(a,b).output.Hits(c).HSPs)
                if
secondmet.PKS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
```

```
                    PKShitsidx(a,2).output(hitcount).scaffold                =
secondmet.PKS(a,b).output.Hits(c).Name;
                    PKShitsidx(a,2).output(hitcount).genome                  =
secondmet.PKS(a,b).genome;
                    PKShitsidx(a,2).output(hitcount).domain                  =
secondmet.PKS(a,b).domain;
                    PKShitsidx(a,2).output(hitcount).indexleft               =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    PKShitsidx(a,2).output(hitcount).indexright              =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end
%count ACP domain and index
hitcount = 1;
for a = 1:length(secondmet.PKS(:,1)) %%iterate through genomes
    hitcount = 1;
    for                                    b                                 =
1+size(PKS,2)+size(PKS,2):size(PKS,2)+size(PKS,2)+(length(secondmet.PKS(1,:))
/3) %%iterate through domains
        for  c  =  1:length(secondmet.PKS(a,b).output.Hits)  %%iterate  through
genome hits
            for    d    =    1:length(secondmet.PKS(a,b).output.Hits(c).HSPs)
%%iterate through scaffold hits
                if
secondmet.PKS(a,b).output.Hits(c).HSPs(d).Identities.Percent >= idenval
                    PKShitsidx(a,3).output(hitcount).scaffold                =
secondmet.PKS(a,b).output.Hits(c).Name;
                    PKShitsidx(a,3).output(hitcount).genome                  =
secondmet.PKS(a,b).genome;
                    PKShitsidx(a,3).output(hitcount).domain                  =
secondmet.PKS(a,b).domain;
                    PKShitsidx(a,3).output(hitcount).indexleft               =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                    PKShitsidx(a,3).output(hitcount).indexright              =
secondmet.PKS(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end

%%%%reduce hits by checking colocalization of domains%%
PKScluster = struct('output',[]);
output = struct('scaffold',[],'genome',[],'indexright',[],'indexleft',[]);
check1 = length(PKShitsidx);
clusterdist = 10000;

for a = 1:check1 %% iterate through num of genomes

            numhits1 = length(PKShitsidx(a,1).output);
            numhits2 = length(PKShitsidx(a,2).output);
```

```
            numhits3 = length(PKShitsidx(a,3).output);

            clustercount = 1;

        for c = 1:numhits1 %%going through PKS domains KS domain

            for d = 1:numhits2 %checks against AT domain
                    distcheck1 = clusterdist + 100;
                    distcheck2 = clusterdist + 100;
                    distcheck3 = clusterdist + 100;
                    distcheck4 = clusterdist + 100;
                tf1                                                    =
strcmp(PKShitsidx(a,1).output(c).scaffold,PKShitsidx(a,2).output(d).scaffold)
;
                if tf1 ==1
                    distcheck1  =  abs(PKShitsidx(a,1).output(c).indexright  -
PKShitsidx(a,2).output(d).indexright);
                    distcheck2  =  abs(PKShitsidx(a,1).output(c).indexright  -
PKShitsidx(a,2).output(d).indexleft);
                    distcheck3  =  abs(PKShitsidx(a,1).output(c).indexleft  -
PKShitsidx(a,2).output(d).indexright);
                    distcheck4  =  abs(PKShitsidx(a,1).output(c).indexleft  -
PKShitsidx(a,2).output(d).indexleft);

                end


                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    tfcheck2 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                        arrayidx                                           =
[PKShitsidx(a,1).output(c).indexright,PKShitsidx(a,1).output(c).indexleft,PKS
hitsidx(a,2).output(d).indexright,PKShitsidx(a,2).output(d).indexleft];
                        arrayidx = sort(arrayidx);
                        PKScluster(a).output(clustercount).scaffold        =
PKShitsidx(a,1).output(c).scaffold;
                        PKScluster(a).output(clustercount).genome = a;
                        PKScluster(a).output(clustercount).indexright      =
arrayidx(4);
                        PKScluster(a).output(clustercount).indexleft       =
arrayidx(1);
                        clustercount = clustercount + 1;
                end

                if tfcheck1 ==1
                    break;
                end
```

```
            end
        end
end

%%%%sort PKScluster struct to eliminate duplicates

PKSclustersort = struct('output',[]);
for i = 1:length(PKScluster)
if isempty(PKScluster(i).output)
    continue
end
PKSfields = fieldnames(PKScluster(i).output);
outputsort = struct2cell(PKScluster(i).output);
sz = size(outputsort);
outputsort = reshape(outputsort, sz(1), []);
outputsort = outputsort';
PKSfields = PKSfields';
outputsort = sortrows(outputsort, [1 4 3]);
outputsort = cell2struct(outputsort, PKSfields, 2);
PKSclustersort(i).output = outputsort;

end

%%%%eliminate duplicates
PKSclean = struct('output',[]);
distcheck = 8000;

for i = 1:length(PKSclustersort)
    b = 1;
    hitslen = length(PKSclustersort(i).output)-1;
         if length(PKSclustersort(i).output) == 1
             hitslen = 1;
         end
  for a = 1:hitslen

     if a == 1
         PKSclean(i).output(b).scaffold                                    =
PKSclustersort(i).output(a).scaffold;
         PKSclean(i).output(b).genome = PKSclustersort(i).output(a).genome;
         PKSclean(i).output(b).indexright                                  =
PKSclustersort(i).output(a).indexright;
         PKSclean(i).output(b).indexleft                                   =
PKSclustersort(i).output(a).indexleft;
     end
     if hitslen == 1
         continue
     end
     distcheck9 = distcheck + 100;
     distcheck10 = distcheck + 100;
     distcheck11 = distcheck + 100;
     distcheck12 = distcheck + 100;
     tf3                                                                   =
strcmp(PKSclustersort(i).output(a).scaffold,PKSclustersort(i).output(a+1).sca
ffold);
           if tf3 == 1
```

```
            distcheck9     =     abs(PKSclustersort(i).output(a).indexright     -
PKSclustersort(i).output(a+1).indexright);
            distcheck10    =     abs(PKSclustersort(i).output(a).indexright     -
PKSclustersort(i).output(a+1).indexleft);
            distcheck11    =     abs(PKSclustersort(i).output(a).indexleft      -
PKSclustersort(i).output(a+1).indexright);
            distcheck12    =     abs(PKSclustersort(i).output(a).indexleft      -
PKSclustersort(i).output(a+1).indexleft);
            end

                    tfcheck3 = 0;
                    boolean3                                                    =
[distcheck9,distcheck10,distcheck11,distcheck12];
                    for x=1:length(boolean3)
                        if boolean3(x) <= distcheck
                            tfcheck3 = 1;
                            break
                        end
                    end
%                     if tfcheck3 == 1
%                                                                 arrayidx   =
[NRPSclustersort(i).output(a).indexright,NRPSclustersort(i).output(a).indexle
ft,NRPSclustersort(i).output(a+1).indexright,NRPSclustersort(i).output(a+1).i
ndexleft];
%                         arrayidx = sort(arrayidx);
%                         NRPSclean(i).output(b).indexright = arrayidx(4);
%                         NRPSclean(i).output(b).indexleft = arrayidx(1);
%                     end

                    if tf3 ~=1 || tfcheck3 ~= 1
                        b = b+1;
                        PKSclean(i).output(b).scaffold                          =
PKSclustersort(i).output(a+1).scaffold;
                        PKSclean(i).output(b).genome                            =
PKSclustersort(i).output(a+1).genome;
                        PKSclean(i).output(b).indexright                        =
PKSclustersort(i).output(a+1).indexright;
                        PKSclean(i).output(b).indexleft                         =
PKSclustersort(i).output(a+1).indexleft;
                    end
                    if tf3 ==1 && tfcheck3 ==1 && a~=1
                        arrayidx                                                =
[PKSclustersort(i).output(a).indexright,PKSclustersort(i).output(a).indexleft
,PKSclustersort(i).output(a+1).indexright,PKSclustersort(i).output(a+1).index
left];
                        arrayidx = sort(arrayidx);
                        PKSclean(i).output(b).indexright = arrayidx(4);
                        PKSclean(i).output(b).indexleft = arrayidx(1);
                    end
    end

end


cd(directory);
cd(resultsfolder);
```

```
save('PKSresults.mat','PKSclean');
```

Script 7: auxgeneblast.m

```
%%%%blast secondary metabolite canonical genes%%%%%%%%%%%%%%%%%%%%
%%%%Colocalization of gene clusters

%                                  blastfiles                                    =
'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes\resultsfolder';
%specify here the location of blast files
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)
blastfiles                                                                    =
strcat('C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2\',results
folder); %specify here the location of blast files
gbfiles = dir('*.nhr');
numgenomes = length(gbfiles);

cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
cd Auxgenes_2nd_Met

%%copy TF blast files
for i = 1:10
    dom = sprintf('TF_%d.fasta',i);
    copyfile(dom,blastfiles);
end
for i = 1:10
    dom = sprintf('ZNbind_%d.fasta',i);
    copyfile(dom,blastfiles);
end


%%%size of TF domain reference
TFnum1 = 2;
TFnum2 = 10;
TF = strings(TFnum1,TFnum2);

for i = 1:10
    dom = sprintf('TF_%d.fasta',i);
    TF(1,i) = dom;
end
for i = 1:10
    dom = sprintf('ZNbind_%d.fasta',i);
    TF(2,i) = dom;
end

auxgenes  =  struct('TF',[],'P450',[],'FMO',[],'MT',[],'TE',[]);  %  categories
of secondary met genes checked for
```

```
%%blasting secondmet genes
cd(directory);
cd(resultsfolder);
for a = 1:numgenomes
    z=1;
    for b =1:size(TF,1)
        for c = 1:size(TF,2)
% genomenum = targetidx(a,2);
genomenum = a;
query = TF(b,c);
query = convertStringsToChars(query);
outA = sprintf('_hitsin_%d',genomenum);
outnameTF = strcat(strtok(TF(b,c),'.'),outA);
outname = strcat(strtok(TF(b,c),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
outputTF = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
auxgenes.TF(a,z).gen_tar = outnameTF;
auxgenes.TF(a,z).genome = genomenum; %genome
auxgenes.TF(a,z).domain = TF(b,c); %gene
auxgenes.TF(a,z).output = outputTF;
z=z+1;
        end
   end
end

%%%%%%%%%%%%%%%%%%%%%%check for P450's%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
cd Auxgenes_2nd_Met

%%copy P450 blast files
for i = 1:10
    dom = sprintf('P450_%d.fasta',i);
    copyfile(dom,blastfiles);
end

%%%size of NRPS domain reference
P450num1 = 1;
P450num2 = 10;
P450 = strings(P450num1,P450num2);

for i = 1:10
    dom = sprintf('P450_%d.fasta',i);
    P450(1,i) = dom;
end

%%blasting secondmet genes
cd(directory);
cd(resultsfolder);
for a = 1:numgenomes
```

```
    z=1;
    for b =1:length(P450)

% genomenum = targetidx(a,2);
genomenum = a;
query = P450(b);
query = convertStringsToChars(query);
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(P450(b),'.'),outA);
outname = strcat(strtok(P450(b),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
auxgenes.P450(a,z).gen_tar = outname1;
auxgenes.P450(a,z).genome = genomenum; %genome
auxgenes.P450(a,z).domain = P450(b); %gene
auxgenes.P450(a,z).output = output;
z=z+1;

   end
end
%%%%%%%%%%%%%%%%%%%%%%check for FMO's%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
cd Auxgenes_2nd_Met

%%copy MT blast files
for i = 1:10
    dom = sprintf('FMO_%d.fasta',i);
    copyfile(dom,blastfiles);
end

%%%size of FMO domain reference
FMOnum1 = 1;
FMOnum2 = 10;
FMO = strings(FMOnum1,FMOnum2);

for i = 1:10
    dom = sprintf('FMO_%d.fasta',i);
    FMO(1,i) = dom;
end

%%blasting secondmet genes
cd(directory);
cd(resultsfolder);
for a = 1:numgenomes
    z=1;
    for b =1:length(FMO)

% genomenum = targetidx(a,2);
```

```
genomenum = a;
query = FMO(b);
query = convertStringsToChars(query);
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(FMO(b),'.'),outA);
outname = strcat(strtok(FMO(b),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output = blastlocal('InputQuery', query ,'Database', dbname
,'Program',%'tblastn'); %if text files is not wanted
auxgenes.FMO(a,z).gen_tar = outname1;
auxgenes.FMO(a,z).genome = genomenum; %genome
auxgenes.FMO(a,z).domain = FMO(b); %gene
auxgenes.FMO(a,z).output = output;
z=z+1;

   end
end

%%%%%%%%%%%%%%%%%%%%%%check for FMO's%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab
cd Auxgenes_2nd_Met

%%copy MT blast files
for i = 1:10
    dom = sprintf('MT_%d.fasta',i);
    copyfile(dom,blastfiles);
end

%%%size of MT domain reference
MTnum1 = 1;
MTnum2 = 10;
MT = strings(MTnum1,MTnum2);

for i = 1:10
    dom = sprintf('MT_%d.fasta',i);
    MT(1,i) = dom;
end

%%blasting secondmet genes
cd(directory);
cd(resultsfolder);
for a = 1:numgenomes
    z=1;
    for b =1:length(MT)

% genomenum = targetidx(a,2);
genomenum = a;
query = MT(b);
query = convertStringsToChars(query);
```

```
outA = sprintf('_hitsin_%d',genomenum);
outname1 = strcat(strtok(MT(b),'.'),outA);
outname = strcat(strtok(MT(b),'.'),outA,'.txt');
outname = convertStringsToChars(outname);
dbname = sprintf('%d.fasta',genomenum);

%output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname);
output = blastlocal('InputQuery', query ,'Database', dbname ,'Program',
'tblastn','ToFile',outname,'expect',50);
%output    =    blastlocal('InputQuery',    query    ,'Database',    dbname
,'Program',%'tblastn'); %if text files is not wanted
auxgenes.MT(a,z).gen_tar = outname1;
auxgenes.MT(a,z).genome = genomenum; %genome
auxgenes.MT(a,z).domain = MT(b); %gene
auxgenes.MT(a,z).output = output;
z=z+1;

   end
end


cd(directory);
cd(resultsfolder);
save('auxgenesblast.mat','auxgenes','TF','P450','FMO','MT');
```

Script 8: auxgeneblast.m

```
%%%%%store auxgeneblast data%%%%%%%%%%
clear;
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';

% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)

load('auxgenesblast.mat');


%%%%%%%%%%%count number hits of TF hits%%%%%%%%%%%%%%%%%%

genomenum = size(auxgenes.TF,1);
TFhitsidx = struct('output',cell(genomenum,size(TF,1)));
output                                                                    =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);


%%reduce hits by identity%%
%%%% scoring paramters for hits
Scoreval = 20;
Lengthval = 0.1;
Eval = 1E-20;
```

```
identval = 15;
%%%%%%%%%%%%%%%%%%%

%count TF domain and index
for a = 1:length(auxgenes.TF(:,1))
    hitcount = 1;
    for b = 1:(length(auxgenes.TF(1,:))/2)
        for c = 1:length(auxgenes.TF(a,b).output.Hits)
            for d = 1:length(auxgenes.TF(a,b).output.Hits(c).HSPs)
                scorenum = 0;
                lengthnum = 0;
                evalnum = 0;
                ident = 0;
                 if
auxgenes.TF(a,b).output.Hits(c).HSPs(d).Identities.Percent >= identval
                     ident = 1;
                 end
                 cover                                                  =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).QueryIndices;
                 coverval = cover(2)-cover(1);
                 if                          coverval                   >
Lengthval*auxgenes.TF(a,b).output.Hits(c).Length
                     lengthnum = 1;
                 end
                 if auxgenes.TF(a,b).output.Hits(c).HSPs(d).Expect < Eval
                     evalnum = 1;
                 end
                 if ident == 1
                    TFhitsidx(a,1).output(hitcount).scaffold            =
auxgenes.TF(a,b).output.Hits(c).Name;
                     TFhitsidx(a,1).output(hitcount).genome             =
auxgenes.TF(a,b).genome;
                     TFhitsidx(a,1).output(hitcount).domain             =
auxgenes.TF(a,b).domain;
                     TFhitsidx(a,1).output(hitcount).indexleft          =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                     TFhitsidx(a,1).output(hitcount).indexright         =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                     hitcount = hitcount + 1;
                end
            end
        end
    end
end
%count ZNbind domain and index
for a = 1:length(auxgenes.TF(:,1))
    hitcount = 1;
    for b = 1+size(TF,2):size(TF,2)+(length(auxgenes.TF(1,:))/2)
        for c = 1:length(auxgenes.TF(a,b).output.Hits)
            for d = 1:length(auxgenes.TF(a,b).output.Hits(c).HSPs)
                scorenum = 0;
                lengthnum = 0;
                evalnum = 0;
                ident = 0;
                 if
auxgenes.TF(a,b).output.Hits(c).HSPs(d).Identities.Percent >= identval
                     ident = 1;
```

```
                end
                cover                                                    =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).QueryIndices;
                coverval = cover(2)-cover(1);
                if                          coverval                     >
Lengthval*auxgenes.TF(a,b).output.Hits(c).Length
                    lengthnum = 1;
                end
                if auxgenes.TF(a,b).output.Hits(c).HSPs(d).Expect < Eval
                    evalnum = 1;
                end
                if ident == 1
                   TFhitsidx(a,2).output(hitcount).scaffold              =
auxgenes.TF(a,b).output.Hits(c).Name;
                   TFhitsidx(a,2).output(hitcount).genome                =
auxgenes.TF(a,b).genome;
                   TFhitsidx(a,2).output(hitcount).domain                =
auxgenes.TF(a,b).domain;
                   TFhitsidx(a,2).output(hitcount).indexleft             =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).SubjectIndices(1);
                   TFhitsidx(a,2).output(hitcount).indexright            =
auxgenes.TF(a,b).output.Hits(c).HSPs(d).SubjectIndices(2);
                   hitcount = hitcount + 1;
                end
            end
        end
    end
end

%%%%sort TF struct to eliminate duplicates

TFsort = struct('output',cell(genomenum,size(TF,1)));
for a = 1:size(TFhitsidx,1)
    for b = 1:size(TFhitsidx,2)
        TFfields = fieldnames(TFhitsidx(a,b).output);
        outputsort = struct2cell(TFhitsidx(a,b).output);
        sz = size(outputsort);
        outputsort = reshape(outputsort, sz(1), []);
        outputsort = outputsort';
        TFfields = TFfields';
        outputsort = sortrows(outputsort, [1 5 4]);
        outputsort = cell2struct(outputsort, TFfields, 2);
        TFsort(a,b).output = outputsort;
    end
end

%%%%eliminate duplicates
TFclean = struct('output',cell(genomenum,size(TF,1)));
clusterdist = 3000;

for a = 1:size(TFsort,1)
  for b = 1:size(TFsort,2)
    count = 1;
    hitslen = length(TFsort(a,b).output)-1;
         if length(TFsort(a,b).output) == 1
             hitslen = 1;
```

```
        end
    for c = 1:hitslen

        if c == 1
            TFclean(a,b).output(count).scaffold                                    =
TFsort(a,b).output(c).scaffold;
            TFclean(a,b).output(count).genome                                      =
TFsort(a,b).output(c).genome;
            TFclean(a,b).output(count).indexright                                  =
TFsort(a,b).output(c).indexright;
            TFclean(a,b).output(count).indexleft                                   =
TFsort(a,b).output(c).indexleft;
        end
        distcheck9 = clusterdist + 100;
        distcheck10 = clusterdist + 100;
        distcheck11 = clusterdist + 100;
        distcheck12 = clusterdist + 100;
        tf3                                                                        =
strcmp(TFsort(a,b).output(c).scaffold,TFsort(a,b).output(c+1).scaffold);
            if tf3 == 1
            distcheck9     =     abs(TFsort(a,b).output(c).indexright     -
TFsort(a,b).output(c+1).indexright);
            distcheck10     =     abs(TFsort(a,b).output(c).indexright     -
TFsort(a,b).output(c+1).indexleft);
            distcheck11     =     abs(TFsort(a,b).output(c).indexleft     -
TFsort(a,b).output(c+1).indexright);
            distcheck12     =     abs(TFsort(a,b).output(c).indexleft     -
TFsort(a,b).output(c+1).indexleft);
            end

                    tfcheck3 = 0;
                    boolean3                                                       =
[distcheck9,distcheck10,distcheck11,distcheck12];
                    for x=1:length(boolean3)
                        if boolean3(x) <= clusterdist
                            tfcheck3 = 1;
                            break
                        end
                    end

                    if tf3 ~= 1 || tfcheck3 ~= 1
                        count = count+1;
                        TFclean(a,b).output(count).scaffold                        =
TFsort(a,b).output(c+1).scaffold;
                        TFclean(a,b).output(count).genome                          =
TFsort(a,b).output(c+1).genome;
                        TFclean(a,b).output(count).indexright                      =
TFsort(a,b).output(c+1).indexright;
                        TFclean(a,b).output(count).indexleft                       =
TFsort(a,b).output(c+1).indexleft;
                    end
                    if tf3 ==1 && tfcheck3 ==1 && c~=1
                        arrayidx                                                   =
[TFsort(a,b).output(c).indexright,TFsort(a,b).output(c).indexleft,TFsort(a,b)
.output(c+1).indexright,TFsort(a,b).output(c+1).indexleft];
                        arrayidx = sort(arrayidx);
```

```
                            TFclean(a,b).output(count).indexright            =
arrayidx(4);
                            TFclean(a,b).output(count).indexleft             =
arrayidx(1);
                        end
    end
  end
end

%%%%%%%%%%%%count number hits of P450 hits%%%%%%%%%%%%%%%%%%
genomenum = size(auxgenes.P450,1);
P450hitsidx = struct('output',cell(genomenum,size(P450,1)));
output                                                                       =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);


%%reduce hits by identity%%
%%%% scoring paramters for hits
Scoreval = 100;
Lengthval = 0.1;
Eval = 1E-20;
identval = 15;
%%%%%%%%%%%%%%%%%%%%
%count P450 domain and index
for a = 1:length(auxgenes.P450(:,1))
    hitcount = 1;
    for b = 1:(length(auxgenes.P450(1,:))/2)
        for c = 1:length(auxgenes.P450(a).output.Hits)
            for d = 1:length(auxgenes.P450(a).output.Hits(c).HSPs)
                scorenum = 0;
                lengthnum = 0;
                evalnum = 0;
                ident = 0;
                 if
auxgenes.P450(a).output.Hits(c).HSPs(d).Identities.Percent >= identval
                     ident = 1;
                 end
                 cover                                                       =
auxgenes.P450(a).output.Hits(c).HSPs(d).QueryIndices;
                 coverval = cover(2)-cover(1);
                 if                          coverval                        >
Lengthval*auxgenes.P450(a).output.Hits(c).Length
                     lengthnum = 1;
                 end
                 if auxgenes.P450(a).output.Hits(c).HSPs(d).Expect < Eval
                     evalnum = 1;
                 end
                 if ident == 1
                    P450hitsidx(a).output(hitcount).scaffold                 =
auxgenes.P450(a).output.Hits(c).Name;
                    P450hitsidx(a).output(hitcount).genome                   =
auxgenes.P450(a).genome;
                    P450hitsidx(a).output(hitcount).domain                   =
auxgenes.P450(a).domain;
                    P450hitsidx(a).output(hitcount).indexleft                =
auxgenes.P450(a).output.Hits(c).HSPs(d).SubjectIndices(1);
```

```
                    P450hitsidx(a).output(hitcount).indexright                =
auxgenes.P450(a).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end


%%%%sort P450 struct to eliminate duplicates

P450sort = struct('output',cell(genomenum,size(P450,1)));
for a = 1:size(P450hitsidx,1)

        P450fields = fieldnames(P450hitsidx(a).output);
        outputsort = struct2cell(P450hitsidx(a).output);
        sz = size(outputsort);
        outputsort = reshape(outputsort, sz(1), []);
        outputsort = outputsort';
        P450fields = P450fields';
        outputsort = sortrows(outputsort, [1 5 4]);
        outputsort = cell2struct(outputsort, P450fields, 2);
        P450sort(a).output = outputsort;

end

%%%%eliminate duplicates
P450clean = struct('output',cell(genomenum,size(P450,1)));
clusterdist = 3000;

for a = 1:size(P450sort,1)

    count = 1;
    hitslen = length(P450sort(a).output)-1;
         if length(P450sort(a).output) == 1
             hitslen = 1;
         end
     for c = 1:hitslen

         if c == 1
             P450clean(a).output(count).scaffold                              =
P450sort(a).output(c).scaffold;
             P450clean(a).output(count).genome                                =
P450sort(a).output(c).genome;
             P450clean(a).output(count).indexright                            =
P450sort(a).output(c).indexright;
             P450clean(a).output(count).indexleft                             =
P450sort(a).output(c).indexleft;
         end
         distcheck9 = clusterdist + 100;
         distcheck10 = clusterdist + 100;
         distcheck11 = clusterdist + 100;
         distcheck12 = clusterdist + 100;
```

```
        P4503                                                                =
strcmp(P450sort(a).output(c).scaffold,P450sort(a).output(c+1).scaffold);
               if P4503 == 1
               distcheck9     =     abs(P450sort(a).output(c).indexright     -
P450sort(a).output(c+1).indexright);
               distcheck10     =     abs(P450sort(a).output(c).indexright     -
P450sort(a).output(c+1).indexleft);
               distcheck11     =     abs(P450sort(a).output(c).indexleft     -
P450sort(a).output(c+1).indexright);
               distcheck12     =     abs(P450sort(a).output(c).indexleft     -
P450sort(a).output(c+1).indexleft);
               end

                        P450check3 = 0;
                        boolean3                                             =
[distcheck9,distcheck10,distcheck11,distcheck12];
                        for x=1:length(boolean3)
                            if boolean3(x) <= clusterdist
                                P450check3 = 1;
                                break
                            end
                        end

                        if P4503 ~= 1 || P450check3 ~= 1
                            count = count+1;
                            P450clean(a).output(count).scaffold              =
P450sort(a).output(c+1).scaffold;
                            P450clean(a).output(count).genome                =
P450sort(a).output(c+1).genome;
                            P450clean(a).output(count).indexright            =
P450sort(a).output(c+1).indexright;
                            P450clean(a).output(count).indexleft             =
P450sort(a).output(c+1).indexleft;
                        end
                        if P4503 ==1 && P450check3 ==1 && c~=1
                            arrayidx                                         =
[P450sort(a).output(c).indexright,P450sort(a).output(c).indexleft,P450sort(a)
.output(c+1).indexright,P450sort(a).output(c+1).indexleft];
                            arrayidx = sort(arrayidx);
                            P450clean(a).output(count).indexright            =
arrayidx(4);
                            P450clean(a).output(count).indexleft             =
arrayidx(1);
                        end
    end

end

%%%%%%%%%%%%count number hits of FMO hits%%%%%%%%%%%%%%%%%%%
genomenum = size(auxgenes.FMO,1);
FMOhitsidx = struct('output',cell(genomenum,size(FMO,1)));
output                                                                       =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);


%%reduce hits by identity%%
```

```
%%%% scoring paramters for hits
Scoreval = 100;
Lengthval = 0.1;
Eval = 1E-20;
identval = 15;
%%%%%%%%%%%%%%%%%%%
%count FMO domain and index
for a = 1:length(auxgenes.FMO(:,1))
    hitcount = 1;
    for b = 1:(length(auxgenes.FMO(1,:))/2)
        for c = 1:length(auxgenes.FMO(a).output.Hits)
            for d = 1:length(auxgenes.FMO(a).output.Hits(c).HSPs)
                scorenum = 0;
                lengthnum = 0;
                evalnum = 0;
                ident = 0;
                 if  auxgenes.FMO(a).output.Hits(c).HSPs(d).Identities.Percent
>= identval
                     ident = 1;
                 end
                 cover = auxgenes.FMO(a).output.Hits(c).HSPs(d).QueryIndices;
                 coverval = cover(2)-cover(1);
                 if                          coverval                        >
Lengthval*auxgenes.FMO(a).output.Hits(c).Length
                     lengthnum = 1;
                 end
                 if auxgenes.FMO(a).output.Hits(c).HSPs(d).Expect < Eval
                     evalnum = 1;
                 end
                 if ident == 1
                    FMOhitsidx(a).output(hitcount).scaffold                  =
auxgenes.FMO(a).output.Hits(c).Name;
                    FMOhitsidx(a).output(hitcount).genome                    =
auxgenes.FMO(a).genome;
                    FMOhitsidx(a).output(hitcount).domain                    =
auxgenes.FMO(a).domain;
                    FMOhitsidx(a).output(hitcount).indexleft                 =
auxgenes.FMO(a).output.Hits(c).HSPs(d).SubjectIndices(1);
                    FMOhitsidx(a).output(hitcount).indexright                =
auxgenes.FMO(a).output.Hits(c).HSPs(d).SubjectIndices(2);
                    hitcount = hitcount + 1;
                end
            end
        end
    end
end


%%%%sort FMO struct to eliminate duplicates

FMOsort = struct('output',cell(genomenum,size(FMO,1)));
for a = 1:size(FMOhitsidx,1)

        FMOfields = fieldnames(FMOhitsidx(a).output);
        outputsort = struct2cell(FMOhitsidx(a).output);
        sz = size(outputsort);
```

```
        outputsort = reshape(outputsort, sz(1), []);
        outputsort = outputsort';
        FMOfields = FMOfields';
        outputsort = sortrows(outputsort, [1 5 4]);
        outputsort = cell2struct(outputsort, FMOfields, 2);
        FMOsort(a).output = outputsort;

end

%%%%eliminate duplicates
FMOclean = struct('output',cell(genomenum,size(FMO,1)));
clusterdist = 3000;

for a = 1:size(FMOsort,1)

    count = 1;
    hitslen = length(FMOsort(a).output)-1;
         if length(FMOsort(a).output) == 1
             hitslen = 1;
         end
     for c = 1:hitslen

         if c == 1
             FMOclean(a).output(count).scaffold                                    =
FMOsort(a).output(c).scaffold;
             FMOclean(a).output(count).genome = FMOsort(a).output(c).genome;
             FMOclean(a).output(count).indexright                                  =
FMOsort(a).output(c).indexright;
             FMOclean(a).output(count).indexleft                                   =
FMOsort(a).output(c).indexleft;
         end
         distcheck9 = clusterdist + 100;
         distcheck10 = clusterdist + 100;
         distcheck11 = clusterdist + 100;
         distcheck12 = clusterdist + 100;
         FMO3                                                                      =
strcmp(FMOsort(a).output(c).scaffold,FMOsort(a).output(c+1).scaffold);
              if FMO3 == 1
              distcheck9     =     abs(FMOsort(a).output(c).indexright     -
FMOsort(a).output(c+1).indexright);
              distcheck10     =     abs(FMOsort(a).output(c).indexright     -
FMOsort(a).output(c+1).indexleft);
              distcheck11     =     abs(FMOsort(a).output(c).indexleft     -
FMOsort(a).output(c+1).indexright);
              distcheck12     =     abs(FMOsort(a).output(c).indexleft     -
FMOsort(a).output(c+1).indexleft);
              end

                      FMOcheck3 = 0;
                      boolean3                                                   =
[distcheck9,distcheck10,distcheck11,distcheck12];
                      for x=1:length(boolean3)
                          if boolean3(x) <= clusterdist
                              FMOcheck3 = 1;
                              break
                          end
```

```
                        end

                        if FMO3 ~= 1 || FMOcheck3 ~= 1
                            count = count+1;
                            FMOclean(a).output(count).scaffold                =
FMOsort(a).output(c+1).scaffold;
                            FMOclean(a).output(count).genome                  =
FMOsort(a).output(c+1).genome;
                            FMOclean(a).output(count).indexright              =
FMOsort(a).output(c+1).indexright;
                            FMOclean(a).output(count).indexleft               =
FMOsort(a).output(c+1).indexleft;
                        end
                        if FMO3 ==1 && FMOcheck3 ==1 && c~=1
                            arrayidx                                          =
[FMOsort(a).output(c).indexright,FMOsort(a).output(c).indexleft,FMOsort(a).ou
tput(c+1).indexright,FMOsort(a).output(c+1).indexleft];
                            arrayidx = sort(arrayidx);
                            FMOclean(a).output(count).indexright              =
arrayidx(4);
                            FMOclean(a).output(count).indexleft               =
arrayidx(1);
                        end
    end

end

%%%%%%%%%%%%count number hits of MT hits%%%%%%%%%%%%%%%%%%%
genomenum = size(auxgenes.MT,1);
MThitsidx = struct('output',cell(genomenum,size(MT,1)));
output                                                                        =
struct('scaffold',[],'genome',[],'domain',[],'indexright',[],'indexleft',[]);


%%reduce hits by identity%%
%%%% scoring paramters for hits
Scoreval = 100;
Lengthval = 0.1;
Eval = 1E-20;
identval = 15;
%%%%%%%%%%%%%%%%%%%%
%count MT domain and index
for a = 1:length(auxgenes.MT(:,1))
    hitcount = 1;
    for b = 1:(length(auxgenes.MT(1,:))/2)
        for c = 1:length(auxgenes.MT(a).output.Hits)
            for d = 1:length(auxgenes.MT(a).output.Hits(c).HSPs)
                scorenum = 0;
                lengthnum = 0;
                evalnum = 0;
                ident = 0;
                 if   auxgenes.MT(a).output.Hits(c).HSPs(d).Identities.Percent
>= identval
                     ident = 1;
                 end
                 cover = auxgenes.MT(a).output.Hits(c).HSPs(d).QueryIndices;
```

```
                coverval = cover(2)-cover(1);
                if coverval > Lengthval*auxgenes.MT(a).output.Hits(c).Length
                    lengthnum = 1;
                end
                if auxgenes.MT(a).output.Hits(c).HSPs(d).Expect < Eval
                    evalnum = 1;
                end
                if ident == 1
                   MThitsidx(a).output(hitcount).scaffold                    =
auxgenes.MT(a).output.Hits(c).Name;
                   MThitsidx(a).output(hitcount).genome                      =
auxgenes.MT(a).genome;
                   MThitsidx(a).output(hitcount).domain                      =
auxgenes.MT(a).domain;
                   MThitsidx(a).output(hitcount).indexleft                   =
auxgenes.MT(a).output.Hits(c).HSPs(d).SubjectIndices(1);
                   MThitsidx(a).output(hitcount).indexright                  =
auxgenes.MT(a).output.Hits(c).HSPs(d).SubjectIndices(2);
                   hitcount = hitcount + 1;
               end
            end
        end
    end
end


%%%%sort MT struct to eliminate duplicates

MTsort = struct('output',cell(genomenum,size(MT,1)));
for a = 1:size(MThitsidx,1)

        MTfields = fieldnames(MThitsidx(a).output);
        outputsort = struct2cell(MThitsidx(a).output);
        sz = size(outputsort);
        outputsort = reshape(outputsort, sz(1), []);
        outputsort = outputsort';
        MTfields = MTfields';
        outputsort = sortrows(outputsort, [1 5 4]);
        outputsort = cell2struct(outputsort, MTfields, 2);
        MTsort(a).output = outputsort;

end

%%%%eliminate duplicates
MTclean = struct('output',cell(genomenum,size(MT,1)));
clusterdist = 3000;

for a = 1:size(MTsort,1)

    count = 1;
    hitslen = length(MTsort(a).output)-1;
         if length(MTsort(a).output) == 1
             hitslen = 1;
         end
     for c = 1:hitslen
```

```
        if c == 1
            MTclean(a).output(count).scaffold                                    =
MTsort(a).output(c).scaffold;
            MTclean(a).output(count).genome = MTsort(a).output(c).genome;
            MTclean(a).output(count).indexright                                  =
MTsort(a).output(c).indexright;
            MTclean(a).output(count).indexleft                                   =
MTsort(a).output(c).indexleft;
        end
        distcheck9 = clusterdist + 100;
        distcheck10 = clusterdist + 100;
        distcheck11 = clusterdist + 100;
        distcheck12 = clusterdist + 100;
        MT3                                                                      =
strcmp(MTsort(a).output(c).scaffold,MTsort(a).output(c+1).scaffold);
            if MT3 == 1
            distcheck9      =      abs(MTsort(a).output(c).indexright      -
MTsort(a).output(c+1).indexright);
            distcheck10     =      abs(MTsort(a).output(c).indexright      -
MTsort(a).output(c+1).indexleft);
            distcheck11     =      abs(MTsort(a).output(c).indexleft       -
MTsort(a).output(c+1).indexright);
            distcheck12     =      abs(MTsort(a).output(c).indexleft       -
MTsort(a).output(c+1).indexleft);
            end

                    MTcheck3 = 0;
                    boolean3                                                     =
[distcheck9,distcheck10,distcheck11,distcheck12];
                    for x=1:length(boolean3)
                        if boolean3(x) <= clusterdist
                            MTcheck3 = 1;
                            break
                        end
                    end

                    if MT3 ~= 1 || MTcheck3 ~= 1
                        count = count+1;
                        MTclean(a).output(count).scaffold                        =
MTsort(a).output(c+1).scaffold;
                        MTclean(a).output(count).genome                          =
MTsort(a).output(c+1).genome;
                        MTclean(a).output(count).indexright                      =
MTsort(a).output(c+1).indexright;
                        MTclean(a).output(count).indexleft                       =
MTsort(a).output(c+1).indexleft;
                    end
                    if MT3 ==1 && MTcheck3 ==1 && c~=1
                        arrayidx                                                 =
[MTsort(a).output(c).indexright,MTsort(a).output(c).indexleft,MTsort(a).outpu
t(c+1).indexright,MTsort(a).output(c+1).indexleft];
                        arrayidx = sort(arrayidx);
                        MTclean(a).output(count).indexright                      =
arrayidx(4);
                        MTclean(a).output(count).indexleft = arrayidx(1);
```

```
                        end
    end

end


cd(directory);
cd(resultsfolder);

save('auxgenesclean.mat','TFclean','P450clean','FMOclean','MTclean');
```

Script 9: auxgenecheck.m

```
%%%%%find secondary metabolite gene clusters next to target hits%%%%%%%%
clear;
% cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes\resultsfolder
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)


load('targethits.mat');
load('NRPSresults.mat');
load('PKSresults.mat');
load('auxgenesclean.mat');
load('genomes.mat');
% cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes\resultsfolder

cd(directory);
cd(resultsfolder);


clustersinfo = struct('target',[],'genome',[],'auxgenes',[],'core',[]);
clustersinfo.target = struct('name',[]);
clustersinfo.genome = struct('name',[]);

clusterdist = 50000; %%%%colocalization of cluster leeway
clusterdist2 = 3000;
%%%%find NRPS clusters near targets%%%%%
clustertargethit = 0;
for a = 1:length(targetfinalhits) %% iterate through num of genomes
        count = 0;
        for b = 1:length(targetfinalhits(a).Hits) %%going through positive
target hits
            %%%%%%%%%%%%%%%%%%%%%%check for NRPS
clusters%%%%%%%%%%%%%%%%%%%%%%
            for c = 1: length(NRPSclean(a).output) %%%check through NRPS
                    distcheck1 = clusterdist + 100;
                    distcheck2 = clusterdist + 100;
                    distcheck3 = clusterdist + 100;
                    distcheck4 = clusterdist + 100;
```

```
                tf1                                                          =
strcmp(targetfinalhits(a).Hits(b).scaffold,NRPSclean(a).output(c).scaffold);
                if tf1 ==1
                    distcheck1  =  abs(targetfinalhits(a).Hits(b).indexright  -
NRPSclean(a).output(c).indexright);
                    distcheck2  =  abs(targetfinalhits(a).Hits(b).indexright  -
NRPSclean(a).output(c).indexleft);
                    distcheck3  =  abs(targetfinalhits(a).Hits(b).indexleft  -
NRPSclean(a).output(c).indexright);
                    distcheck4  =  abs(targetfinalhits(a).Hits(b).indexleft  -
NRPSclean(a).output(c).indexleft);
                end
                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                   clustertargethit = clustertargethit + 1;
                   count = count + 1;
                   %%%%store info in struc
                   clustersinfo(clustertargethit).target.output              =
targetfinalhits(a).Hits(b);
                   clustersinfo(clustertargethit).target.name                =
targetkey(targetfinalhits(a).Hits(b).target,2);
                   clustersinfo(clustertargethit).target.output.output       =
rmfield(clustersinfo(clustertargethit).target.output.output,'Hits');
                   clustersinfo(clustertargethit).target.output.output       =
rmfield(clustersinfo(clustertargethit).target.output.output,'Statistics');
                   clustersinfo(clustertargethit).target.output.HSPs         =
rmfield(clustersinfo(clustertargethit).target.output.HSPs,'Alignment');
                   clustersinfo(clustertargethit).genome                     =
key(targetfinalhits(a).Hits(b).genome,2);
                   clustersinfo(clustertargethit).core.output                =
NRPSclean(a).output(c);
                   clustersinfo(clustertargethit).core.name = 'NRPS';
                   clustersinfo(clustertargethit).hitnumber                  =
clustertargethit;

                   %%%%%%%%%%check for TF genes%%%%%%%%%%%%%%%%%%
                   znnum = 0;
                   tfnum = 0;
                   for   aa   =   1:length(TFclean(a,2).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                 =
strcmp(TFclean(a,2).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                       =
abs(TFclean(a,2).output(aa).indexright - NRPSclean(a).output(c).indexright);
```

```
                            distcheck2                                          =
abs(TFclean(a,2).output(aa).indexright - NRPSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(TFclean(a,2).output(aa).indexleft - NRPSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(TFclean(a,2).output(aa).indexleft - NRPSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            znnum = znnum + 1;

clustersinfo(clustertargethit).auxgenes.TF.Znbinddomain(znnum)              =
TFclean(a,2).output(aa);
                        end
                    end
                        for  bb  =  1:length(TFclean(a,1).output)  %%%check
for TF domain
                                    distcheck1 = clusterdist + 100;
                                    distcheck2 = clusterdist + 100;
                                    distcheck3 = clusterdist + 100;
                                    distcheck4 = clusterdist + 100;
                                tfbb                                            =
strcmp(TFclean(a,1).output(bb).scaffold,NRPSclean(a).output(c).scaffold);
                                    if tfbb ==1
                                    distcheck1                                  =
abs(TFclean(a,1).output(bb).indexright - NRPSclean(a).output(c).indexright);
                                    distcheck2                                  =
abs(TFclean(a,1).output(bb).indexright - NRPSclean(a).output(c).indexleft);
                                    distcheck3                                  =
abs(TFclean(a,1).output(bb).indexleft - NRPSclean(a).output(c).indexright);
                                    distcheck4                                  =
abs(TFclean(a,1).output(bb).indexleft - NRPSclean(a).output(c).indexleft);
                                    end
                                boolean1                                        =
[distcheck1,distcheck2,distcheck3,distcheck4];
                                tfbbcheck1 = 0;
                                    for i=1:length(boolean1)
                                        if boolean1(i) <= clusterdist
                                            tfbbcheck1 = 1;
                                            break
                                        end
                                    end

                            if tfbbcheck1 == 1
                                tfnum = tfnum + 1;

clustersinfo(clustertargethit).auxgenes.TF.TF(tfnum)                        =
TFclean(a,1).output(bb);
                            end
```

```
                    end
                %%%%%%%%%%check for P450 genes%%%%%%%%%%%%%%%%%%%
                p450num = 0;
                for   aa   =   1:length(P450clean(a).output)   %%%check   for
Znbinding domain
                        distcheck1 = clusterdist + 100;
                        distcheck2 = clusterdist + 100;
                        distcheck3 = clusterdist + 100;
                        distcheck4 = clusterdist + 100;
                    tfaa                                                    =
strcmp(P450clean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                        if tfaa ==1
                        distcheck1                                          =
abs(P450clean(a).output(aa).indexright - NRPSclean(a).output(c).indexright);
                        distcheck2                                          =
abs(P450clean(a).output(aa).indexright - NRPSclean(a).output(c).indexleft);
                        distcheck3                                          =
abs(P450clean(a).output(aa).indexleft - NRPSclean(a).output(c).indexright);
                        distcheck4                                          =
abs(P450clean(a).output(aa).indexleft - NRPSclean(a).output(c).indexleft);
                        end
                    boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                    tfaacheck1 = 0;
                        for i=1:length(boolean1)
                            if boolean1(i) <= clusterdist
                                tfaacheck1 = 1;
                                break
                            end
                        end
                    if tfaacheck1 == 1
                        p450num = p450num + 1;

clustersinfo(clustertargethit).auxgenes.P450(p450num)                       =
P450clean(a).output(aa);
                    end
                end
                 %%%%%%%%%%check for FMO genes%%%%%%%%%%%%%%%%%%%
                FMOnum = 0;
                 for   aa   =   1:length(FMOclean(a).output)   %%%check   for
Znbinding domain
                        distcheck1 = clusterdist + 100;
                        distcheck2 = clusterdist + 100;
                        distcheck3 = clusterdist + 100;
                        distcheck4 = clusterdist + 100;
                    tfaa                                                    =
strcmp(FMOclean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                        if tfaa ==1
                        distcheck1                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexright);
                        distcheck2                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexleft);
                        distcheck3  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
                        distcheck4  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                        end
```

```
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            FMOnum = FMOnum + 1;

clustersinfo(clustertargethit).auxgenes.FMO(FMOnum) = FMOclean(a).output(aa);
                        end
                    end
                     %%%%%%%%%%check for MT genes%%%%%%%%%%%%%%%%%%
                   MTnum = 0;
                    for   aa   =   1:length(MTclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(MTclean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexright);
                            distcheck2  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexleft);
                            distcheck3  =  abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
                            distcheck4  =  abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           MTnum = MTnum + 1;
                           clustersinfo(clustertargethit).auxgenes.MT(MTnum)
= MTclean(a).output(aa);
                        end
                    end
                end
             end
            %%%%%%%%%%%%%%%%%%%%%%check for PKS clusters%%%%%%%%%%%%%%%%%%%%%
            for c = 1: length(PKSclean(a).output) %%%check through PKS
                     distcheck1 = clusterdist + 100;
                     distcheck2 = clusterdist + 100;
                     distcheck3 = clusterdist + 100;
```

```
                    distcheck4 = clusterdist + 100;
                tf1                                                         =
strcmp(targetfinalhits(a).Hits(b).scaffold,PKSclean(a).output(c).scaffold);
                if tf1 ==1
                    distcheck1  =  abs(targetfinalhits(a).Hits(b).indexright  -
PKSclean(a).output(c).indexright);
                    distcheck2  =  abs(targetfinalhits(a).Hits(b).indexright  -
PKSclean(a).output(c).indexleft);
                    distcheck3  =  abs(targetfinalhits(a).Hits(b).indexleft  -
PKSclean(a).output(c).indexright);
                    distcheck4  =  abs(targetfinalhits(a).Hits(b).indexleft  -
PKSclean(a).output(c).indexleft);
                end
                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                   clustertargethit = clustertargethit + 1;
                   clustersinfo(clustertargethit).target.output              =
targetfinalhits(a).Hits(b);
                   clustersinfo(clustertargethit).target.name                =
targetkey(targetfinalhits(a).Hits(b).target,2);
                   clustersinfo(clustertargethit).target.output.output       =
rmfield(clustersinfo(clustertargethit).target.output.output,'Hits');
                   clustersinfo(clustertargethit).target.output.output       =
rmfield(clustersinfo(clustertargethit).target.output.output,'Statistics');
                   clustersinfo(clustertargethit).target.output.HSPs         =
rmfield(clustersinfo(clustertargethit).target.output.HSPs,'Alignment');
                   clustersinfo(clustertargethit).genome                     =
key(targetfinalhits(a).Hits(b).genome,2);
                   clustersinfo(clustertargethit).core.output                =
PKSclean(a).output(c);
                   clustersinfo(clustertargethit).core.name = 'PKS';
                   clustersinfo(clustertargethit).hitnumber                  =
clustertargethit;
                   %%%%%%%%%%check for TF genes%%%%%%%%%%%%%%%%%%
                   znnum = 0;
                   tfnum = 0;
                   for   aa   =   1:length(TFclean(a,2).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                 =
strcmp(TFclean(a,2).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                       =
abs(TFclean(a,2).output(aa).indexright - PKSclean(a).output(c).indexright);
                            distcheck2                                       =
abs(TFclean(a,2).output(aa).indexright - PKSclean(a).output(c).indexleft);
```

```
                            distcheck3                                          =
abs(TFclean(a,2).output(aa).indexleft - PKSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(TFclean(a,2).output(aa).indexleft - PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           znnum = znnum + 1;

clustersinfo(clustertargethit).auxgenes.TF.Znbinddomain(znnum)              =
TFclean(a,2).output(aa);
                        end
                    end
                            for  bb  =  1:length(TFclean(a,1).output)  %%%check
for TF domain
                                    distcheck1 = clusterdist + 100;
                                    distcheck2 = clusterdist + 100;
                                    distcheck3 = clusterdist + 100;
                                    distcheck4 = clusterdist + 100;
                                tfbb                                            =
strcmp(TFclean(a,1).output(bb).scaffold,PKSclean(a).output(c).scaffold);
                                    if tfbb ==1
                                    distcheck1                                  =
abs(TFclean(a,1).output(bb).indexright - PKSclean(a).output(c).indexright);
                                    distcheck2                                  =
abs(TFclean(a,1).output(bb).indexright - PKSclean(a).output(c).indexleft);
                                    distcheck3                                  =
abs(TFclean(a,1).output(bb).indexleft - PKSclean(a).output(c).indexright);
                                    distcheck4                                  =
abs(TFclean(a,1).output(bb).indexleft - PKSclean(a).output(c).indexleft);
                                    end
                                boolean1                                        =
[distcheck1,distcheck2,distcheck3,distcheck4];
                                tfbbcheck1 = 0;
                                    for i=1:length(boolean1)
                                        if boolean1(i) <= clusterdist
                                            tfbbcheck1 = 1;
                                            break
                                        end
                                    end

                            if tfbbcheck1 == 1
                            tfnum = tfnum + 1;

clustersinfo(clustertargethit).auxgenes.TF.TF(tfnum)                        =
TFclean(a,1).output(bb);
                            end
                            end
```

```
                    %%%%%%%%%%check for P450 genes%%%%%%%%%%%%%%%%%%%
                   p450num = 0;
                   for   aa   =   1:length(P450clean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                          tfaa                                                  =
strcmp(P450clean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(P450clean(a).output(aa).indexright - PKSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(P450clean(a).output(aa).indexright - PKSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(P450clean(a).output(aa).indexleft - PKSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(P450clean(a).output(aa).indexleft - PKSclean(a).output(c).indexleft);
                            end
                          boolean1                                              =
[distcheck1,distcheck2,distcheck3,distcheck4];
                          tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                          if tfaacheck1 == 1
                             p450num = p450num + 1;

clustersinfo(clustertargethit).auxgenes.P450(p450num)                           =
P450clean(a).output(aa);
                          end
                    end
                      %%%%%%%%%%check for FMO genes%%%%%%%%%%%%%%%%%%%
                   FMOnum = 0;
                    for   aa   =   1:length(FMOclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                          tfaa                                                  =
strcmp(FMOclean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexleft);
                            distcheck3  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
```

```
                            distcheck4  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           FMOnum = FMOnum + 1;

clustersinfo(clustertargethit).auxgenes.FMO(FMOnum) = FMOclean(a).output(aa);
                        end
                    end
                     %%%%%%%%%%check for MT genes%%%%%%%%%%%%%%%%%%
                   MTnum = 0;
                    for   aa   =   1:length(MTclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(MTclean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexright);
                            distcheck2  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexleft);
                            distcheck3  =  abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
                            distcheck4  =  abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           MTnum = MTnum + 1;
                           clustersinfo(clustertargethit).auxgenes.MT(MTnum)
= MTclean(a).output(aa);
                        end
                    end
                end
            end
        end
end
```

```
cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2
cd(resultsfolder)

save('clusterfinalhits.mat','clustersinfo');
```

Script 10: targethithchecks.m

```
%%%comparison of target hits and cluster hits

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Read and store target hit data
%Readblast looks at blast output struct for info
clear;
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
%                                 directory                                    =
'C:\Users\Lab\Desktop\Genome_Mining_Matlab\antifungal_tests\mining';
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)
load('targetresults.mat');
xsize = size(results);
numtargets = xsize(1);
numgenomes = xsize(2);

%%%%make database of hits
targethits = struct('Hits',cell(size(results,2),size(results,1)));
for a =1:numgenomes
    for b =1:numtargets
        count = 0;
        for c = 1:length(results(b,a).output.Hits)
           for d = 1:length(results(b,a).output.Hits(c).HSPs)    %
                    count = count+1;
                    targethits(a,b).Hits(count).output                       =
results(b,a).output;
                    targethits(a,b).Hits(count).HSPs                         =
results(b,a).output.Hits(c).HSPs(d);
                    targethits(a,b).Hits(count).genome = a;
                    gentar = sprintf('target%d_hit_genome%d \n',b,a);
                    targethits(a,b).Hits(count).gen_tar = gentar;
                    targethits(a,b).Hits(count).target = b;
                    targethits(a,b).Hits(count).scaffold                     =
results(b,a).output.Hits(c).Name;
                    targethits(a,b).Hits(count).indexleft                    =
results(b,a).output.Hits(c).HSPs(d).SubjectIndices(1);
                    targethits(a,b).Hits(count).indexright                   =
results(b,a).output.Hits(c).HSPs(d).SubjectIndices(2);
           end
        end

    end
end
```

```
test2 = zeros(size(targethits,1),size(targethits,2));
for a = 1:size(targethits,1)
    for b = 1:size(targethits,2)
        test2(a,b) = length(targethits(a,b).Hits);
    end
end

%%%%sort targethit struct to eliminate duplicates

targetsort = struct('Hits',cell(size(targethits,1),size(targethits,2)));
for a = 1:size(targethits,1)
    for b = 1:size(targethits,2)
        if isempty(targethits(a,b).Hits) == 1
            continue
        end
        targetfields = fieldnames(targethits(a,b).Hits);
        outputsort = struct2cell(targethits(a,b).Hits);
        sz = size(outputsort);
        outputsort = reshape(outputsort, sz(1), []);
        outputsort = outputsort';
        targetfields = targetfields';
        outputsort = sortrows(outputsort, [5 6 7]);
        outputsort = cell2struct(outputsort, targetfields, 2);
        targetsort(a,b).Hits = outputsort;
    end
end
%%%%%make database of target hits

targetidx = struct('Hits',cell(numgenomes,numtargets)); %%%%target hit struct
for data
%                                 targetidx.Hits                                 =
struct('output',[],'genome',[],'target',[],'gen_tar',[],'scaffold',[],'indexr
ight',[],'indexleft',[]);
targethitsx = zeros(numgenomes,numtargets); %%%hit array for reference

%%%% scoring paramaters for hits%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Scoreval = 20;
Lengthval = 0.1;
Eval = 1E-100;
identval = 20;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%clean up hits based on scoring values
for a =1:numgenomes
    for b =1:numtargets
        count = 0;
        for c = 1:length(targetsort(a,b).Hits)
        scorenum = 0;
        lengthnum = 0;
        evalnum = 0;
        ident = 0;

    %           if results(a,b).output.Hits(c).HSPs(1).Score >= Scoreval
    %               scorenum = 1;
    %           end
```

```
                if     targetsort(a,b).Hits(c).HSPs.Identities.Percent     >=
identval
                    ident = 1;
                end
                cover = targetsort(a,b).Hits(c).HSPs.QueryIndices;
                coverval = cover(2)-cover(1);
                if                          coverval                         >
Lengthval*targetsort(a,b).Hits(c).output.Length
                    lengthnum = 1;
                end
                if targetsort(a,b).Hits(c).HSPs.Expect < Eval
                    evalnum = 1;
                end

%                 if ident + lengthnum + evalnum == 3
                  if ident == 1 &&  evalnum ==1
                    count = count+1;
                    targetidx(a,b).Hits(count).output                        =
targetsort(a,b).Hits(c).output;
                    targetidx(a,b).Hits(count).HSPs                          =
targetsort(a,b).Hits(c).HSPs;
                    targetidx(a,b).Hits(count).genome = a;
                    gentar = sprintf('target%d_hit_genome%d \n',b,a);
                    targetidx(a,b).Hits(count).gen_tar = gentar;
                    targetidx(a,b).Hits(count).target = b;
                    targetidx(a,b).Hits(count).scaffold                      =
targetsort(a,b).Hits(c).scaffold;
                    targetidx(a,b).Hits(count).indexleft                     =
targetsort(a,b).Hits(c).indexleft;
                    targetidx(a,b).Hits(count).indexright                    =
targetsort(a,b).Hits(c).indexright;
                  end

        end
    end
end

testx = zeros(size(targetidx,1),size(targetidx,2));
for a = 1:size(targetidx,1)
    for b = 1:size(targetidx,2)
        testx(a,b) = length(targetidx(a,b).Hits);
    end
end

%%%%eliminate duplicates
targetidxclean = struct('Hits',cell(size(targetidx,1),size(targetidx,2)));
clusterdist = 1000;

for a = 1:size(targetidx,1) %genomes
       for b = 1:size(targetidx,2) %targets
         countclean = 1;
         hitslen = length(targetidx(a,b).Hits);
         if length(targetidx(a,b).Hits) == 1
             hitslen = 1;
         else
             hitslen = length(targetidx(a,b).Hits)-1;
```

```
        end
           for c = 1:hitslen %hits

                if c == 1
                    targetidxclean(a,b).Hits(c).scaffold                    =
targetidx(a,b).Hits(c).scaffold;
                    targetidxclean(a,b).Hits(c).genome                      =
targetidx(a,b).Hits(c).genome;
                    targetidxclean(a,b).Hits(c).indexright                  =
targetidx(a,b).Hits(c).indexright;
                    targetidxclean(a,b).Hits(c).indexleft                   =
targetidx(a,b).Hits(c).indexleft;
                    targetidxclean(a,b).Hits(c).gen_tar                     =
targetidx(a,b).Hits(c).gen_tar;
                    targetidxclean(a,b).Hits(c).target                      =
targetidx(a,b).Hits(c).target;
                    targetidxclean(a,b).Hits(c).output                      =
targetidx(a,b).Hits(c).output;
                    targetidxclean(a,b).Hits(c).HSPs                        =
targetidx(a,b).Hits(c).HSPs;
                end
                if hitslen == 1
                   continue
                end
                distcheck9 = clusterdist + 100;
                distcheck10 = clusterdist + 100;
                distcheck11 = clusterdist + 100;
                distcheck12 = clusterdist + 100;
                tf3                                                         =
strcmp(targetidx(a,b).Hits(c).scaffold,targetidx(a,b).Hits(c+1).scaffold);
                    if tf3 == 1
                    distcheck9  =  abs(targetidx(a,b).Hits(c).indexright  -
targetidx(a,b).Hits(c+1).indexright);
                    distcheck10  =  abs(targetidx(a,b).Hits(c).indexright  -
targetidx(a,b).Hits(c+1).indexleft);
                    distcheck11  =  abs(targetidx(a,b).Hits(c).indexleft  -
targetidx(a,b).Hits(c+1).indexright);
                    distcheck12  =  abs(targetidx(a,b).Hits(c).indexleft  -
targetidx(a,b).Hits(c+1).indexleft);
                    end

                            tfcheck3 = 0;
                            boolean3                                        =
[distcheck9,distcheck10,distcheck11,distcheck12];
                            for x=1:length(boolean3)
                                if boolean3(x) <= clusterdist
                                    tfcheck3 = 1;
                                    break
                                end
                            end

                            if tf3 ~= 1 || tfcheck3 ~= 1
                                countclean = countclean+1;

targetidxclean(a,b).Hits(countclean).scaffold                               =
targetidx(a,b).Hits(c+1).scaffold;
```

```
targetidxclean(a,b).Hits(countclean).genome                                    =
targetidx(a,b).Hits(c+1).genome;

targetidxclean(a,b).Hits(countclean).indexright                                =
targetidx(a,b).Hits(c+1).indexright;

targetidxclean(a,b).Hits(countclean).indexleft                                 =
targetidx(a,b).Hits(c+1).indexleft;

targetidxclean(a,b).Hits(countclean).gen_tar                                   =
targetidx(a,b).Hits(c+1).gen_tar;

targetidxclean(a,b).Hits(countclean).target                                    =
targetidx(a,b).Hits(c+1).target;

targetidxclean(a,b).Hits(countclean).output                                    =
targetidx(a,b).Hits(c+1).output;
                                    targetidxclean(a,b).Hits(countclean).HSPs
= targetidx(a,b).Hits(c+1).HSPs;
                                end
                                if tf3 ==1 && tfcheck3 ==1 && a~=1
                                    arrayidx                                   =
[targetidx(a,b).Hits(c).indexright,targetidx(a,b).Hits(c).indexleft,targetidx
(a,b).Hits(c+1).indexright,targetidx(a,b).Hits(c+1).indexleft];
                                    arrayidx = sort(arrayidx);

targetidxclean(a,b).Hits(countclean).indexright = arrayidx(4);

targetidxclean(a,b).Hits(countclean).indexleft = arrayidx(1);
                                end
            end
        end
end

for a = 1:size(targetidxclean,1)
    for b = 1:size(targetidxclean,2)
        targethitsx(a,b) = length(targetidxclean(a,b).Hits);
    end
end

for a = 1:size(targetidxclean,1)
    for b = 1:size(targetidxclean,2)
        for c = 1:length(targetidxclean(a,b).Hits)
        targetidxclean(a,b).Hits(c).Hitnum                                     =
length(targetidxclean(a,b).Hits);
        end
    end
end


cd(directory);
cd(resultsfolder);
file2 = fopen('targethits_2nd.txt','w+');
for a=1:numtargets
   average = mean(targethitsx(:,a));
```

```
   medval = median(targethitsx(:,a));

   statstxt1 = sprintf('average hits of target%d is %d \n',a,average);
   statstxt2 = sprintf('median of target%d hits is %d \n', a, medval);

   fprintf(file2, statstxt1);
   fprintf(file2, statstxt2);
   fprintf(file2, '\n \n');
end

cd(directory);

file1 = sprintf('%dTargets_%dgenomes_2nd.xlsx',numtargets,numgenomes);
xlswrite(file1, targethitsx);
xlswrite('testx',testx);
xlswrite('test2',test2);

%%%%reduce hits index by checking for targets greater than average
targetfinalhits = struct('Hits',[]);
Hits                                                                    =
struct('output',[],'genome',[],'target',[],'gen_tar',[],'scaffold',[],'indexr
ight',[],'indexleft',[]);
t = 1; %%%%1 if we want to check targets greater than average, 0 if we don't
want to check

if t == 1
for a = 1:numgenomes
    countfinal = 0;
    for b = 1:numtargets
       average = mean(targethitsx(:,b));
       medval = median(targethitsx(:,b));

       if      length(targetidxclean(a,b).Hits)      >      average      ||
length(targetidxclean(a,b).Hits) > medval
           positivehit   =   sprintf('\n   target%d_hit_genome%d   :%d   hits
\n',b,a,length(targetidxclean(a,b).Hits));
           fprintf(file2, positivehit);

           for c = 1:length(targetidxclean(a,b).Hits)
               countfinal = countfinal + 1;
               targetfinalhits(a).Hits(countfinal).output                    =
targetidxclean(a,b).Hits(c).output;
               targetfinalhits(a).Hits(countfinal).genome                    =
targetidxclean(a,b).Hits(c).genome;
               targetfinalhits(a).Hits(countfinal).target                    =
targetidxclean(a,b).Hits(c).target;
               targetfinalhits(a).Hits(countfinal).gen_tar                   =
targetidxclean(a,b).Hits(c).gen_tar;
               targetfinalhits(a).Hits(countfinal).scaffold                  =
targetidxclean(a,b).Hits(c).scaffold;
               targetfinalhits(a).Hits(countfinal).indexleft                 =
targetidxclean(a,b).Hits(c).indexleft;
               targetfinalhits(a).Hits(countfinal).indexright                =
targetidxclean(a,b).Hits(c).indexright;
               targetfinalhits(a).Hits(countfinal).averagetarget = average;
```

```
                targetfinalhits(a).Hits(countfinal).mediantarget = medval;
                targetfinalhits(a).Hits(countfinal).targethits                =
length(targetidxclean(a,b).Hits);
                targetfinalhits(a).Hits(countfinal).HSPs                      =
targetidxclean(a,b).Hits(c).HSPs;
                targetfinalhits(a).Hits(countfinal).Hitnum                    =
targetidxclean(a,b).Hits(c).Hitnum;

                %%%print location of hits%%%%%5
                print0 = sprintf('hit #%d',c);
                fprintf(file2, print0);
                print1                          =                          strcat('\n
scaffold:',targetfinalhits(a).Hits(countfinal).scaffold,'\n');
                fprintf(file2, print1);
                numleft                                                       =
num2str(targetfinalhits(a).Hits(countfinal).indexleft);
                print2 = strcat('indexleft:',numleft,'\n');
                fprintf(file2, print2);
                numright                                                      =
num2str(targetfinalhits(a).Hits(countfinal).indexright);
                print3 = strcat('indexright:',numright,'\n');
                fprintf(file2, print3);
            end
        end
    end
end
elseif t == 0
    for a = 1:numgenomes
    countfinal = 0;
    for b = 1:numtargets

            for c = 1:length(targetidxclean(a,b).Hits)
                countfinal = countfinal + 1;
                targetfinalhits(a).Hits(countfinal).output                    =
targetidxclean(a,b).Hits(c).output;
                targetfinalhits(a).Hits(countfinal).genome                    =
targetidxclean(a,b).Hits(c).genome;
                targetfinalhits(a).Hits(countfinal).target                    =
targetidxclean(a,b).Hits(c).target;
                targetfinalhits(a).Hits(countfinal).gen_tar                   =
targetidxclean(a,b).Hits(c).gen_tar;
                targetfinalhits(a).Hits(countfinal).scaffold                  =
targetidxclean(a,b).Hits(c).scaffold;
                targetfinalhits(a).Hits(countfinal).indexleft                 =
targetidxclean(a,b).Hits(c).indexleft;
                targetfinalhits(a).Hits(countfinal).indexright                =
targetidxclean(a,b).Hits(c).indexright;
                targetfinalhits(a).Hits(countfinal).averagetarget = average;
                targetfinalhits(a).Hits(countfinal).mediantarget = medval;
                targetfinalhits(a).Hits(countfinal).targethits                =
length(targetidxclean(a,b).Hits);
                targetfinalhits(a).Hits(countfinal).HSPs                      =
targetidxclean(a,b).Hits(c).HSPs;
                targetfinalhits(a).Hits(countfinal).Hitnum                    =
targetidxclean(a,b).Hits(c).Hitnum;
                %%%print location of hits%%%%%5
```

```
                print0 = sprintf('hit #%d',c);
                fprintf(file2, print0);
                print1                          =                          strcat('\n
scaffold:',targetfinalhits(a).Hits(countfinal).scaffold,'\n');
                fprintf(file2, print1);
                numleft                                                             =
num2str(targetfinalhits(a).Hits(countfinal).indexleft);
                print2 = strcat('indexleft:',numleft,'\n');
                fprintf(file2, print2);
                numright                                                            =
num2str(targetfinalhits(a).Hits(countfinal).indexright);
                print3 = strcat('indexright:',numright,'\n');
                fprintf(file2, print3);
            end
        end
    end
end

cd(directory);
cd(resultsfolder);


save('targethits.mat','targetfinalhits','-v7.3');
```

Script 11: targetclusterfindv3.m

```
%%%%%find secondary metabolite gene clusters next to target hits%%%%%%%%
clear;
% directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains';
directory = 'C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_strains2';
% directory = 'cd C:\Users\Lab\Desktop\Genome_Mining_Matlab\In_house_genomes'
cd(directory);
% resultsfolder = 'newgenomes';
resultsfolder = 'Cyp51results';
cd(resultsfolder)
load('targethits.mat');
load('NRPSresults.mat');
load('PKSresults.mat');
load('terpeneresults.mat');
load('auxgenesclean.mat');
load('genomes.mat');

mkdir results
cd results
filecluster = fopen('results_02282020_Cyp51_PKS_NRPS.txt','w+');

residuepos = 10; %%%position of residue to be checked for

clusterdist = 20000; %%%%colocalization of secondary metabolite genes
clusterdist2 = 20000; %%%%colocalization of target and core gene
```

```
printini1 = sprintf('\nthreshold distance between target and core enzyme is
%d bp',clusterdist2);
printini2 = sprintf('\nthreshold distance between secondary metabolite genes
and enzyme and target is %d bp',clusterdist);

fprintf(filecluster,printini1);
fprintf(filecluster,printini2);


terpenecheck = 0; %set to 1 if you want to check terpene clusters
NRPS_PKScheck = 1; %set to 1 if you want to check NRPS and PKS clusers

%%%%find clusters near targets%%%%%


clustertargethit = 0;
for a = 1:length(targetfinalhits) %% iterate through num of genomes
        for  b  =  1:length(targetfinalhits(a).Hits)  %%going  through  positive
target hits
            if NRPS_PKScheck == 1
             %%%%%%%%%%%%%%%%%%%%check                for                NRPS
clusters%%%%%%%%%%%%%%%%%%%%
            for c = 1: length(NRPSclean(a).output) %%%check through NRPS
                    distcheck1 = clusterdist2 + 100;
                    distcheck2 = clusterdist2 + 100;
                    distcheck3 = clusterdist2 + 100;
                    distcheck4 = clusterdist2 + 100;
                tf1                                                          =
strcmp(targetfinalhits(a).Hits(b).scaffold,NRPSclean(a).output(c).scaffold);
                if tf1 ==1
                    distcheck1  =  abs(targetfinalhits(a).Hits(b).indexright  -
NRPSclean(a).output(c).indexright);
                    distcheck2  =  abs(targetfinalhits(a).Hits(b).indexright  -
NRPSclean(a).output(c).indexleft);
                    distcheck3  =  abs(targetfinalhits(a).Hits(b).indexleft  -
NRPSclean(a).output(c).indexright);
                    distcheck4  =  abs(targetfinalhits(a).Hits(b).indexleft  -
NRPSclean(a).output(c).indexleft);
                end
                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist2
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                   clustertargethit = clustertargethit + 1;
                   print0 = sprintf('\nhit#%d',clustertargethit);
                   fprintf(filecluster,print0);
                   print1 = sprintf('\ntarget T%d hit in genome %d near NRPS
cluster\n',targetfinalhits(a).Hits(b).target,targetfinalhits(a).Hits(b).genom
e);
                   fprintf(filecluster, print1);
```

```
%                   %%%%%%
                    targetname1       =       sprintf('target       T%d       is
',targetfinalhits(a).Hits(b).target);
                    targetname2                                                 =
targetkey(targetfinalhits(a).Hits(b).target,2);
                    printtarget = strcat(targetname1,{' '},targetname2);
                    fprintf(filecluster,printtarget);
                    targid        =        sprintf('\ntarget        id        is
%d',targetfinalhits(a).Hits(b).HSPs.Identities.Percent); %%%%%%%Print ID%
                    fprintf(filecluster, targid);
                    targe     =     sprintf('\ntarget     evalue     is     %d',
targetfinalhits(a).Hits(b).HSPs.Expect); %%%%%%%%%%%%%%%Print Evalue
                    fprintf(filecluster, targe);
                    genomename1       =       sprintf('\ngenome       %d       is
',targetfinalhits(a).Hits(b).genome);
                    genomename2 = key(targetfinalhits(a).Hits(b).genome,2);
                    printgenome = strcat(genomename1,{' '},genomename2);
                    fprintf(filecluster,printgenome);

                    hitnum1 = num2str(targetfinalhits(a).Hits(b).Hitnum);
                    targethitprint   =   strcat('\nnumber   of   target   hits   in
',genomename2,' is ',hitnum1);
                    fprintf(filecluster,targethitprint);
                    avgnum1                                                     =
num2str(targetfinalhits(a).Hits(b).averagetarget);
                    targetavgprint   =   strcat('\naverage   target   hits   in   all
genomes is ',avgnum1);
                    mednum1                                                     =
num2str(targetfinalhits(a).Hits(b).mediantarget);
                    targetmedprint   =   strcat('\nmedian   target   hits   in   all
genomes is ',mednum1);
                    fprintf(filecluster,targetavgprint);
                    fprintf(filecluster,targetmedprint);
%                   %%%%%%
                    print2          =          strcat('\ntarget          scaffold:
',targetfinalhits(a).Hits(b).scaffold,'\n');
                    fprintf(filecluster, print2);
                    numleft = num2str(targetfinalhits(a).Hits(b).indexleft);
                    print3 = strcat('target indexleft: ',numleft,'\n');
                    fprintf(filecluster, print3);
                    numright = num2str(targetfinalhits(a).Hits(b).indexright);
                    print4 = strcat('target indexright: ',numright,'\n');
                    fprintf(filecluster,print4);
                    numleft = num2str(NRPSclean(a).output(c).indexleft);
                    print6 = strcat('NRPS indexleft: ',numleft,'\n');
                    fprintf(filecluster, print6);
                    numright = num2str(NRPSclean(a).output(c).indexright);
                    print7 = strcat('NRPS indexright: ',numright,'\n');
                    fprintf(filecluster, print7);
                    %%%%%output residue checks for target%%%%%%%%

                    if    targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)    <=
residuepos && targetfinalhits(a).Hits(b).HSPs.QueryIndices(2) >= residuepos
                        count                                                   =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1);
```

```
                        for                             i                               =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1):targetfinalhits(a).Hits(b).HS
Ps.QueryIndices(2)
                            if count == residuepos
                                resstr = num2str(residuepos);
                                print8 = strcat('\nresidue #',resstr,' is_   ',
targetfinalhits(a).Hits(b).HSPs.Alignment(3,i-
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)+1),'\n');
                                fprintf(filecluster, print8);
                                break;
                            elseif
targetfinalhits(a).Hits(b).HSPs.Alignment(1,i) == '-'
                                count = count;
                            else
                                count = count+1;
                            end
                        end
                    end
                    %%%%%%%%%%check for TF genes%%%%%%%%%%%%%%%%%%%
                    znnum = 0;
                    tfnum = 0;
                    for   aa   =   1:length(TFclean(a,2).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(TFclean(a,2).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(TFclean(a,2).output(aa).indexright - NRPSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(TFclean(a,2).output(aa).indexright - NRPSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(TFclean(a,2).output(aa).indexleft - NRPSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(TFclean(a,2).output(aa).indexleft - NRPSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           znnum = znnum + 1;
                           print1  =  sprintf('ZN  binding  domain  gene  #%d  in
cluster\n',znnum);
                           fprintf(filecluster, print1);
                           numleft                                              =
num2str(TFclean(a,2).output(aa).indexleft);
                           print3       =       strcat('ZNbind       indexleft:
',numleft,'\n');
```

```
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(TFclean(a,2).output(aa).indexright);
                            print4      =      strcat('ZNbind      indexright:
',numright,'\n');
                            fprintf(filecluster, print4);
                        end
                    end
                            for  bb  =  1:length(TFclean(a,1).output)  %%%check
for TF domain
                                    distcheck1 = clusterdist + 100;
                                    distcheck2 = clusterdist + 100;
                                    distcheck3 = clusterdist + 100;
                                    distcheck4 = clusterdist + 100;
                                tfbb                                                =
strcmp(TFclean(a,1).output(bb).scaffold,NRPSclean(a).output(c).scaffold);
                                    if tfbb ==1
                                    distcheck1                                      =
abs(TFclean(a,1).output(bb).indexright - NRPSclean(a).output(c).indexright);
                                    distcheck2                                      =
abs(TFclean(a,1).output(bb).indexright - NRPSclean(a).output(c).indexleft);
                                    distcheck3                                      =
abs(TFclean(a,1).output(bb).indexleft - NRPSclean(a).output(c).indexright);
                                    distcheck4                                      =
abs(TFclean(a,1).output(bb).indexleft - NRPSclean(a).output(c).indexleft);
                                    end
                                boolean1                                            =
[distcheck1,distcheck2,distcheck3,distcheck4];
                                tfbbcheck1 = 0;
                                    for i=1:length(boolean1)
                                        if boolean1(i) <= clusterdist
                                            tfbbcheck1 = 1;
                                            break
                                        end
                                    end

                            if tfbbcheck1 == 1
                            tfnum = tfnum + 1;
                            print1     =     sprintf('TF     gene     #%d     in
cluster\n',tfnum);
                            fprintf(filecluster, print1);
                            numleft                                                 =
num2str(TFclean(a,1).output(bb).indexleft);
                            print3 = strcat('TF indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(TFclean(a,1).output(bb).indexright);
                            print4 = strcat('TF indexright: ',numleft,'\n');
                            fprintf(filecluster, print4);
                            end
                            end
                    %%%%%%%%%%check for P450 genes%%%%%%%%%%%%%%%%%%%
                    p450num = 0;
                    for   aa   =   1:length(P450clean(a).output)   %%%check   for
Znbinding domain
                             distcheck1 = clusterdist + 100;
                             distcheck2 = clusterdist + 100;
```

```
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(P450clean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(P450clean(a).output(aa).indexright - NRPSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(P450clean(a).output(aa).indexright - NRPSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(P450clean(a).output(aa).indexleft - NRPSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(P450clean(a).output(aa).indexleft - NRPSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           p450num = p450num + 1;
                           print1 = sprintf('P450 #%d in cluster\n',p450num);
                           fprintf(filecluster, print1);
                           numleft                                              =
num2str(P450clean(a).output(aa).indexleft);
                           print3 = strcat('P450 indexleft: ',numleft,'\n');
                           fprintf(filecluster, print3);
                           numright                                             =
num2str(P450clean(a).output(aa).indexright);
                           print4       =       strcat('P450       indexright:
',numright,'\n');
                           fprintf(filecluster, print4);
                        end
                    end
                     %%%%%%%%%%check for FMO genes%%%%%%%%%%%%%%%%%%
                   FMOnum = 0;
                    for   aa   =   1:length(FMOclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(FMOclean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(FMOclean(a).output(aa).indexright - NRPSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(FMOclean(a).output(aa).indexright - NRPSclean(a).output(c).indexleft);
                            distcheck3  =  abs(FMOclean(a).output(aa).indexleft
- NRPSclean(a).output(c).indexright);
                            distcheck4  =  abs(FMOclean(a).output(aa).indexleft
- NRPSclean(a).output(c).indexleft);
```

```
                        end
                    boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                    tfaacheck1 = 0;
                        for i=1:length(boolean1)
                            if boolean1(i) <= clusterdist
                                tfaacheck1 = 1;
                                break
                            end
                        end
                    if tfaacheck1 == 1
                       FMOnum = FMOnum + 1;
                       print1 = sprintf('FMO #%d in cluster\n',FMOnum);
                       fprintf(filecluster, print1);
                       numleft                                              =
num2str(FMOclean(a).output(aa).indexleft);
                       print3 = strcat('FMO indexleft: ',numleft,'\n');
                       fprintf(filecluster, print3);
                       numright                                             =
num2str(FMOclean(a).output(aa).indexright);
                       print4 = strcat('FMO indexright: ',numright,'\n');
                       fprintf(filecluster, print4);
                    end
                end
                 %%%%%%%%%%check for MT genes%%%%%%%%%%%%%%%%%%
               MTnum = 0;
                for   aa   =   1:length(MTclean(a).output)   %%%check   for
Znbinding domain
                        distcheck1 = clusterdist + 100;
                        distcheck2 = clusterdist + 100;
                        distcheck3 = clusterdist + 100;
                        distcheck4 = clusterdist + 100;
                    tfaa                                                    =
strcmp(MTclean(a).output(aa).scaffold,NRPSclean(a).output(c).scaffold);
                        if tfaa ==1
                        distcheck1  =  abs(MTclean(a).output(aa).indexright
- NRPSclean(a).output(c).indexright);
                        distcheck2  =  abs(MTclean(a).output(aa).indexright
- NRPSclean(a).output(c).indexleft);
                        distcheck3  =  abs(MTclean(a).output(aa).indexleft
- NRPSclean(a).output(c).indexright);
                        distcheck4  =  abs(MTclean(a).output(aa).indexleft
- NRPSclean(a).output(c).indexleft);
                        end
                    boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                    tfaacheck1 = 0;
                        for i=1:length(boolean1)
                            if boolean1(i) <= clusterdist
                                tfaacheck1 = 1;
                                break
                            end
                        end
                    if tfaacheck1 == 1
                       MTnum = MTnum + 1;
                       print1 = sprintf('MT #%d in cluster\n',MTnum);
                       fprintf(filecluster, print1);
```

```
                            numleft                                                 =
num2str(MTclean(a).output(aa).indexleft);
                            print3 = strcat('MT indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(MTclean(a).output(aa).indexright);
                            print4 = strcat('MT indexright: ',numright,'\n');
                            fprintf(filecluster, print4);
                        end
                    end
                end
             end
            %%%%%%%%%%%%%%%%%%%%%%check for PKS clusters%%%%%%%%%%%%%%%%%%%%%%
            for c = 1: length(PKSclean(a).output) %%%check through PKS
                    distcheck1 = clusterdist2 + 100;
                    distcheck2 = clusterdist2 + 100;
                    distcheck3 = clusterdist2 + 100;
                    distcheck4 = clusterdist2 + 100;
                tf1                                                                 =
strcmp(targetfinalhits(a).Hits(b).scaffold,PKSclean(a).output(c).scaffold);
                if tf1 ==1
                    distcheck1  =  abs(targetfinalhits(a).Hits(b).indexright  -
PKSclean(a).output(c).indexright);
                    distcheck2  =  abs(targetfinalhits(a).Hits(b).indexright  -
PKSclean(a).output(c).indexleft);
                    distcheck3  =  abs(targetfinalhits(a).Hits(b).indexleft  -
PKSclean(a).output(c).indexright);
                    distcheck4  =  abs(targetfinalhits(a).Hits(b).indexleft  -
PKSclean(a).output(c).indexleft);
                end
                    boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                    tfcheck1 = 0;
                    for i=1:length(boolean1)
                        if boolean1(i) <= clusterdist2
                            tfcheck1 = 1;
                            break
                        end
                    end
                if tfcheck1 == 1
                   clustertargethit = clustertargethit + 1;
                   print0 = sprintf('\nhit#%d',clustertargethit);
                   fprintf(filecluster,print0);
                   print1 = sprintf('\ntarget T%d hit in genome %d near PKS
cluster\n',targetfinalhits(a).Hits(b).target,targetfinalhits(a).Hits(b).genom
e);
                   fprintf(filecluster, print1);
                   %                    %%%%%%
                   targetname1       =       sprintf('target       T%d       is
',targetfinalhits(a).Hits(b).target);
                   targetname2                                                      =
targetkey(targetfinalhits(a).Hits(b).target,2);
                   printtarget = strcat(targetname1,{' '},targetname2);
                   fprintf(filecluster,printtarget);
                   targid        =        sprintf('\ntarget        id        is
%d',targetfinalhits(a).Hits(b).HSPs.Identities.Percent); %%%%%%%Print ID%
                   fprintf(filecluster, targid);
```

```
                    targe     =     sprintf('\ntarget     evalue     is     %d',
targetfinalhits(a).Hits(b).HSPs.Expect); %%%%%%%%%%%%%%%Print Evalue
                    fprintf(filecluster, targe);
                    genomename1       =       sprintf('\ngenome       %d       is
',targetfinalhits(a).Hits(b).genome);
                    genomename2 = key(targetfinalhits(a).Hits(b).genome,2);
                    printgenome = strcat(genomename1,{' '},genomename2);
                    fprintf(filecluster,printgenome);

                    hitnum1 = num2str(targetfinalhits(a).Hits(b).Hitnum);
                    targethitprint   =   strcat('\nnumber   of   target   hits   in
',genomename2,' is ',hitnum1);
                    fprintf(filecluster,targethitprint);
                    avgnum1                                                    =
num2str(targetfinalhits(a).Hits(b).averagetarget);
                    targetavgprint   =   strcat('\naverage   target   hits   in   all
genomes is ',avgnum1);
                    mednum1                                                    =
num2str(targetfinalhits(a).Hits(b).mediantarget);
                    targetmedprint   =   strcat('\nmedian   target   hits   in   all
genomes is ',mednum1);
                    fprintf(filecluster,targetavgprint);
                    fprintf(filecluster,targetmedprint);
%                     %%%%%%

                    print2         =         strcat('\ntarget         scaffold:
',targetfinalhits(a).Hits(b).scaffold,'\n');
                    fprintf(filecluster, print2);
                    numleft = num2str(targetfinalhits(a).Hits(b).indexleft);
                    print3 = strcat('target indexleft: ',numleft,'\n');
                    fprintf(filecluster, print3);
                    numright = num2str(targetfinalhits(a).Hits(b).indexright);
                    print4 = strcat('target indexright: ',numright,'\n');
                    fprintf(filecluster,print4);
                    numleft = num2str(PKSclean(a).output(c).indexleft);
                    print6 = strcat('PKS indexleft: ',numleft,'\n');
                    fprintf(filecluster, print6);
                    numright = num2str(PKSclean(a).output(c).indexright);
                    print7 = strcat('PKS indexright: ',numright,'\n');
                    fprintf(filecluster, print7);
                    %%%%%output residue checks for target%%%%%%%%

                    if    targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)    <=
residuepos && targetfinalhits(a).Hits(b).HSPs.QueryIndices(2) >= residuepos
                        count                                                  =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1);
                        for                              i                     =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1):targetfinalhits(a).Hits(b).HS
Ps.QueryIndices(2)
                            if count == residuepos
                                resstr = num2str(residuepos);
                                print8 = strcat('\nresidue #',resstr,' is_   ',
targetfinalhits(a).Hits(b).HSPs.Alignment(3,i-
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)+1),'\n');
                                fprintf(filecluster, print8);
                                break;
```

```
                            elseif
targetfinalhits(a).Hits(b).HSPs.Alignment(1,i) == '-'
                                count = count;
                            else
                                count = count+1;
                            end
                        end
                    end
                    %%%%%%%%%%check for TF genes%%%%%%%%%%%%%%%%%%
                    znnum = 0;
                    tfnum = 0;
                    for   aa   =   1:length(TFclean(a,2).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(TFclean(a,2).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(TFclean(a,2).output(aa).indexright - PKSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(TFclean(a,2).output(aa).indexright - PKSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(TFclean(a,2).output(aa).indexleft - PKSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(TFclean(a,2).output(aa).indexleft - PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            znnum = znnum + 1;
                            print1  =  sprintf('ZN  binding  domain  #%d  gene  in
cluster\n',znnum);
                            fprintf(filecluster, print1);
                            numleft                                             =
num2str(TFclean(a,2).output(aa).indexleft);
                            print3      =      strcat('ZNbind      indexleft:
',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                            =
num2str(TFclean(a,2).output(aa).indexright);
                            print4      =      strcat('ZNbind      indexright:
',numright,'\n');
                            fprintf(filecluster, print4);
                        end
                    end
                            for  bb  =  1:length(TFclean(a,1).output)  %%%check
for TF domain
```

```
                                    distcheck1 = clusterdist + 100;
                                    distcheck2 = clusterdist + 100;
                                    distcheck3 = clusterdist + 100;
                                    distcheck4 = clusterdist + 100;
                                tfbb                                                =
strcmp(TFclean(a,1).output(bb).scaffold,PKSclean(a).output(c).scaffold);
                                    if tfbb ==1
                                    distcheck1                                      =
abs(TFclean(a,1).output(bb).indexright - PKSclean(a).output(c).indexright);
                                    distcheck2                                      =
abs(TFclean(a,1).output(bb).indexright - PKSclean(a).output(c).indexleft);
                                    distcheck3                                      =
abs(TFclean(a,1).output(bb).indexleft - PKSclean(a).output(c).indexright);
                                    distcheck4                                      =
abs(TFclean(a,1).output(bb).indexleft - PKSclean(a).output(c).indexleft);
                                    end
                                boolean1                                            =
[distcheck1,distcheck2,distcheck3,distcheck4];
                                tfbbcheck1 = 0;
                                    for i=1:length(boolean1)
                                        if boolean1(i) <= clusterdist
                                            tfbbcheck1 = 1;
                                            break
                                        end
                                    end

                            if tfbbcheck1 == 1
                            tfnum = tfnum + 1;
                            print1     =     sprintf('TF     gene     #%d     in
cluster\n',tfnum);
                            fprintf(filecluster, print1);
                            numleft                                                 =
num2str(TFclean(a,1).output(bb).indexleft);
                            print3 = strcat('TF indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(TFclean(a,1).output(bb).indexright);
                            print4 = strcat('TF indexright: ',numleft,'\n');
                            fprintf(filecluster, print4);
                            end
                            end


                     %%%%%%%%%%check for P450 genes%%%%%%%%%%%%%%%%%%%
                    p450num = 0;
                    for   aa   =   1:length(P450clean(a).output)   %%%check   for
Znbinding domain
                             distcheck1 = clusterdist + 100;
                             distcheck2 = clusterdist + 100;
                             distcheck3 = clusterdist + 100;
                             distcheck4 = clusterdist + 100;
                          tfaa                                                      =
strcmp(P450clean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                             if tfaa ==1
                             distcheck1                                             =
abs(P450clean(a).output(aa).indexright - PKSclean(a).output(c).indexright);
```

```
                            distcheck2                                          =
abs(P450clean(a).output(aa).indexright - PKSclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(P450clean(a).output(aa).indexleft - PKSclean(a).output(c).indexright);
                            distcheck4                                          =
abs(P450clean(a).output(aa).indexleft - PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           p450num = p450num + 1;
                           print1 = sprintf('P450 #%d in cluster\n',p450num);
                           fprintf(filecluster, print1);
                           numleft                                              =
num2str(P450clean(a).output(aa).indexleft);
                           print3 = strcat('P450 indexleft: ',numleft,'\n');
                           fprintf(filecluster, print3);
                           numright                                             =
num2str(P450clean(a).output(aa).indexright);
                           print4       =       strcat('P450       indexright:
',numright,'\n');
                           fprintf(filecluster, print4);
                        end
                    end
                      %%%%%%%%%%check for FMO genes%%%%%%%%%%%%%%%%%%%
                   FMOnum = 0;
                    for   aa   =   1:length(FMOclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(FMOclean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexright);
                            distcheck2                                          =
abs(FMOclean(a).output(aa).indexright - PKSclean(a).output(c).indexleft);
                            distcheck3  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
                            distcheck4  =  abs(FMOclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
```

```
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            FMOnum = FMOnum + 1;
                            print1 = sprintf('FMO #%d in cluster\n',FMOnum);
                            fprintf(filecluster, print1);
                            numleft                                                 =
num2str(FMOclean(a).output(aa).indexleft);
                            print3 = strcat('FMO indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(FMOclean(a).output(aa).indexright);
                            print4 = strcat('FMO indexright: ',numright,'\n');
                            fprintf(filecluster, print4);
                        end
                    end
                     %%%%%%%%%%check for MT genes%%%%%%%%%%%%%%%%%%
                   MTnum = 0;
                    for   aa   =   1:length(MTclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                        =
strcmp(MTclean(a).output(aa).scaffold,PKSclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexright);
                            distcheck2  =  abs(MTclean(a).output(aa).indexright
- PKSclean(a).output(c).indexleft);
                            distcheck3   =   abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexright);
                            distcheck4   =   abs(MTclean(a).output(aa).indexleft
- PKSclean(a).output(c).indexleft);
                            end
                        boolean1                                                    =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            MTnum = MTnum + 1;
                            print1 = sprintf('MT #%d in cluster\n',MTnum);
                            fprintf(filecluster, print1);
                            numleft                                                 =
num2str(MTclean(a).output(aa).indexleft);
                            print3 = strcat('MT indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(MTclean(a).output(aa).indexright);
                            print4 = strcat('MT indexright: ',numright,'\n');
```

```
                        fprintf(filecluster, print4);
                    end
                end
            end
        end
         end
        if terpenecheck == 1
         %%%%%%%%%%%%%%%%%%%%%%check              for              terpene
clusters%%%%%%%%%%%%%%%%%%%%%
        for  c  =  1:  length(terpeneclean(a).output)  %%%check  through
terpene
                distcheck1 = clusterdist2 + 100;
                distcheck2 = clusterdist2 + 100;
                distcheck3 = clusterdist2 + 100;
                distcheck4 = clusterdist2 + 100;
            tf1                                                          =
strcmp(targetfinalhits(a).Hits(b).scaffold,terpeneclean(a).output(c).scaffold
);
            if tf1 ==1
                distcheck1  =  abs(targetfinalhits(a).Hits(b).indexright  -
terpeneclean(a).output(c).indexright);
                distcheck2  =  abs(targetfinalhits(a).Hits(b).indexright  -
terpeneclean(a).output(c).indexleft);
                distcheck3  =  abs(targetfinalhits(a).Hits(b).indexleft  -
terpeneclean(a).output(c).indexright);
                distcheck4  =  abs(targetfinalhits(a).Hits(b).indexleft  -
terpeneclean(a).output(c).indexleft);
            end
                boolean1 = [distcheck1,distcheck2,distcheck3,distcheck4];
                tfcheck1 = 0;
                for i=1:length(boolean1)
                    if boolean1(i) <= clusterdist2
                        tfcheck1 = 1;
                        break
                    end
                end
            if tfcheck1 == 1
               clustertargethit = clustertargethit + 1;
               print0 = sprintf('\nhit#%d',clustertargethit);
               fprintf(filecluster,print0);
               print1  =  sprintf('\ntarget  T%d  hit  in  genome  %d  near
terpene
cluster\n',targetfinalhits(a).Hits(b).target,targetfinalhits(a).Hits(b).genom
e);
                fprintf(filecluster, print1);
                print11                    =                    strcat('this
is_',terpeneclean(a).output(c).type,'_cluster\n');
                fprintf(filecluster, print11);

%                   %%%%%%
                targetname1       =       sprintf('target       T%d       is
',targetfinalhits(a).Hits(b).target);
                targetname2                                                  =
targetkey(targetfinalhits(a).Hits(b).target,2);
                printtarget = strcat(targetname1,{' '},targetname2);
                fprintf(filecluster,printtarget);
```

```
                    targid       =       sprintf('\ntarget       id       is
%d',targetfinalhits(a).Hits(b).HSPs.Identities.Percent); %%%%%%%Print ID%
                    fprintf(filecluster, targid);
                    targe     =     sprintf('\ntarget     evalue     is     %d',
targetfinalhits(a).Hits(b).HSPs.Expect); %%%%%%%%%%%%%%%Print Evalue
                    fprintf(filecluster, targe);
                    genomename1      =      sprintf('\ngenome      %d      is
',targetfinalhits(a).Hits(b).genome);
                    genomename2 = key(targetfinalhits(a).Hits(b).genome,2);
                    printgenome = strcat(genomename1,{' '},genomename2);
                    fprintf(filecluster,printgenome);

                    hitnum1 = num2str(targetfinalhits(a).Hits(b).Hitnum);
                    targethitprint  =  strcat('\nnumber  of  target  hits  in
',genomename2,' is ',hitnum1);
                    fprintf(filecluster,targethitprint);
                    avgnum1                                                 =
num2str(targetfinalhits(a).Hits(b).averagetarget);
                    targetavgprint  =  strcat('\naverage  target  hits  in  all
genomes is ',avgnum1);
                    mednum1                                                 =
num2str(targetfinalhits(a).Hits(b).mediantarget);
                    targetmedprint  =  strcat('\nmedian  target  hits  in  all
genomes is ',mednum1);
                    fprintf(filecluster,targetavgprint);
                    fprintf(filecluster,targetmedprint);
%                     %%%%%%
                    print2         =         strcat('\ntarget         scaffold:
',targetfinalhits(a).Hits(b).scaffold,'\n');
                    fprintf(filecluster, print2);
                    numleft = num2str(targetfinalhits(a).Hits(b).indexleft);
                    print3 = strcat('target indexleft: ',numleft,'\n');
                    fprintf(filecluster, print3);
                    numright = num2str(targetfinalhits(a).Hits(b).indexright);
                    print4 = strcat('target indexright: ',numright,'\n');
                    fprintf(filecluster,print4);
                    numleft = num2str(terpeneclean(a).output(c).indexleft);
                    print6 = strcat('terpene indexleft: ',numleft,'\n');
                    fprintf(filecluster, print6);
                    numright = num2str(terpeneclean(a).output(c).indexright);
                    print7 = strcat('terpene indexright: ',numright,'\n');
                    fprintf(filecluster, print7);
                    %%%%%output residue checks for target%%%%%%%%

                    if    targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)    <=
residuepos && targetfinalhits(a).Hits(b).HSPs.QueryIndices(2) >= residuepos
                        count                                               =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1);
                        for                         i                       =
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1):targetfinalhits(a).Hits(b).HS
Ps.QueryIndices(2)
                            if count == residuepos
                                resstr = num2str(residuepos);
                                print8 = strcat('\nresidue #',resstr,' is_   ',
targetfinalhits(a).Hits(b).HSPs.Alignment(3,i-
targetfinalhits(a).Hits(b).HSPs.QueryIndices(1)+1),'\n');
```

```
                                fprintf(filecluster, print8);
                                break;
                            elseif
targetfinalhits(a).Hits(b).HSPs.Alignment(1,i) == '-'
                                count = count;
                            else
                                count = count+1;
                            end
                        end
                    end
                    %%%%%%%%%%check for TF genes%%%%%%%%%%%%%%%%%%
                    znnum = 0;
                    tfnum = 0;
                    for   aa   =   1:length(TFclean(a,2).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(TFclean(a,2).output(aa).scaffold,terpeneclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                          =
abs(TFclean(a,2).output(aa).indexright                                          -
terpeneclean(a).output(c).indexright);
                            distcheck2                                          =
abs(TFclean(a,2).output(aa).indexright                                          -
terpeneclean(a).output(c).indexleft);
                            distcheck3                                          =
abs(TFclean(a,2).output(aa).indexleft                                           -
terpeneclean(a).output(c).indexright);
                            distcheck4                                          =
abs(TFclean(a,2).output(aa).indexleft - terpeneclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                            znnum = znnum + 1;
                            print1  =  sprintf('ZN  binding  domain  gene  #%d  in
cluster\n',znnum);
                            fprintf(filecluster, print1);
                            numleft                                             =
num2str(TFclean(a,2).output(aa).indexleft);
                            print3      =      strcat('ZNbind      indexleft:
',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                            =
num2str(TFclean(a,2).output(aa).indexright);
                            print4      =      strcat('ZNbind      indexright:
',numright,'\n');
```

```
                        fprintf(filecluster, print4);
                    end
                end
                        for  bb  =  1:length(TFclean(a,1).output)  %%%check
for TF domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfbb                                                =
strcmp(TFclean(a,1).output(bb).scaffold,terpeneclean(a).output(c).scaffold);
                            if tfbb ==1
                            distcheck1                                      =
abs(TFclean(a,1).output(bb).indexright                                      -
terpeneclean(a).output(c).indexright);
                            distcheck2                                      =
abs(TFclean(a,1).output(bb).indexright                                      -
terpeneclean(a).output(c).indexleft);
                            distcheck3                                      =
abs(TFclean(a,1).output(bb).indexleft                                       -
terpeneclean(a).output(c).indexright);
                            distcheck4                                      =
abs(TFclean(a,1).output(bb).indexleft - terpeneclean(a).output(c).indexleft);
                            end
                        boolean1                                            =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfbbcheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfbbcheck1 = 1;
                                    break
                                end
                            end

                        if tfbbcheck1 == 1
                        tfnum = tfnum + 1;
                        print1     =     sprintf('TF     gene     #%d     in
cluster\n',tfnum);
                        fprintf(filecluster, print1);
                        numleft                                             =
num2str(TFclean(a,1).output(bb).indexleft);
                        print3 = strcat('TF indexleft: ',numleft,'\n');
                        fprintf(filecluster, print3);
                        numright                                            =
num2str(TFclean(a,1).output(bb).indexright);
                        print4 = strcat('TF indexright: ',numleft,'\n');
                        fprintf(filecluster, print4);
                        end
                        end
                %%%%%%%%%%check for P450 genes%%%%%%%%%%%%%%%%%%%
                p450num = 0;
                for   aa   =   1:length(P450clean(a).output)   %%%check   for
Znbinding domain
                         distcheck1 = clusterdist + 100;
                         distcheck2 = clusterdist + 100;
                         distcheck3 = clusterdist + 100;
                         distcheck4 = clusterdist + 100;
```

```
                        tfaa                                                =
strcmp(P450clean(a).output(aa).scaffold,terpeneclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                      =
abs(P450clean(a).output(aa).indexright                                      -
terpeneclean(a).output(c).indexright);
                            distcheck2                                      =
abs(P450clean(a).output(aa).indexright                                      -
terpeneclean(a).output(c).indexleft);
                            distcheck3                                      =
abs(P450clean(a).output(aa).indexleft                                       -
terpeneclean(a).output(c).indexright);
                            distcheck4                                      =
abs(P450clean(a).output(aa).indexleft - terpeneclean(a).output(c).indexleft);
                            end
                        boolean1                                            =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           p450num = p450num + 1;
                           print1 = sprintf('P450 #%d in cluster\n',p450num);
                           fprintf(filecluster, print1);
                           numleft                                          =
num2str(P450clean(a).output(aa).indexleft);
                           print3 = strcat('P450 indexleft: ',numleft,'\n');
                           fprintf(filecluster, print3);
                           numright                                         =
num2str(P450clean(a).output(aa).indexright);
                           print4       =       strcat('P450       indexright:
',numright,'\n');
                           fprintf(filecluster, print4);
                        end
                    end
                     %%%%%%%%%%check for FMO genes%%%%%%%%%%%%%%%%%%%
                   FMOnum = 0;
                    for   aa   =   1:length(FMOclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                =
strcmp(FMOclean(a).output(aa).scaffold,terpeneclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1                                      =
abs(FMOclean(a).output(aa).indexright                                       -
terpeneclean(a).output(c).indexright);
                            distcheck2                                      =
abs(FMOclean(a).output(aa).indexright - terpeneclean(a).output(c).indexleft);
                            distcheck3  =  abs(FMOclean(a).output(aa).indexleft
- terpeneclean(a).output(c).indexright);
```

```
                            distcheck4  =  abs(FMOclean(a).output(aa).indexleft
- terpeneclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           FMOnum = FMOnum + 1;
                           print1 = sprintf('FMO #%d in cluster\n',FMOnum);
                           fprintf(filecluster, print1);
                           numleft                                              =
num2str(FMOclean(a).output(aa).indexleft);
                           print3 = strcat('FMO indexleft: ',numleft,'\n');
                           fprintf(filecluster, print3);
                           numright                                             =
num2str(FMOclean(a).output(aa).indexright);
                           print4 = strcat('FMO indexright: ',numright,'\n');
                           fprintf(filecluster, print4);
                        end
                    end
                     %%%%%%%%%%check for MT genes%%%%%%%%%%%%%%%%%%%
                   MTnum = 0;
                    for   aa   =   1:length(MTclean(a).output)   %%%check   for
Znbinding domain
                            distcheck1 = clusterdist + 100;
                            distcheck2 = clusterdist + 100;
                            distcheck3 = clusterdist + 100;
                            distcheck4 = clusterdist + 100;
                        tfaa                                                    =
strcmp(MTclean(a).output(aa).scaffold,terpeneclean(a).output(c).scaffold);
                            if tfaa ==1
                            distcheck1  =  abs(MTclean(a).output(aa).indexright
- terpeneclean(a).output(c).indexright);
                            distcheck2  =  abs(MTclean(a).output(aa).indexright
- terpeneclean(a).output(c).indexleft);
                            distcheck3  =  abs(MTclean(a).output(aa).indexleft
- terpeneclean(a).output(c).indexright);
                            distcheck4  =  abs(MTclean(a).output(aa).indexleft
- terpeneclean(a).output(c).indexleft);
                            end
                        boolean1                                                =
[distcheck1,distcheck2,distcheck3,distcheck4];
                        tfaacheck1 = 0;
                            for i=1:length(boolean1)
                                if boolean1(i) <= clusterdist
                                    tfaacheck1 = 1;
                                    break
                                end
                            end
                        if tfaacheck1 == 1
                           MTnum = MTnum + 1;
```

```
                            print1 = sprintf('MT #%d in cluster\n',MTnum);
                            fprintf(filecluster, print1);
                            numleft                                                 =
num2str(MTclean(a).output(aa).indexleft);
                            print3 = strcat('MT indexleft: ',numleft,'\n');
                            fprintf(filecluster, print3);
                            numright                                                =
num2str(MTclean(a).output(aa).indexright);
                            print4 = strcat('MT indexright: ',numright,'\n');
                            fprintf(filecluster, print4);
                        end
                    end
                end
            end
           end%%%%%%end of terpene loop
        end
end

totalhits = sprintf('\nTotal of %d Hits!',clustertargethit);
fprintf(filecluster, totalhits);
```

8. References

1. L. Katz and R. H. Baltz, *J Ind Microbiol Biotechnol*, 2016, **43**, 155-176.
2. G. L. Challis, *J Med Chem*, 2008, **51**, 2618-2628.
3. A. L. Demain and A. Fang, *Adv Biochem Eng Biotechnol*, 2000, **69**, 1-39.
4. D. J. Newman, G. M. Cragg and D. G. I. Kingston, *Practice of Medicinal Chemistry, 3rd Edition*, 2008, DOI: Doi 10.1016/B978-0-12-374194-3.00008-1, 159-186.
5. R. Quinn, *Am J Public Health*, 2013, **103**, 426-434.
6. W. Zheng, N. Thorne and J. C. McKew, *Drug Discov Today*, 2013, **18**, 1067-1073.
7. J. G. Moffat, J. Rudolph and D. Bailey, *Nat Rev Drug Discov*, 2014, **13**, 588-602.
8. M. L. Nelson and S. B. Levy, *Ann N Y Acad Sci*, 2011, **1241**, 17-32.
9. D. Tedesco and L. Haragsim, *J Transplant*, 2012, **2012**, 230386.
10. A. Endo, *Proc Jpn Acad Ser B Phys Biol Sci*, 2010, **86**, 484-493.
11. B. O. Bachmann, S. G. Van Lanen and R. H. Baltz, *J Ind Microbiol Biot*, 2014, **41**, 175-184.
12. J. M. Winter, S. Behnken and C. Hertweck, *Current opinion in chemical biology*, 2011, **15**, 22-31.
13. K. Scherlach and C. Hertweck, *Org Biomol Chem*, 2006, **4**, 3517-3520.
14. Y. M. Chiang, E. Szewczyk, T. Nayak, A. D. Davidson, J. F. Sanchez, H. C. Lo, W. Y. Ho, H. Simityan, E. Kuo, A. Praseuth, K. Watanabe, B. R. Oakley and C. C. C. Wang, *Chem Biol*, 2008, **15**, 527-532.
15. J. Staunton and K. J. Weissman, *Nat Prod Rep*, 2001, **18**, 380-416.
16. S. Dutta, J. R. Whicher, D. A. Hansen, W. A. Hale, J. A. Chemler, G. R. Congdon, A. R. Narayan, K. Hakansson, D. H. Sherman, J. L. Smith and G. Skiniotis, *Nature*, 2014, **510**, 512-517.
17. R. J. Cox and T. J. Simpson, *Methods Enzymol*, 2009, **459**, 49-78.
18. Y. H. Chooi and Y. Tang, *J Org Chem*, 2012, **77**, 9933-9953.
19. Y. Tang, C. Y. Kim, Mathews, II, D. E. Cane and C. Khosla, *Proc Natl Acad Sci U S A*, 2006, **103**, 11124-11129.
20. J. Lim, R. Kong, E. Murugan, C. L. Ho, Z. X. Liang and D. Yang, *Plos One*, 2011, **6**.
21. J. T. Zheng and A. T. Keatinge-Clay, *Medchemcomm*, 2013, **4**, 34-40.
22. D. L. Akey, J. R. Razelun, J. Tehranisa, D. H. Sherman, W. H. Gerwick and J. L. Smith, *Structure*, 2010, **18**, 94-105.
23. B. D. Ames, C. Nguyen, J. Bruegger, P. Smith, W. Xu, S. Ma, E. Wong, S. Wong, X. K. Xie, J. W. H. Li, J. C. Vederas, Y. Tang and S. C. Tsai, *P Natl Acad Sci USA*, 2012, **109**, 11144-11149.
24. P. J. Bhetariya, M. Prajapati, A. Bhaduri, R. S. Mandal, A. Varma, T. Madan, Y. Singh and P. U. Sarma, *Evolutionary bioinformatics online*, 2016, **12**, 109-119.
25. B. R. Miller and A. M. Gulick, *Methods Mol Biol*, 2016, **1401**, 3-29.
26. E. J. Drake, B. R. Miller, C. Shi, J. T. Tarrasch, J. A. Sundlov, C. L. Allen, G. Skiniotis, C. C. Aldrich and A. M. Gulick, *Nature*, 2016, **529**, 235-238.
27. J. Beld, E. C. Sonnenschein, C. R. Vickery, J. P. Noel and M. D. Burkart, *Nat Prod Rep*, 2014, **31**, 61-108.

28. D. Keszenman-Pereyra, S. Lawrence, M. E. Twfieg, J. Price and G. Turner, *Curr Genet*, 2003, **43**, 186-190.
29. T. Kittila, A. Mollo, L. K. Charkoudian and M. J. Cryle, *Angewandte Chemie*, 2016, **55**, 9834-9840.
30. M. Strieker, A. Tanovic and M. A. Marahiel, *Current opinion in structural biology*, 2010, **20**, 234-240.
31. C. Rausch, T. Weber, O. Kohlbacher, W. Wohlleben and D. H. Huson, *Nucleic Acids Res*, 2005, **33**, 5799-5808.
32. M. A. Fischbach and C. T. Walsh, *Chem Rev*, 2006, **106**, 3468-3496.
33. S. A. Sieber and M. A. Marahiel, *Chem Rev*, 2005, **105**, 715-738.
34. C. T. Walsh, H. W. Chen, T. A. Keating, B. K. Hubbard, H. C. Losey, L. S. Luo, C. G. Marshall, D. A. Miller and H. M. Patel, *Current opinion in chemical biology*, 2001, **5**, 525-534.
35. K. Bloudoff and T. M. Schmeing, *Bba-Proteins Proteom*, 2017, **1865**, 1587-1604.
36. E. A. Felnagle, E. E. Jackson, Y. A. Chan, A. M. Podevels, A. D. Berti, M. D. McMahon and M. G. Thomas, *Mol Pharmaceut*, 2008, **5**, 191-211.
37. P. A. Fawcett, J. J. Usher, J. A. Huddleston, R. C. Bleaney, J. J. Nisbet and E. P. Abraham, *Biochem J*, 1976, **157**, 651-660.
38. H. D. Mootz and M. A. Marahiel, *J Bacteriol*, 1997, **179**, 6843-6850.
39. K. M. Hoyer, C. Mahlert and M. A. Marahiel, *Chem Biol*, 2007, **14**, 13-22.
40. A. Lawen and R. Zocher, *The Journal of biological chemistry*, 1990, **265**, 11355-11360.
41. R. A. Cacho, W. Jiang, Y. H. Chooi, C. T. Walsh and Y. Tang, *Journal of the American Chemical Society*, 2012, **134**, 16781-16790.
42. X. Gao, S. W. Haynes, B. D. Ames, P. Wang, L. P. Vien, C. T. Walsh and Y. Tang, *Nature chemical biology*, 2012, **8**, 823-830.
43. J. Zhang, N. Liu, R. A. Cacho, Z. Gong, Z. Liu, W. Qin, C. Tang, Y. Tang and J. Zhou, *Nature chemical biology*, 2016, **12**, 1001-1003.
44. Y. Gao, R. B. Honzatko and R. J. Peters, *Natural Product Reports*, 2012, **29**, 1153-1175.
45. D. W. Christianson, *Chemical Reviews*, 2017, **117**, 11570-11648.
46. J. I. M. Rajaonarivony, J. Gershenzon and R. Croteau, *Arch Biochem Biophys*, 1992, **296**, 49-57.
47. R. Croteau, R. E. Ketchum, R. M. Long, R. Kaspera and M. R. Wildung, *Phytochem Rev*, 2006, **5**, 75-97.
48. D. K. Liscombe, G. V. Louie and J. P. Noel, *Nat Prod Rep*, 2012, **29**, 1238-1250.
49. R. V. K. Cochrane and J. C. Vederas, *Accounts Chem Res*, 2014, **47**, 3148-3161.
50. M. C. Tang, Y. Zou, K. Watanabe, C. T. Walsh and Y. Tang, *Chem Rev*, 2017, **117**, 5226-5333.
51. T. Vogt and P. Jones, *Trends Plant Sci*, 2000, **5**, 380-386.
52. P. Wang, X. Gao and Y. Tang, *Current opinion in chemical biology*, 2012, **16**, 362-369.
53. Y. Q. Cheng, J. M. Coughlin, S. K. Lim and B. Shen, *Method Enzymol*, 2009, **459**, 165-186.
54. C. T. Walsh, *Nature chemical biology*, 2015, **11**, 620-624.

55. J. R. Cashman, *Biochem Bioph Res Co*, 2005, **338**, 599-604.
56. L. F. Wu, S. Meng and G. L. Tang, *Bba-Proteins Proteom*, 2016, **1864**, 453-470.
57. Y. Matsuda, T. Awakawa, T. Wakimoto and I. Abe, *Journal of the American Chemical Society*, 2013, **135**, 10962-10965.
58. Q. W. Yang and S. H. Sze, *Genome Res*, 2008, **18**, 949-956.
59. S. Ballouz, A. R. Francis, R. T. Lan and M. M. Tanaka, *Plos Comput Biol*, 2010, **6**.
60. K. K. Sharma and H. N. Arst, *Curr Genet*, 1985, **9**, 299-304.
61. N. P. Keller and T. M. Hohn, *Fungal Genet Biol*, 1997, **21**, 17-29.
62. Y. Yin, M. H. Cai, X. S. Zhou, Z. Y. Li and Y. X. Zhang, *Appl Microbiol Biot*, 2016, **100**, 7787-7798.
63. L. Hang, N. Liu and Y. Tang, *Acs Catal*, 2016, **6**, 5935-5945.
64. A. A. Zhgun, M. V. Dumina, T. M. Voinova, V. V. Dzhavakhiya and M. A. Eldarov, *Appl Biochem Micro+*, 2018, **54**, 188-197.
65. W. Tao, M. H. Lee, J. Wu, N. H. Kim, J. C. Kim, E. Chung, E. C. Hwang and S. W. Lee, *Appl Environ Microbiol*, 2012, **78**, 6295-6301.
66. W. V. Shaw, *Crc Cr Rev Bioch Mol*, 1983, **14**, 1-46.
67. Y. Yan, N. Liu and Y. Tang, *Nat Prod Rep*, 2020, DOI: 10.1039/c9np00050j.
68. K. H. Almabruk, L. K. Dinh and B. Philmus, *ACS Chem Biol*, 2018, **13**, 1426-1437.
69. E. C. O'Neill, M. Schorn, C. B. Larson and N. Millan-Aguinaga, *Crit Rev Microbiol*, 2019, **45**, 255-277.
70. C. R. Hutchinson, J. Kennedy, C. Park, S. Kendrew, K. Auclair and J. Vederas, *Anton Leeuw Int J G*, 2000, **78**, 287-295.
71. C. D. Campbell and J. C. Vederas, *Biopolymers*, 2010, **93**, 755-763.
72. C. f. D. C. a. Prevention, 2019.
73. G. V. Research, 2018.
74. A. Butts and D. J. Krysan, *Plos Pathog*, 2012, **8**.
75. B. DiDomenico, *Curr Opin Microbiol*, 1999, **2**, 509-515.
76. L. Alcazar-Fuoli, E. Mellado, G. Garcia-Effron, J. R. Lopez, J. O. Grimalt, J. M. Cuenca-Estrella and J. L. Rodriguez-Tudela, *Steroids*, 2008, **73**, 339-347.
77. F. Karst and F. Lacroute, *Mol Gen Genet*, 1977, **154**, 269-277.
78. S. T. Behmer, N. Olszewski, J. Sebastiani, S. Palka, G. Sparacino, E. Sciarrno and R. J. Grebenok, *Front Plant Sci*, 2013, **4**.
79. H. Schaller, *Prog Lipid Res*, 2003, **42**, 163-175.
80. W. de Souza and J. C. Rodrigues, *Interdiscip Perspect Infect Dis*, 2009, **2009**, 642502.
81. H. K. Saini, A. S. Arneja and N. S. Dhalla, *Can J Cardiol*, 2004, **20**, 333-346.
82. M. Zavrel, B. D. Esquivel and T. C. White, in *Handbook of Antimicrobial Resistance*, eds. A. Berghuis, G. Matlashewski, M. A. Wainberg and D. Sheppard, Springer New York, New York, NY, 2017, DOI: 10.1007/978-1-4939-0694-9_29, pp. 423-452.
83. W. D. Nes, *Chemical Reviews*, 2011, **111**, 6423-6451.
84. P. D. Sonawane, J. Pollier, S. Panda, J. Szymanski, H. Massalha, M. Yona, T. Unger, S. Malitsky, P. Arendt, L. Pauwels, E. Almekias-Siegl, I. Rogachev, S.

Meir, P. D. Cardenas, A. Masri, M. Petrikov, H. Schaller, A. A. Schaffer, A. Kamble, A. P. Giri, A. Goossens and A. Aharoni, *Nat Plants*, 2016, **3**, 16205.
85. R. A. Moreau, B. D. Whitaker and K. B. Hicks, *Prog Lipid Res*, 2002, **41**, 457-500.
86. C. W. Roberts, R. McLeod, D. W. Rice, M. Ginger, M. L. Chance and L. J. Goad, *Mol Biochem Parasitol*, 2003, **126**, 129-142.
87. P. Benveniste, *Annu Rev Plant Phys*, 1986, **37**, 275-308.
88. I. Buhaescu and H. Izzedine, *Clin Biochem*, 2007, **40**, 575-584.
89. W. Eisenreich, A. Bacher, D. Arigoni and F. Rohdich, *Cell Mol Life Sci*, 2004, **61**, 1401-1426.
90. G. Popjak and J. W. Cornforth, *Adv Enzymol Rel S Bi*, 1960, **22**, 281-335.
91. E. J. Corey, S. P. T. Matsuda and B. Bartel, *P Natl Acad Sci USA*, 1994, **91**, 2211-2215.
92. J. X. Zhang, L. P. Li, Q. Z. Lv, L. Yan, Y. Wang and Y. Y. Jiang, *Front Microbiol*, 2019, **10**.
93. H. R. Waterham, *Febs Lett*, 2006, **580**, 5442-5449.
94. J. L. Gaylor, *Biochem Biophys Res Commun*, 2002, **292**, 1139-1146.
95. C. Fernandez, M. Martin, D. Gomez-Coronado and M. A. Lasuncion, *J Lipid Res*, 2005, **46**, 920-929.
96. A. Trenin, *Russ J Bioorg Chem*+, 2013, **39**, 565-587.
97. J. S. Burg and P. J. Espenshade, *Prog Lipid Res*, 2011, **50**, 403-410.
98. B. A. Stermer, G. M. Bianchini and K. L. Korth, *J Lipid Res*, 1994, **35**, 1133-1140.
99. G. A. Reynolds, S. K. Basu, T. F. Osborne, D. J. Chin, G. Gil, M. S. Brown, J. L. Goldstein and K. L. Luskey, *Cell*, 1984, **38**, 275-285.
100. J. A. Tobert, *Nature Reviews Drug Discovery*, 2003, **2**, 517-526.
101. S. M. Grundy, *New Engl J Med*, 1988, **319**, 24-33.
102. D. Dietrich and J. C. Vederas, *Fung Biol-Us*, 2014, DOI: 10.1007/978-1-4939-1191-2_12, 263-287.
103. E. Harunari, H. Komaki and Y. Igarashi, *Beilstein J Org Chem*, 2017, **13**, 441-450.
104. H. Tomoda, I. Namatame and S. Omura, *P Jpn Acad B-Phys*, 2002, **78**, 217-240.
105. A. Nadin and K. C. Nicolaou, *Angew Chem Int Edit*, 1996, **35**, 1623-1656.
106. J. C. Onishi, J. A. Milligan, A. Basilio, J. Bergstrom, J. Curotto, L. Huang, M. Meinz, M. NallinOmstead, F. Pelaez, D. Rew, M. Salvatore, J. Thompson, F. Vicente and M. B. Kurtz, *J Antibiot*, 1997, **50**, 334-338.
107. S. M. Mandala, R. A. Thornton, B. R. Frommer, S. Dreikorn and M. B. Kurtz, *J Antibiot*, 1997, **50**, 339-343.
108. H. Sasaki, T. Hosokawa, M. Sawada and K. Ando, *J Antibiot*, 1973, **26**, 676-680.
109. N. S. Ryder and H. Mieth, *Curr Top Med Mycol*, 1992, **4**, 158-188.
110. J. M. Muhlbacher, *Clin Dermatol*, 1991, **9**, 479-485.
111. J. E. Birnbaum, *J Am Acad Dermatol*, 1990, **23**, 782-785.
112. S. Krishnan-Natesan, *Expert Opin Pharmaco*, 2009, **10**, 2723-2733.
113. G. Lepesheva, T. Hargrove, Y. Kleshchenko, W. Nes, F. Villalta and M. Waterman, *Lipids*, 2008, **43**, 1117-1125.
114. D. I. Zonios and J. E. Bennett, *Semin Resp Crit Care*, 2008, **29**, 198-210.
115. J. M. Hamilton-Miller, *Bacteriol Rev*, 1973, **37**, 166-196.

116. N. A. O'Leary, M. W. Wright, J. R. Brister, S. Ciufo, D. H. R. McVeigh, B. Rajput, B. Robbertse, B. Smith-White, D. Ako-Adjei, A. Astashyn, A. Badretdin, Y. M. Bao, O. Blinkova, V. Brover, V. Chetvernin, J. Choi, E. Cox, O. Ermolaeva, C. M. Farrell, T. Goldfarb, T. Gupta, D. Haft, E. Hatcher, W. Hlavina, V. S. Joardar, V. K. Kodali, W. J. Li, D. Maglott, P. Masterson, K. M. McGarvey, M. R. Murphy, K. O'Neill, S. Pujar, S. H. Rangwala, D. Rausch, L. D. Riddick, C. Schoch, A. Shkeda, S. S. Storz, H. Z. Sun, F. Thibaud-Nissen, I. Tolstoy, R. E. Tully, A. R. Vatsan, C. Wallin, D. Webb, W. Wu, M. J. Landrum, A. Kimchi, T. Tatusova, M. DiCuccio, P. Kitts, T. D. Murphy and K. D. Pruitt, *Nucleic Acids Res*, 2016, **44**, D733-D745.
117. H. Nordberg, M. Cantor, S. Dusheyko, S. Hua, A. Poliakov, I. Shabalov, T. Smirnova, I. V. Grigoriev and I. Dubchak, *Nucleic Acids Res*, 2014, **42**, D26-D31.
118. S. McGinnis and T. L. Madden, *Nucleic Acids Res*, 2004, **32**, W20-W25.
119. M. Stanke and B. Morgenstern, *Nucleic Acids Res*, 2005, **33**, W465-W467.
120. M. Stanke and S. Waack, *Bioinformatics*, 2003, **19 Suppl 2**, ii215-225.
121. M. H. Medema, K. Blin, P. Cimermancic, V. de Jager, P. Zakrzewski, M. A. Fischbach, T. Weber, E. Takano and R. Breitling, *Nucleic Acids Res*, 2011, **39**, W339-W346.
122. N. Khaldi, F. T. Seifuddin, G. Turner, D. Haft, W. C. Nierman, K. H. Wolfe and N. D. Fedorova, *Fungal Genet Biol*, 2010, **47**, 736-741.
123. M. H. Medema, R. Kottmann, P. Yilmaz, M. Cummings, J. B. Biggins, K. Blin, I. de Bruijn, Y. H. Chooi, J. Claesen, R. C. Coates, P. Cruz-Morales, S. Duddela, S. Dusterhus, D. J. Edwards, D. P. Fewer, N. Garg, C. Geiger, J. P. Gomez-Escribano, A. Greule, M. Hadjithomas, A. S. Haines, E. J. N. Helfrich, M. L. Hillwig, K. Ishida, A. C. Jones, C. S. Jones, K. Jungmann, C. Kegler, H. U. Kim, P. Kotter, D. Krug, J. Masschelein, A. V. Melnik, S. M. Mantovani, E. A. Monroe, M. Moore, N. Moss, H. W. Nutzmann, G. H. Pan, A. Pati, D. Petras, F. J. Reen, F. Rosconi, Z. Rui, Z. H. Tian, N. J. Tobias, Y. Tsunematsu, P. Wiemann, E. Wyckoff, X. H. Yan, G. Yim, F. G. Yu, Y. C. Xie, B. Aigle, A. K. Apel, C. J. Balibar, E. P. Balskus, F. Barona-Gomez, A. Bechthold, H. B. Bode, R. Borriss, S. F. Brady, A. A. Brakhage, P. Caffrey, Y. Q. Cheng, J. Clardy, R. J. Cox, R. De Mot, S. Donadio, M. S. Donia, W. A. van der Donk, P. C. Dorrestein, S. Doyle, A. J. M. Driessen, M. Ehling-Schulz, K. D. Entian, M. A. Fischbach, L. Gerwick, W. H. Gerwick, H. Gross, B. Gust, C. Hertweck, M. Hofte, S. E. Jensen, J. H. Ju, L. Katz, L. Kaysser, J. L. Klassen, N. P. Keller, J. Kormanec, O. P. Kuipers, T. Kuzuyama, N. C. Kyrpides, H. J. Kwon, S. Lautru, R. Lavigne, C. Y. Lee, B. Linquan, X. Y. Liu, W. Liu, A. Luzhetskyy, T. Mahmud, Y. Mast, C. Mendez, M. Metsa-Ketela, J. Micklefield, D. A. Mitchell, B. S. Moore, L. M. Moreira, R. Muller, B. A. Neilan, M. Nett, J. Nielsen, F. O'Gara, H. Oikawa, A. Osbourn, M. S. Osburne, B. Ostash, S. M. Payne, J. L. Pernodet, M. Petricek, J. Piel, O. Ploux, J. M. Raaijmakers, J. A. Salas, E. K. Schmitt, B. Scott, R. F. Seipke, B. Shen, D. H. Sherman, K. Sivonen, M. J. Smanski, M. Sosio, E. Stegmann, R. D. Sussmuth, K. Tahlan, C. M. Thomas, Y. Tang, A. W. Truman, M. Viaud, J. D. Walton, C. T. Walsh, T. Weber, G. P. van Wezel, B. Wilkinson, J. M. Willey, W. Wohlleben, G. D. Wright, N. Ziemert, C. S. Zhang, S. B. Zotchev, R. Breitling, E. Takano and F. O. Glockner, *Nature chemical biology*, 2015, **11**, 625-631.

124. M. Alanjary, B. Kronmiller, M. Adamek, K. Blin, T. Weber, D. Huson, B. Philmus and N. Ziemert, *Nucleic Acids Res*, 2017, **45**, W42-W48.
125. J. Yaegashi, M. B. Praseuth, S. W. Tyan, J. F. Sanchez, R. Entwistle, Y. M. Chiang, B. R. Oakley and C. C. C. Wang, *Org Lett*, 2013, **15**, 2862-2865.
126. K. Ishiuchi, T. Nakazawa, F. Yagishita, T. Mino, H. Noguchi, K. Hotta and K. Watanabe, *Journal of the American Chemical Society*, 2013, **135**, 7371-7377.
127. Y. M. Chiang, C. E. Oakley, M. Ahuja, R. Entwistle, A. Schultz, S. L. Chang, C. T. Sung, C. C. C. Wang and B. R. Oakley, *Journal of the American Chemical Society*, 2013, **135**, 7720-7731.
128. H. S. Huang, N. T. Hu, Y. E. Yao, C. Y. Wu, S. W. Chiang and C. N. Sun, *Insect Biochem Mol Biol*, 1998, **28**, 651-658.
129. M. R. Farkhutdinov, F. G. Galiullin, E. G. Davletov, F. Gabbasov Sh and B. N. Safin, *Vopr Med Khim*, 1993, **39**, 60-62.
130. L. J. Trueman, *Methods Mol Biol*, 1995, **49**, 341-354.
131. S. M. Ma, J. W. Li, J. W. Choi, H. Zhou, K. K. Lee, V. A. Moorthie, X. Xie, J. T. Kealey, N. A. Da Silva, J. C. Vederas and Y. Tang, *Science*, 2009, **326**, 589-592.
132. A. Aleksenko and A. J. Clutterbuck, *Mol Microbiol*, 1996, **19**, 565-574.
133. J. H. Yu and T. J. Leonard, *J Bacteriol*, 1995, **177**, 4792-4800.
134. C. S. Nodvig, J. B. Nielsen, M. E. Kogle and U. H. Mortensen, *Plos One*, 2015, **10**, e0133085.
135. T. G. Montague, J. M. Cruz, J. A. Gagnon, G. M. Church and E. Valen, *Nucleic Acids Res*, 2014, **42**, W401-407.
136. J. D. Bergstrom, M. M. Kurtz, D. J. Rew, A. M. Amend, J. D. Karkas, R. G. Bostedor, V. S. Bansal, C. Dufresne, F. L. VanMiddlesworth, O. D. Hensens and et al., *Proc Natl Acad Sci U S A*, 1993, **90**, 80-84.
137. A. Armstrong, P. A. Barsanti, L. H. Jones and G. Ahmed, *J Org Chem*, 2000, **65**, 7020-7032.
138. J. O. Bunte, A. N. Cuzzupe, A. M. Daly and M. A. Rizzacasa, *Angewandte Chemie-International Edition*, 2006, **45**, 6376-6380.
139. S. Caron, D. Stoermer, A. K. Mapp and C. H. Heathcock, *Journal of Organic Chemistry*, 1996, **61**, 9126-9134.
140. E. M. Carreira and J. Dubois, *Journal of the American Chemical Society*, 1994, **116**, 10825-10826.
141. D. A. Evans, J. C. Barrow, J. L. Leighton, A. J. Robichaud and M. J. Sefkow, *Abstr Pap Am Chem S*, 1995, **209**, 49-Orgn.
142. K. D. Freeman-Cook and R. L. Halcomb, *Journal of Organic Chemistry*, 2000, **65**, 6153-6159.
143. T. Kawamata, M. Nagatomo and M. Inoue, *Journal of the American Chemical Society*, 2017, **139**, 1814-1817.
144. S. Nakamura, H. Sato, Y. Hirata, N. Watanabe and S. Hashimoto, *Tetrahedron*, 2005, **61**, 11078-11106.
145. D. A. Nicewicz, A. D. Satterfield, D. C. Schmitt and J. S. Johnson, *Journal of the American Chemical Society*, 2008, **130**, 17281-+.
146. K. C. Nicolaou, E. W. Yue, S. Lagreca, A. Nadin, Z. Yang, J. E. Leresche, T. Tsuri, Y. Naniwa and F. Dericcardis, *Chem-Eur J*, 1995, **1**, 467-494.

147. D. Stoermer, S. Caron and C. H. Heathcock, *Journal of Organic Chemistry*, 1996, **61**, 9115-9125.
148. K. Tomooka, M. Kikuchi, K. Igawa, M. Suzuki, P. H. Keong and T. Nakai, *Angewandte Chemie-International Edition*, 2000, **39**, 4502-+.
149. F. Y. Lim and N. P. Keller, *Nat Prod Rep*, 2014, **31**, 1277-1286.
150. Y. H. Chooi and Y. Tang, *J Org Chem*, 2012, **77**, 9933-9953.
151. R. J. Cox and T. J. Simpson, *Method Enzymol*, 2009, **459**, 49-78.
152. R. J. Cox, *Organic & Biomolecular Chemistry*, 2007, **5**, 2010-2026.
153. K. Scherlach, D. Boettger, N. Remme and C. Hertweck, *Natural Product Reports*, 2010, **27**, 869-886.
154. R. Cox, *Nat Prod Rep*, 2014, **31**, 1405-1424.
155. J. D. Bergstrom, C. Dufresne, G. F. Bills, M. Nallinomstead and K. Byrne, *Annu Rev Microbiol*, 1995, **49**, 607-639.
156. K. M. Byrne, B. H. Arison, M. Nallinomstead and L. Kaplan, *Journal of Organic Chemistry*, 1993, **58**, 1019-1024.
157. C. I. Liu, W. Y. Jeng, W. J. Chang, T. P. Ko and A. H. Wang, *The Journal of biological chemistry*, 2012, **287**, 18750-18757.
158. B. Bonsch, V. Belt, C. Bartel, N. Duensing, M. Koziol, C. M. Lazarus, A. M. Bailey, T. J. Simpson and R. J. Cox, *Chem Commun*, 2016, **52**, 6777-6780.
159. G. F. Bills, F. Pelaez, J. D. Polishook, M. T. Diezmatas, G. H. Harris, W. H. Clapp, C. Dufresne, K. M. Byrne, M. Nallinomstead, R. G. Jenkins, M. Mojena, L. Y. Huang and J. D. Bergstrom, *Mycol Res*, 1994, **98**, 733-739.
160. R. J. Cox, F. Glod, D. Hurley, C. M. Lazarus, T. P. Nicholson, B. A. M. Rudd, T. J. Simpson, B. Wilkinson and Y. Zhang, *Chem Commun*, 2004, DOI: 10.1039/b411973h, 2260-2261.
161. B. S. Moore and C. Hertweck, *Natural Product Reports*, 2002, **19**, 70-99.
162. L. K. Xiang and B. S. Moore, *J Bacteriol*, 2003, **185**, 399-404.
163. R. Fujii, Y. Matsu, A. Minami, S. Nagamine, I. Takeuchi, K. Gomi and H. Oikawa, *Org Lett*, 2015, **17**, 5658-5661.
164. K. Williams, A. J. Szwalbe, N. P. Mulholland, J. L. Vincent, A. M. Bailey, C. L. Willis, T. J. Simpson and R. J. Cox, *Angewandte Chemie-International Edition*, 2016, **55**, 6783-6787.
165. J. Bai, D. Yan, T. Zhang, Y. Guo, Y. Liu, Y. Zou, M. Tang, B. Liu, Q. Wu, S. Yu, Y. Tang and Y. Hu, *Angewandte Chemie*, 2017, **56**, 4782-4786.
166. M. Sato, F. Yagishita, T. Mino, N. Uchiyama, A. Patel, Y. H. Chooi, Y. Goda, W. Xu, H. Noguchi, T. Yamamoto, K. Hotta, K. N. Houk, Y. Tang and K. Watanabe, *Chembiochem*, 2015, **16**, 2294-2298.
167. G. H. Harris, C. Dufresne, H. Joshua, L. A. Koch, D. L. Zink, P. M. Salmon, K. E. Goklen, M. M. Kurtz, D. J. Rew, J. D. Bergstrom and K. E. Wilson, *Bioorg Med Chem Lett*, 1995, **5**, 2403-2408.
168. A. V. Qualley, J. R. Widhalm, F. Adebesin, C. M. Kish and N. Dudareva, *P Natl Acad Sci USA*, 2012, **109**, 16383-16388.
169. J. Piel, C. Hertweck, P. R. Shipley, D. M. Hunt, M. S. Newman and B. S. Moore, *Chem Biol*, 2000, **7**, 943-955.
170. W. Xu, Y. H. Chooi, J. W. Choi, S. Li, J. C. Vederas, N. A. Da Silva and Y. Tang, *Angewandte Chemie*, 2013, **52**, 6472-6475.

171. 1993.
172. A. M. Bolger, M. Lohse and B. Usadel, *Bioinformatics*, 2014, **30**, 2114-2120.
173. A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, A. V. Pyshkin, A. V. Sirotkin, N. Vyahhi, G. Tesler, M. A. Alekseyev and P. A. Pevzner, *J Comput Biol*, 2012, **19**, 455-477.
174. M. Stanke, R. Steinkamp, S. Waack and B. Morgenstern, *Nucleic Acids Res*, 2004, **32**, W309-312.
175. T. Weber, K. Blin, S. Duddela, D. Krug, H. U. Kim, R. Bruccoleri, S. Y. Lee, M. A. Fischbach, R. Muller, W. Wohlleben, R. Breitling, E. Takano and M. H. Medema, *Nucleic Acids Res*, 2015, **43**, W237-W243.
176. X. Yu, F. Liu, Y. Zou, M. C. Tang, L. Hang, K. N. Houk and Y. Tang, *Journal of the American Chemical Society*, 2016, DOI: 10.1021/jacs.6b09464.
177. J. L. R. Tudela and D. W. Denning, *Lancet Infect Dis*, 2017, **17**, 1111-1113.
178. *Nat Microbiol*, 2017, **2**.
179. P. T. McKeny and P. M. Zito, in *StatPearls*, Treasure Island (FL), 2020.
180. G. I. Lepesheva and M. R. Waterman, *Bba-Gen Subjects*, 2007, **1770**, 467-477.
181. G. I. Lepesheva and M. R. Waterman, *Mol Cell Endocrinol*, 2004, **215**, 165-170.
182. S. Emami, P. Tavangar and M. Keighobadi, *Eur J Med Chem*, 2017, **135**, 241-259.
183. E. Mellado, T. M. Diaz-Guerra, M. Cuenca-Estrella and J. L. Rodriguez-Tudela, *J Clin Microbiol*, 2001, **39**, 2431-2438.
184. R. Becher and S. G. Wirsel, *Appl Microbiol Biotechnol*, 2012, **95**, 825-840.
185. R. A. Paul, S. M. Rudramurthy, J. F. Meis, J. W. Mouton and A. Chakrabarti, *Antimicrob Agents Ch*, 2015, **59**, 6615-6619.
186. T. D. Edlind, K. W. Henry, K. A. Metera and S. K. Katiyar, *Med Mycol*, 2001, **39**, 299-302.
187. L. Rodero, E. Mellado, A. C. Rodriguez, A. Salve, L. Guelfand, P. Cahn, M. Cuenca-Estrella, G. Davel and J. L. Rodriguez-Tudela, *Antimicrob Agents Ch*, 2003, **47**, 3653-3656.
188. J. Loffler, S. L. Kelly, H. Hebart, U. Schumacher, C. LassFlorl and H. Einsele, *Fems Microbiol Lett*, 1997, **151**, 263-268.
189. K. Scherlach, J. Schuemann, H. M. Dahse and C. Hertweck, *J Antibiot (Tokyo)*, 2010, **63**, 375-377.
190. J. Yaegashi, M. B. Praseuth, S. W. Tyan, J. F. Sanchez, R. Entwistle, Y. M. Chiang, B. R. Oakley and C. C. Wang, *Org Lett*, 2013, **15**, 2862-2865.
191. S. Chen, Z. Liu, Y. Liu, Y. Lu, L. He and Z. She, *Beilstein J Org Chem*, 2015, **11**, 1187-1193.
192. H. Zhou, X. Sun, N. Li, Q. Che, T. Zhu, Q. Gu and D. Li, *Org Lett*, 2016, **18**, 4670-4673.
193. O. D. Hensens, C. F. Wichmann, J. M. Liesch, F. L. Vanmiddlesworth, K. E. Wilson and R. E. Schwartz, *Tetrahedron*, 1991, **47**, 3915-3924.
194. R. E. Schwartz, C. Dufresne, J. E. Flor, A. J. Kempf, K. E. Wilson, T. Lam, J. Onishi, J. Milligan, R. A. Fromtling, G. K. Abruzzo, R. Jenkins, K. Glazomitsky, G. Bills, L. Zitano, S. M. Delval and M. N. Omstead, *J Antibiot*, 1991, **44**, 463-471.

195. S. Matsukuma, T. Ohtsuka, H. Kotaki, H. Shirai, T. Sano, K. Watanabe, N. Nakayama, Y. Itezono, M. Fujiu, N. Shimma, K. Yokose and T. Okuda, *J Antibiot*, 1992, **45**, 151-159.
196. S. Matsukuma, T. Ohtsuka, H. Kotaki, H. Shirai, T. Sano, K. Watanabe, N. Nakayama, Y. Itezono, M. Fujiu, N. Shimma and et al., *J Antibiot (Tokyo)*, 1992, **45**, 151-159.
197. Y. Aoki, T. Yamazaki, M. Kondoh, Y. Sudoh, N. Nakayama, Y. Sekine, H. Shimada and M. Arisawa, *J Antibiot*, 1992, **45**, 160-170.
198. Y. Aoki, F. Yoshihara, M. Kondoh, Y. Nakamura, N. Nakayama and M. Arisawa, *Antimicrob Agents Ch*, 1993, **37**, 2662-2667.
199. S. Jendrzejewski and P. Ermann, *Tetrahedron Lett*, 1993, **34**, 615-618.
200. T. Honda, A. Satoh, T. Yamada, T. Hayakawa and K. Kanai, *J Chem Soc Perk T 1*, 1998, DOI: DOI 10.1039/a707534k, 397-405.
201. I. Paterson and T. Nowak, *Tetrahedron Lett*, 1996, **37**, 8243-8246.
202. X. F. Yu, V. Cojocaru, G. Mustafa, O. M. H. Salo-Ahen, G. I. Lepesheva and R. C. Wade, *J Mol Recognit*, 2015, **28**, 59-73.
203. H. J. Cools, C. Bayon, S. Atkins, J. A. Lucas and B. A. Fraaije, *Pest Manag Sci*, 2012, **68**, 1034-1040.
204. Z. Ma, T. J. Proffer, J. L. Jacobs and G. W. Sundin, *Appl Environ Microbiol*, 2006, **72**, 2581-2585.
205. L. A. Kelley, S. Mezulis, C. M. Yates, M. N. Wass and M. J. E. Sternberg, *Nat Protoc*, 2015, **10**, 845-858.
206. F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Z. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson and D. G. Higgins, *Mol Syst Biol*, 2011, **7**.
207. J. Yaegashi, B. R. Oakley and C. C. C. Wang, *J Ind Microbiol Biot*, 2014, **41**, 433-442.