

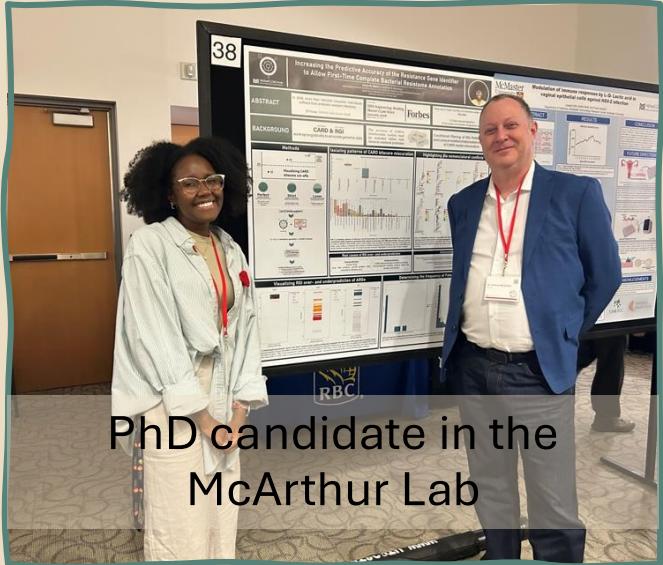
Karyn Mukiri

Predicting the Total Resistome

(hopefully)

2

tl;dr



3

... How did we get here?



B. Tech, Biotechnology (undergraduate)
Class of 2019



Quality Assurance Intern
Coca-Cola
(co-op, 2016)



Research Assistant
Dr. Zeinab Hosseinioudost
Chemical Engineering
(co-op, 2017)



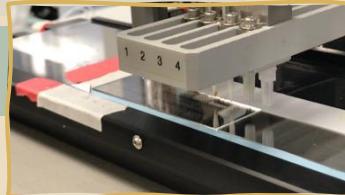
Lab Tech. Assistant
Biotech. Undergrad Labs
(\$\$\$, 2018)



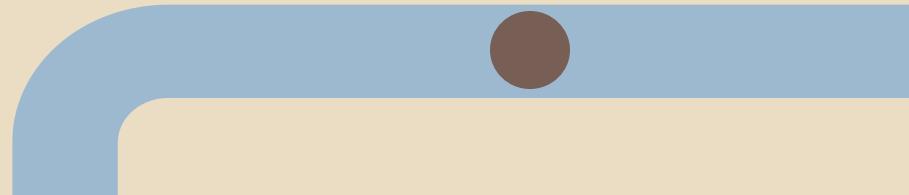
Biochemistry & Biomedical Sciences (grad. school)
2021



Capstone Project: *Biosynthesis of Silver Nanoparticles and Applications in Biosensor Technology*
("thesis", 2018)



Post-grad work
2019



4

... How did we get here?

B.Tech, Biotechnology (undergraduate)
Class of 2019



Biochemistry & Biomedical Sciences (grad. school)
2021



QA & Manufacturing Intern
Coca-Cola (2019)

Desperately Applying to Grad. School
2019-2021

Post-grad work
2019

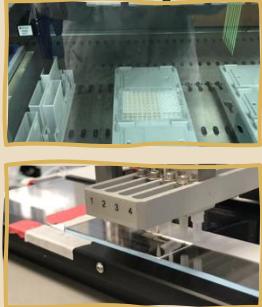
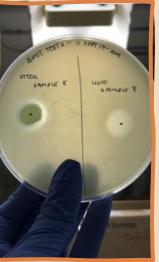
Coca-Cola



5

... How did we get here?

B. Tech, Biotechnology (undergraduate)
Class of 2019



Biochemistry & Biomedical Sciences (grad. school)
2021

MSc, McArthur Lab
Thesis: Predicting the Total Resistome
Transferred to the PhD program in Fall 2022



QA & Manufacturing Intern
Coca-Cola (2019)



Desperately Applying to Grad. School
2019-2021

Post-grad work
2019



Coca-Cola



The Comprehensive Antibiotic Resistance Database

A bioinformatic database of resistance genes, their products and associated phenotypes.

7170 Ontology Terms, 5194 Reference Sequences, 2008 SNPs, 3279 Publications, 5242 AMR Detection Models

Resistome predictions: 413 pathogens, 24291 chromosomes, 2662 genomic islands, 48212 plasmids, 172216 WGS assemblies, 276270 alleles

14 ontology terms | [Hide](#)

- + [cell wall synthesis enzyme targeted by antibiotic](#)
- + [cell membrane component targeted by antibiotic](#)
- + [protein synthesis machinery targeted by antibiotic](#)
- + [nucleic acid synthesis machinery targeted by antibiotic](#)
- + [nucleic acid targeted by antibiotic](#)
- + [folate synthesis enzyme targeted by antibiotic](#)
- + [isoprenoid synthesis enzyme targeted by antibiotic](#)

[Antibiotic target](#)

10 ontology terms | [Hide](#)

- + [antibiotic target alteration](#) [Resistance Mechanism]
- + [antibiotic target replacement](#) [Resistance Mechanism]
- + [antibiotic target protection](#) [Resistance Mechanism]
- + [antibiotic inactivation](#) [Resistance Mechanism]
- + [antibiotic efflux](#) [Resistance Mechanism]
- + [reduced permeability to antibiotic](#) [Resistance Mechanism]
- + [resistance by absence](#) [Resistance Mechanism]

[Mechanism of antibiotic resistance](#)

7

RGI Resistance Gene Identifier

Genome annotation tool

RGI can be used to predict resistomes from protein or nucleotide data based on homology and SNP models. Analyses can be performed via this web portal (20 Mb limit), via the command line, or via use of a [Galaxy wrapper](#). The command line version can be obtained from the [Download section of the CARD website](#). You can additionally install RGI from Conda or run RGI from Docker.

Enter a GenBank accession(s):

Enter accessions seperated by commas

Nucleotide sequences will undergo ORF calling to generate predicted protein sequences. Examples: JN420336.1, AY123251.1, HQ451074.1, AL123456

Upload FASTA sequence file(s):

No file chosen

Upload a [plain text file](#) containing DNA or protein sequence(s) in FASTA format (20 Mb limit). The file can contain more than one FASTA formatted sequence, such as assembly contigs or multiple proteins. Each file will be treated as a single sample.

Select Data Type:

DNA sequence

Protein sequence

Select Criteria:

Perfect and Strict hits only

Perfect, Strict and Loose hits

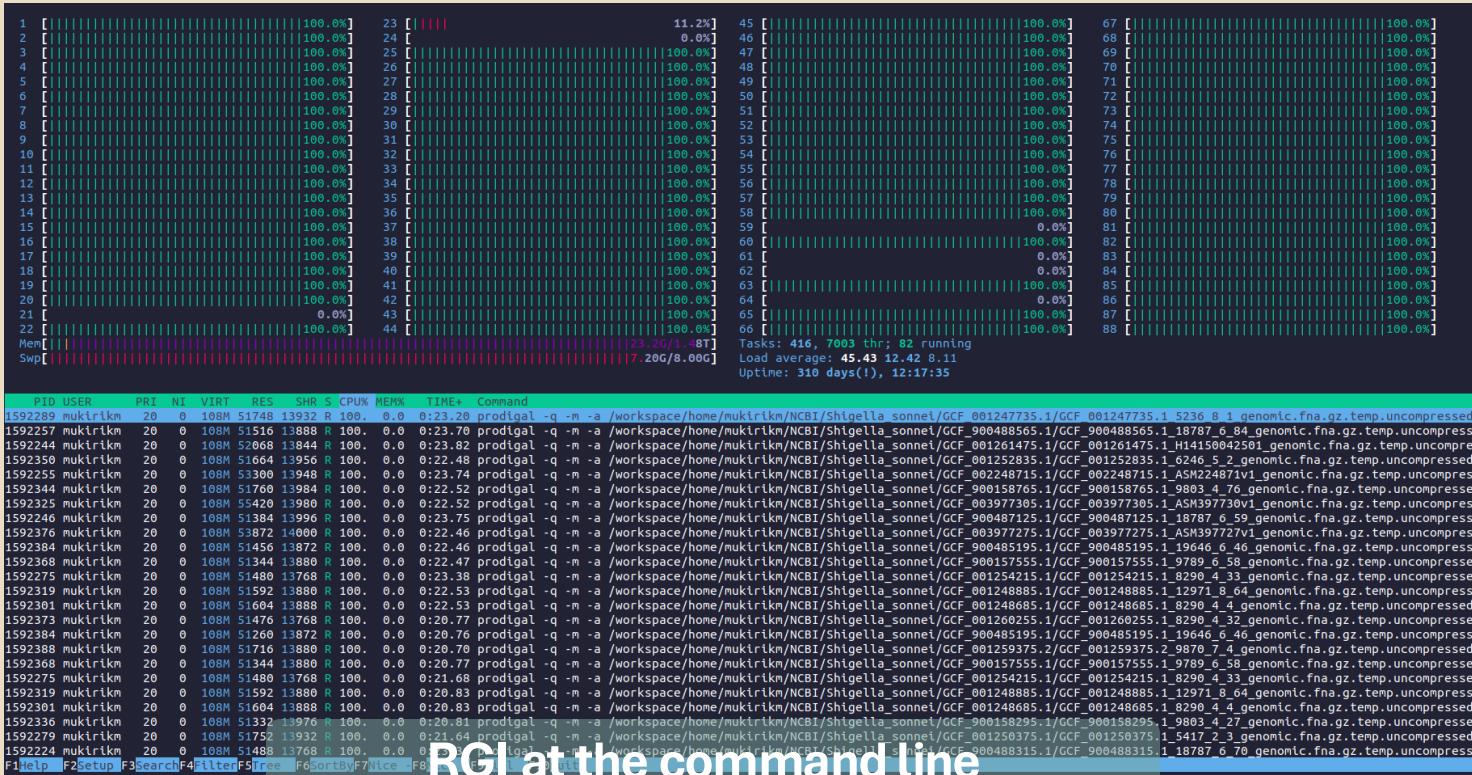
[RGI on the web](#)

7

RGI Resistance Gene Identifier

Genome annotation tool

RGI can be used to predict resistomes from protein or nucleotide data based on homology and SNP models. Analyses can be performed via this web portal (20 Mb limit), via the command line, or via use of a [Galaxy wrapper](#). The command line version can be obtained from the [Download section of the CARD website](#). You can additionally install RGI from Conda or run RGI from Docker.



RGI at the command line

8

RGI Resistance Gene Identifier

RGI can be used to predict resistomes from protein or nucleotide data based on homology and SNP models. Analyses can be performed via this web portal (20 Mb limit), via the command line, or via use of a [Galaxy wrapper](#). The command line version can be obtained from the [Download section of the CARD website](#). You can additionally install RGI from Conda or run RGI from Docker.

Genome annotation tool

```
>my_query
MRAHFRKYQLSQNAKDSAQLRAVGGKVFLSGDLARMVRPDAPEKDDGFNLIRAFQGLKF
SGPMFTGSKHSRADKTNEEPCRVIRRTDAQDTLLFSSQCSVCEISIGSMFSSCAGSVC
ARPHLCVRASCVRPLHQSSRSAGAARLFQEIPDTGSNGWMNLLRFWRDGRFSRMHKHME
ESFRRLLGPIYREHLGSQSSVNIMLPMDTGELFRSEGLHPRRMTLQPWATHRETRRHSGV
FLKNNGTEWRADRLLNREVMVSSSVHRFLPLLDEVAQDFCRSLRRRVQADGFEKAGQHTL
TLDPSPDLFRALEASCHVLYGERIGLFSSCPSPSERFISAVERMLATTPLLYLPPRL
LLRLRASLWTTHATAWDDIFSHAEQRQIQRSYQRLQARPSAAPDCSFPGVLGKLMEAGQLS
LELIRANITELMAGGVDTTAVPLQFALFELARNPDVQECVRAQVLSSWQQASGDPLKALQ
GAPLLKGTIKETRLYPVGITVQRYPVRDIVLQNYHVPAGTLVQVCLYPLGRSAEVFSRP
ECFDPSRWSADADAGSAGGFRSLAFGFGSRQCVRRIAENEMQLLMHILRTFKLTVSST
EELSTKYTLILQPECPPRITFSTLTHQH
```

RGI Criteria	ARO Term	SNP	Detection Criteria	AMR Gene Family	Drug Class	Resistance Mechanism	% Identity of Matching Region	% Length of Reference Sequence
Loose	tmrB		protein homolog model	tunicamycin resistance protein	nucleoside antibiotic	reduced permeability to antibiotic	34.25	91.37
Loose	DHA-7		protein homolog model	DHA beta-lactamase	cephalosporin, cephemycin	antibiotic inactivation	100.0	89.18
Loose	NmcR		protein homolog model	NmcA beta-lactamase	carbapenem, cephalosporin, cephemycin, penam	antibiotic inactivation	46.69	98.64
Perfect	TEM-1		protein homolog model	TEM beta-lactamase	monobactam, cephalosporin, penam, penem	antibiotic inactivation	100.0	100.00
Perfect	AAC(3)-IId		protein homolog model	AAC(3)	aminoglycoside antibiotic	antibiotic inactivation	100.0	100.00
Perfect	NDM-1		protein homolog model	NDM beta-lactamase	carbapenem, cephalosporin, cephemycin, penam	antibiotic inactivation	100.0	100.00

Sequenced-based detection models

protein homolog model	4,805
BLASTP bit-score	4,805
protein variant model	532
single resistance variant	188
BLASTP bit-score	188
multiple resistance variants	22
no association with resistance TB	17
deletion mutation from nucleotide sequence	17
nonsense mutation	16
high confidence TB	14
insertion mutation from nucleotide sequence	14
minimal confidence TB	9
moderate confidence TB	8
insertion mutation from peptide sequence	7
deletion mutation from peptide sequence	7
indeterminate confidence TB	7
co-dependent single resistance variant	6
frameshift mutation	5
snp in promoter region	3
co-dependent nonsense SNP	2
disruptive mutation in regulatory element	1
insertion mutation	1
protein knockout model	18
BLASTP bit-score	18
protein overexpression model	36
single resistance variant	15
BLASTP bit-score	15
multiple resistance variants	1
deletion mutation from peptide sequence	1
nonsense mutation	1
deletion mutation from nucleotide sequence	1
insertion mutation from nucleotide sequence	1
frameshift mutation	1
rRNA gene variant model	187
BLASTN bit-score	87
single resistance variant	86
high confidence TB	4
no association with resistance TB	3
moderate confidence TB	3
multiple resistance variants	2
disruptive mutation in regulatory element	1
minimal confidence TB	1

Meta-model detection models

efflux pump system meta-model	33
efflux pump components	33
gene cluster meta-model	14
gene order	14
protein domain meta-model	1
domain order	1

One big problem:
RGI can't detect every CARD gene or mutation because it's missing code

10

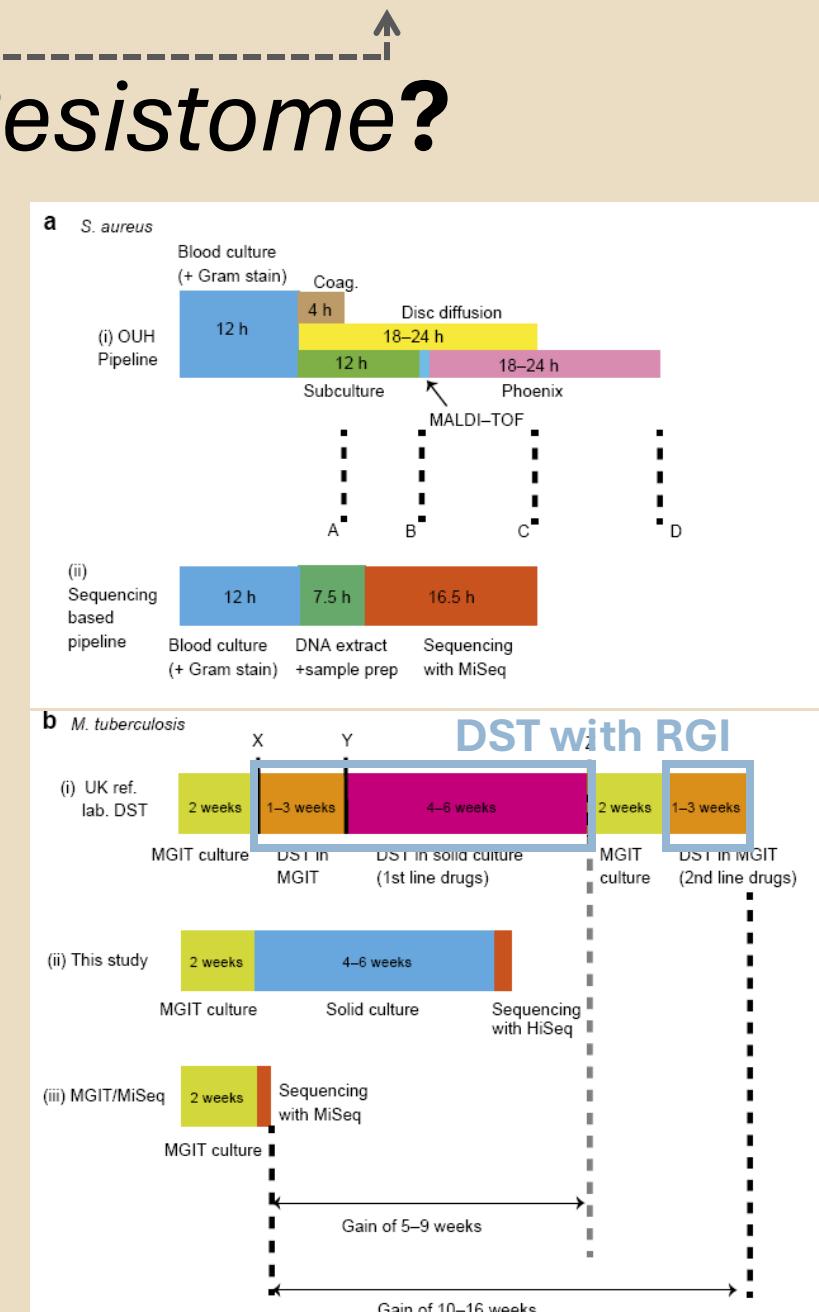
Why Predict the Total Resistome?

Being able to detect all AMR genes and mutations computationally gets us closer to phenotype prediction from genotype

A more complete resistome prediction can speed up diagnostic pipelines

Rapid antibiotic-resistance predictions from genome sequence data for *Staphylococcus aureus* and *Mycobacterium tuberculosis*

[Phelim Bradley](#), [N. Claire Gordon](#), [Timothy M. Walker](#), [Laura Dunn](#), [Simon Heys](#), [Bill Huang](#), [Sarah Earle](#), [Louise J. Pankhurst](#), [Luke Anson](#), [Mariateresa de Cesare](#), [Paolo Piazza](#), [Antonina A. Votintseva](#), [Tanya Golubchik](#), [Daniel J. Wilson](#), [David H. Wyllie](#), [Roland Diel](#), [Stefan Niemann](#), [Silke Feuerriegel](#), [Thomas A. Kohl](#), [Nazir Ismail](#), [Shaheed V. Omar](#), [E. Grace Smith](#), [David Buck](#), [Gil McVean](#), ... [Zamin Iqbal](#) ↗



10

Why Predict the Total Resistome?

Being able to detect all AMR genes and mutations computationally gets us closer to phenotype prediction from genotype

A more complete resistome prediction can speed up diagnostic pipelines

We'll be able to figure out which AMR genes are most prevalent epidemiologically



**So why isn't the code done,
Karyn?????**



12

Research isn't [that] linear

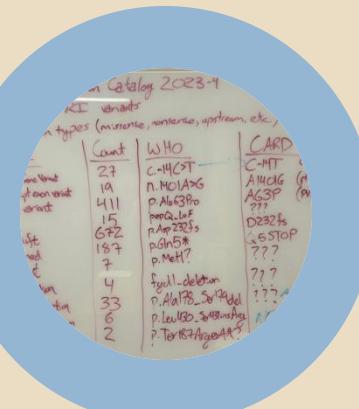
Step 1: Figure out how bad our curation is

- Bad curation leads to bad AMR predictions



Step 2: Fix bad curation and add more genes/mutations

- More data == a better look at the “resistome”



Step 3: Write the code

- Without the code, RGI is blind

```
def main():
    # Your code here
    pass

if __name__ == "__main__":
    main()
```

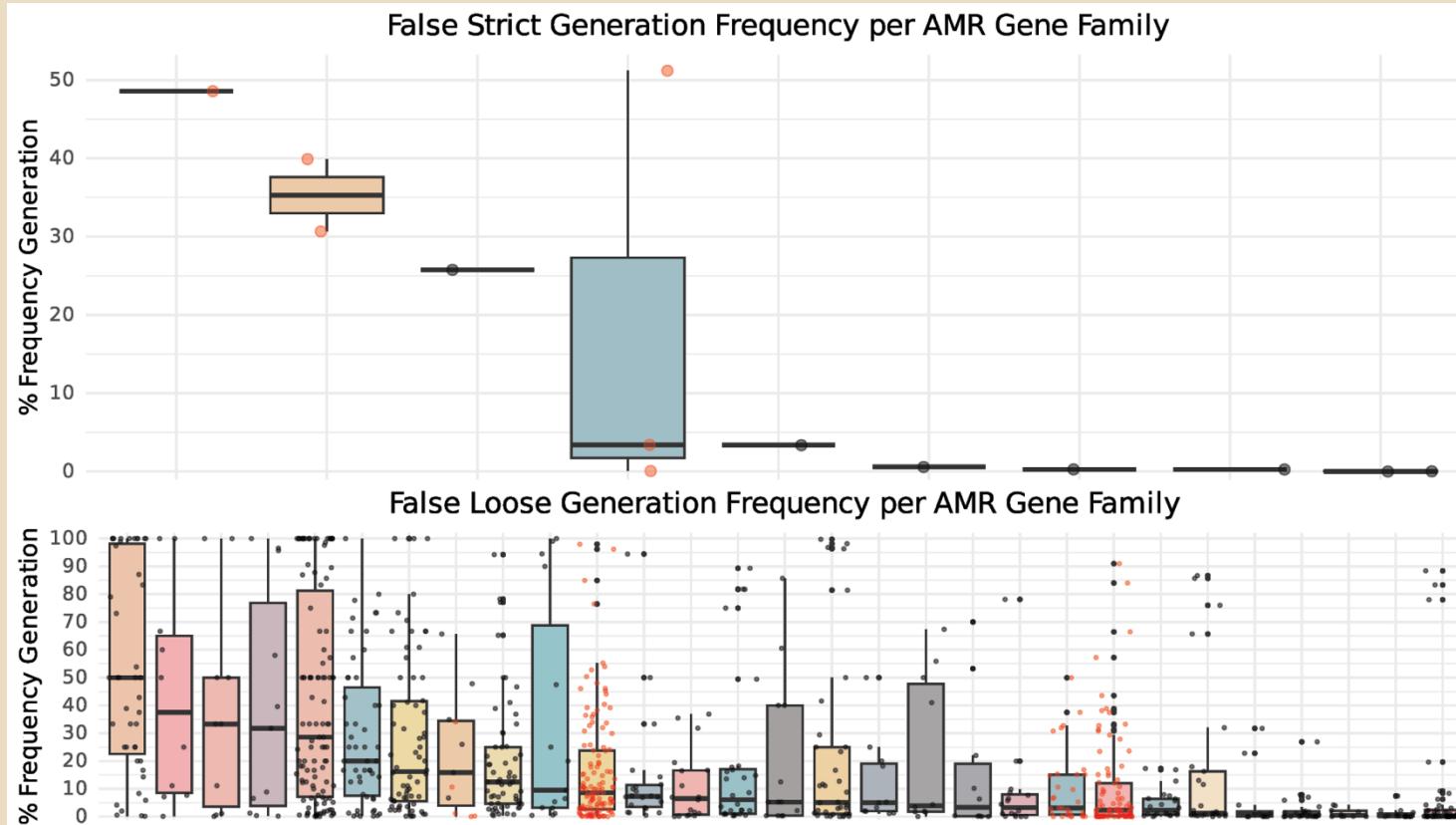
Step 4: Validate the code

- Did we get it right? How accurate are we? Did I break it?



13

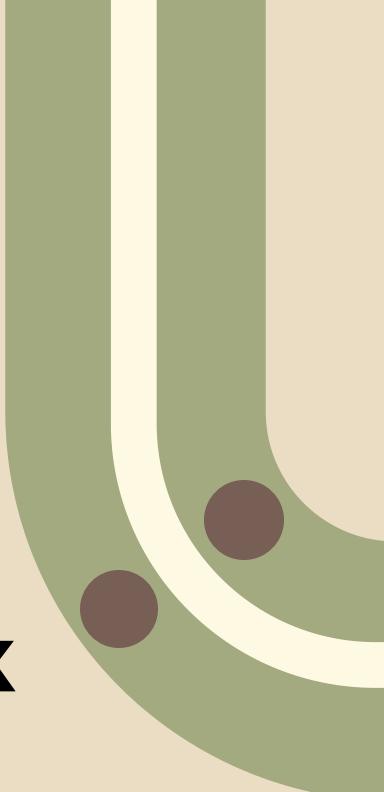
Step 1: Figure out how bad curation is



The answer: Not terrible!

Two findings:

1. RGI has a **very strict prediction algorithm**, so we rarely overpredict resistance
 - You can trust RGI-Perfect and RGI-Strict annotations
2. RGI may be **underpredicting resistance**
 - Mainly beta-lactamases! Terrible nomenclature...



**The last slide condensed 3 years of work
into one figure... Painful.**

File Edit Selection View Go Run Terminal Help

EXPLORER

UNTITLED (WORKSPACE)

- release-rgi
- github
- pytest_cache
- vscode
- app
- __pycache__
- _data
- _db
- diff
- __init__.py
- Analyse.py
- auto_load.py
- baits_annotation.py
- Baits.py
- Base.py
- Blast.py
- build_kmer_sets.py
- BWT.py
- card_annotation.py
- clean.py
- convert_card_json_to_gtf3.py
- ConvertRGIJsonToTSV.py
- Database.py
- Diamond.py
- filepaths.py
- Filter.py
- Galaxy.py
- get_grantham.py
- grantham_table.py
- Heatmap.py
- HomologModel.py 9+
- kmer_query.py
- load.py
- MainPanel.nv

OUTLINE

TIMELINE HomologModel.py

- Fix KeyError associated with overexpress... 11 mos
- File Saved
- Overhaul GS feature and add HSP rank ... k... 1 yr
- File Saved
- File Saved

SONARLINT ISSUE LOCATIONS

cardjson_cracker.py msa_prep.py HomologModel.py 9+

```
release-rgi > app > HomologModel.py >
1 from statistics import mean, median
2
3 from app.get_grantham import get_grantham
4 from app.Base import BaseModel
5 from app.settings import *
6
7
8 class Homolog(BaseModel):
9     """Class for homology searches."""
10    def __init__(self, input_type, loose, input_sequence, xml_file, working_directory, local_database=False, exclude_nudge=True):
11        self.input_type = input_type
12        self.loose = loose
13        self.input_sequence = input_sequence
14        self.xml_file = xml_file
15        self.output = {}
16        self.working_directory = working_directory
17
18        self.local_database = local_database
19        self.data = data_path
20
21        self.exclude_nudge = exclude_nudge
22
23        if self.local_database:
24            self.db = os.path.join(self.data, 'local_db')
25            self.data = os.path.join(self.data, 'local_db')
26
27        if self.exclude_nudge:
28            self.exclude_nudge_db = os.path.join(self.data, 'exclude_nudge_db')
29
30    def __repr__(self):
31        """Returns Homolog class full object."""
32        return "Homolog({})".format(self.__dict__)
33
34    def run(self):
35        """Runs homolog search."""
36        blastResults = {}
37        # print(json.dumps(self.__dict__, indent=2))
38        predicted_genes_dict = {}
39        predicted_genes_dict_protein = {}
```

TERMINAL OUTPUT PROBLEMS DEBUG CONSOLE PORTS

```
release-rgi ) ls -lt
total 38824
-rw-rw-r-- 1 mukirikm mukirikm 534614 Mar 22 2023 detective.work.ods
-rw-rw-r-- 1 mukirikm mukirikm 1174335 Feb 19 2023 GCF_010475225.1_ASM1847522v1_genomic.fna.gz
-rw-rw-r-- 1 mukirikm mukirikm 1230310 Feb 19 2023 old_GCF.fna.gz
-rw-rw-r-- 1 mukirikm mukirikm 1070145 Feb 12 2023 detective.work.txt
-rw-rw-r-- 1 mukirikm mukirikm 35564071 Feb 12 2023 detective.work.json
-rwxrwxr-x 1 mukirikm mukirikm 38 Feb 2 2023 ncardr_run.sh
-rwxrwxr-x 1 mukirikm mukirikm 291 Jan 2 2023 piper_ncardr.sh
drwxrwxr-x 2 mukirikm mukirikm 4096 Jan 25 2023 DIAMOND_results
drwxrwxr-x 2 mukirikm mukirikm 4096 Jan 25 2023 DIAMOND_files
drwxrwxr-x 2 mukirikm mukirikm 4096 Jan 24 2023 BLAST_results
```

R: (not attached) Ln 1, Col 1 Tab Size: 4 UTF-8 LF Python 3.11.3 (python3.11.3:conda)

Modifying the RGI code

HomologModel.py - Untitled (Workspace) - V

File Edit Selection View Go Run Terminal Help

EXPLORER

- UNTITLED (WORKSPACE)
 - the-master-files
 - release-rgi
 - github
 - pytests_cache
 - vscode
 - app
 - __pycache__
 - _data
 - _db
 - diff
 - __init__.py
 - Analyse.py
 - auto_load.py
 - baits_annotation.py
 - Baits.py
 - Base.py
 - Blast.py
 - build_kmer_sets.py
 - BWT.py
 - card_annotation.py
 - clean.py
 - convert_card_json_to_gtf3.py
 - ConvertRGIToTSV.py
 - Database.py
 - Diamond.py
 - filepaths.py
 - Filter.py
 - Galaxy.py
 - get_grantham.py
 - grantham_table.py
 - Heatmap.py
 - HomologModel.py 9+
 - kmer_query.py
 - load.py
 - MainPanel.mw
 - OUTLINE
 - TIMELINE HomologModel.py
 - Fix KeyError associated with overexpress... 11 mos
 - File Saved
 - Overhaul GS feature and add HSP rank ... k... 1 yr
 - File Saved
 - File Saved
 - File Saved
 - SONARLINT ISSUE LOCATIONS

cardjson_cracker.py msa_prep.py HomologModel.py 9+

Open in Browser

Plotter Title TBA Karyn M. Mukiri Model Dot Plots Chaos Plots Bitscore Plots Histogram & Table

ARO hit name adeF

Model dot plot (% identity)

adeF

File model count: 1295

Hit Type

- Strict (Orange)
- Loose (Grey)
- False Strict (Red)

% Identity

protein homolog model

Detection Model Type

Model Type: protein homolog model
% Identity: 59.03
% Positives: 75.45
% Length of Ref Seq: 99.15

Model Type: protein homolog model
% Identity: 44.42
% Positives: 64.01
% Length of Ref Seq: 99.15

Model Type: protein homolog model
% Identity: 26.44
% Positives: 43.15
% Length of Ref Seq: 59.11

Model dot plot (% positives)

adeF

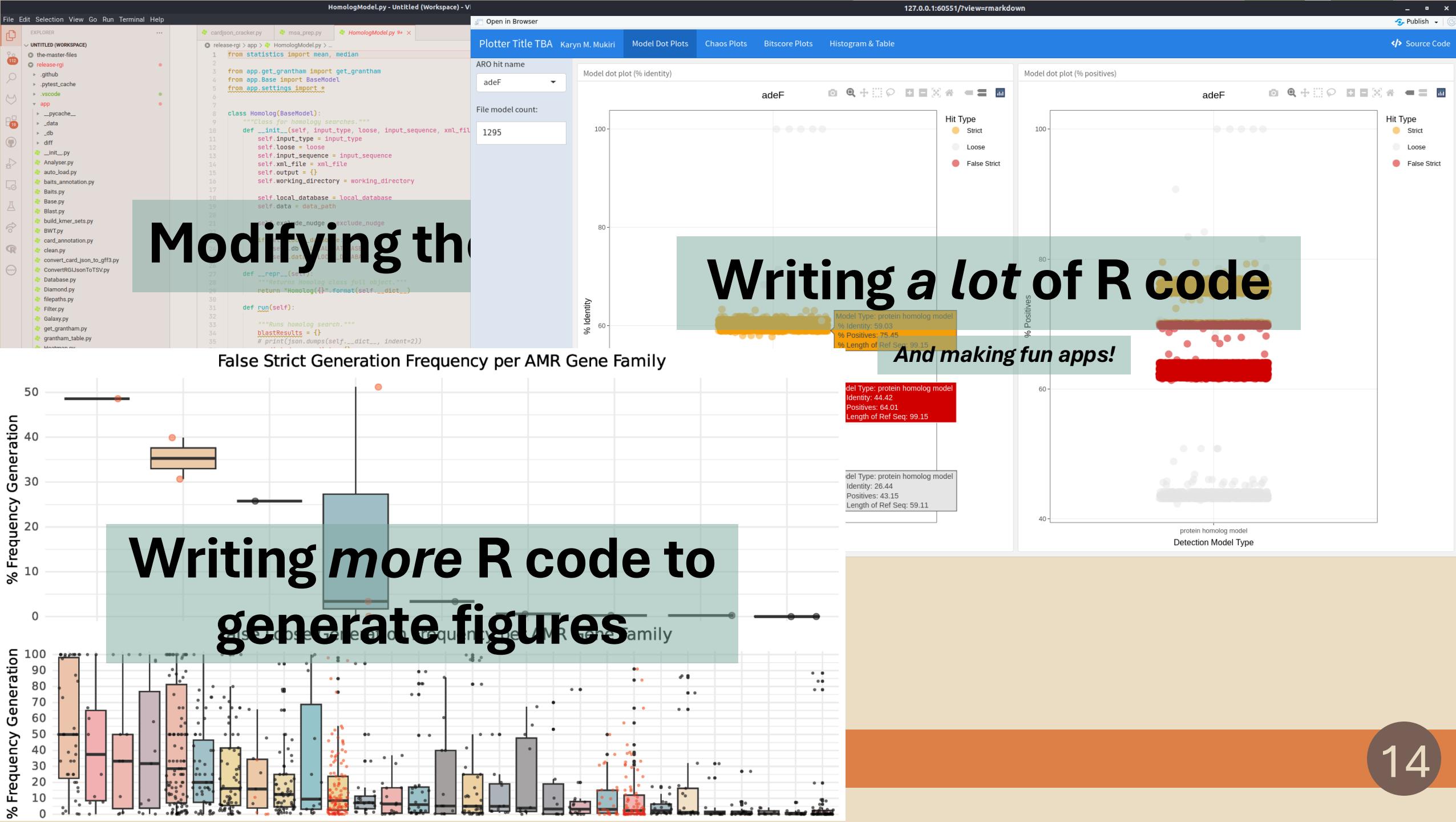
Hit Type

- Strict (Orange)
- Loose (Grey)
- False Strict (Red)

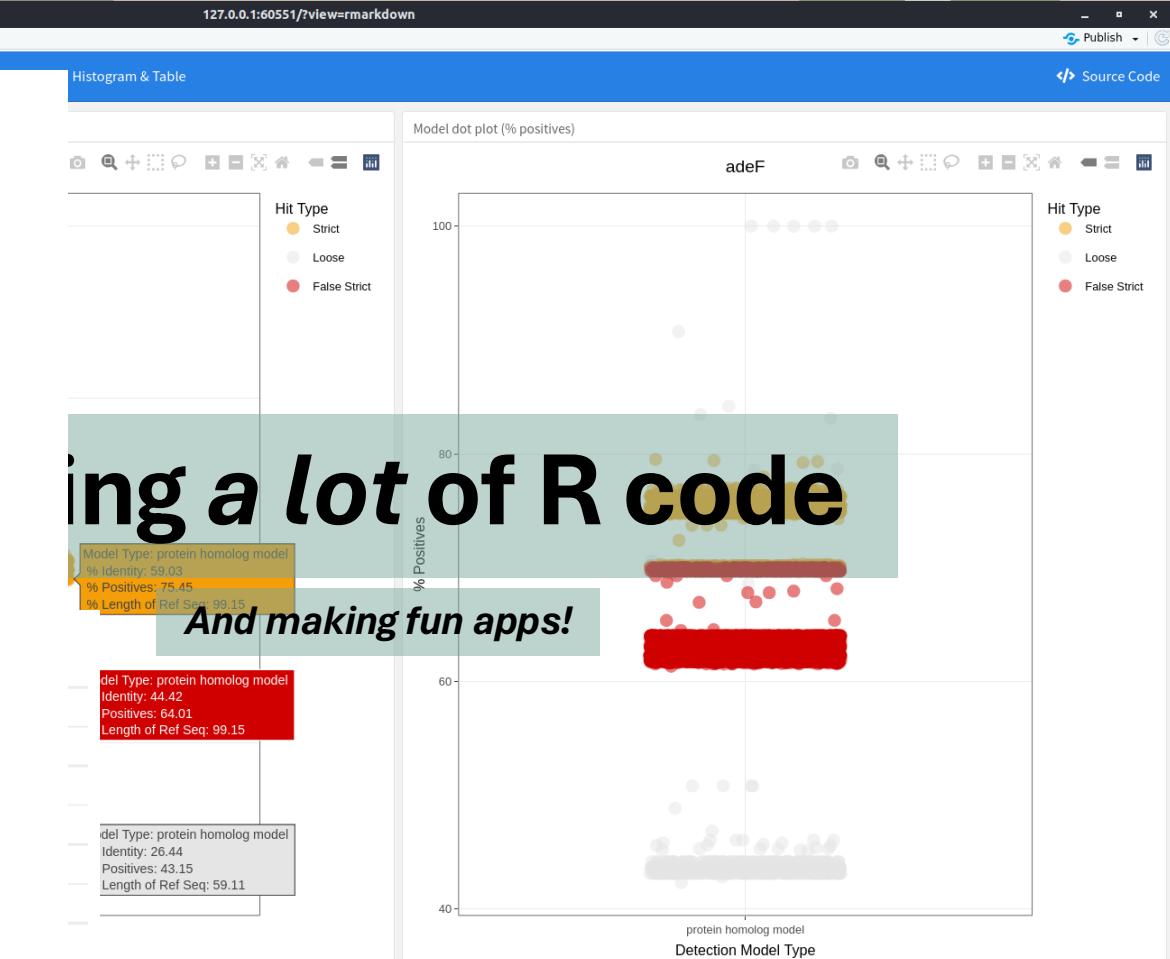
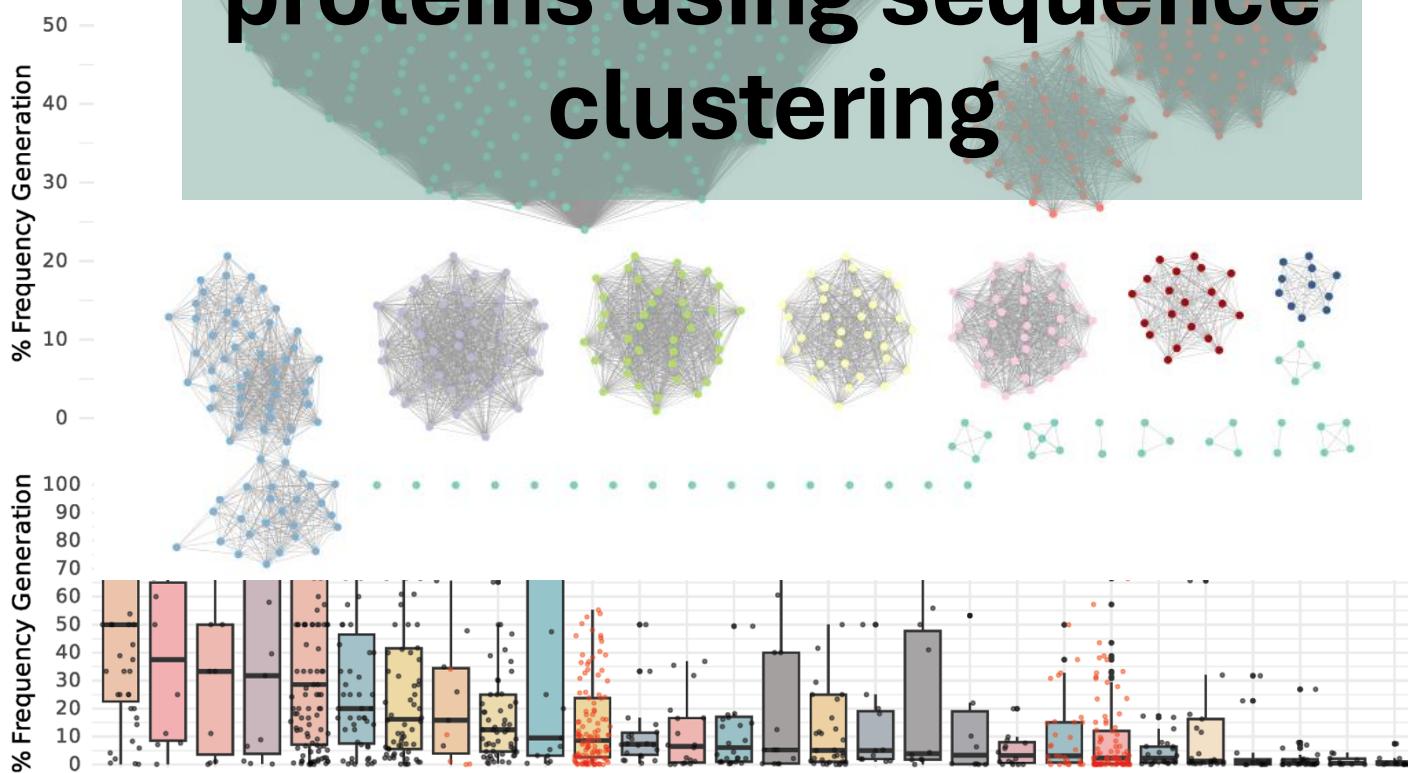
% Positives

protein homolog model

And making fun apps!

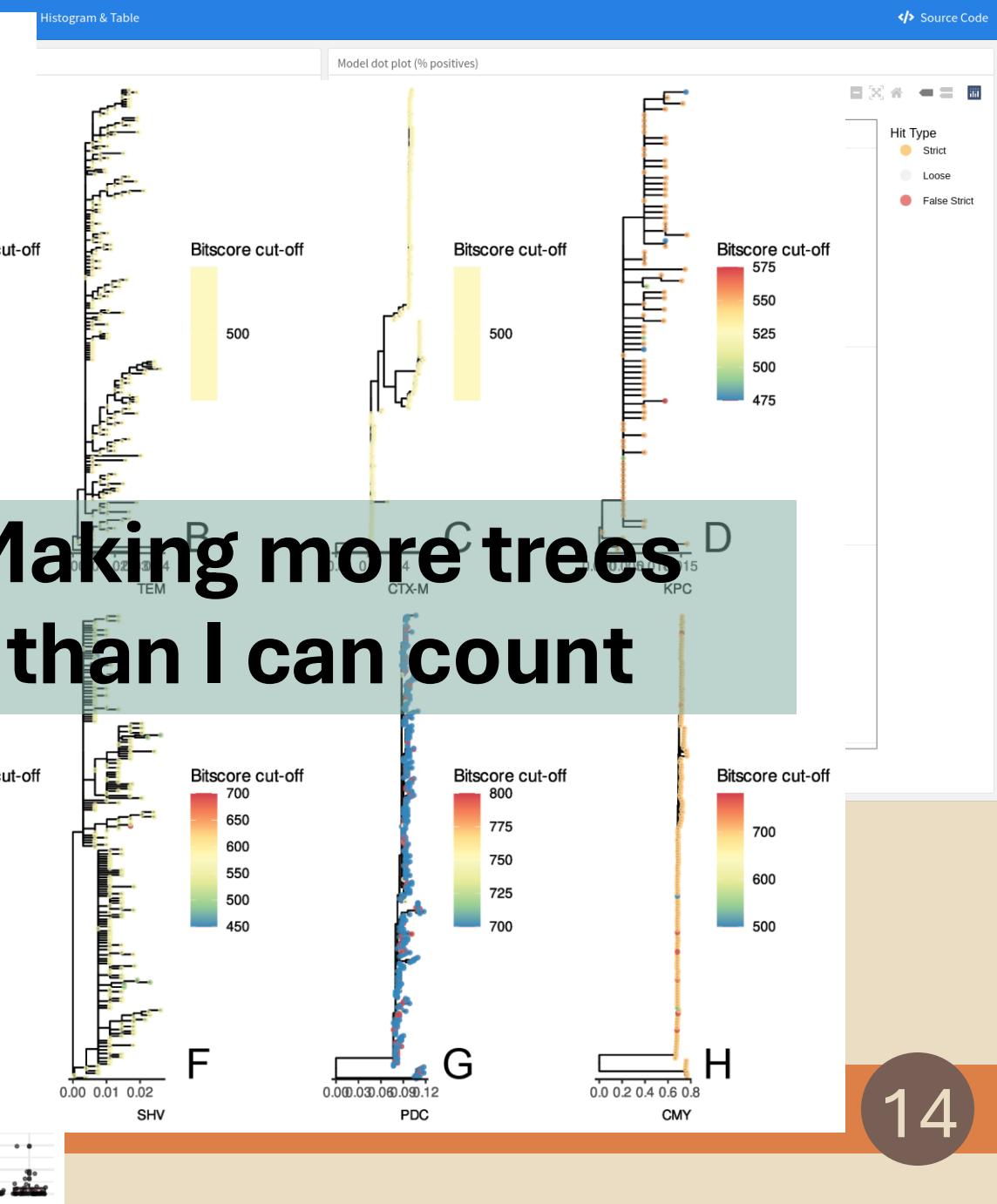
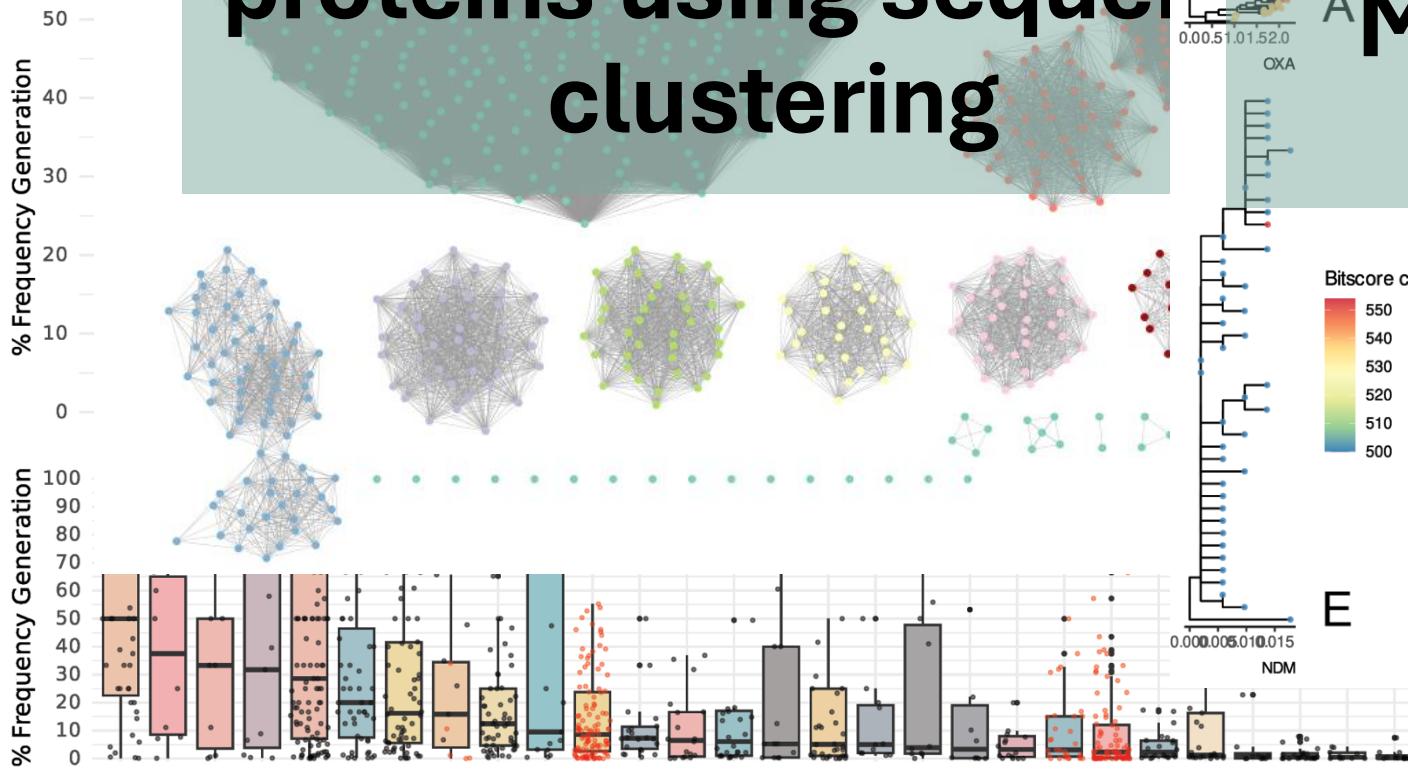


Trying to make sense of proteins using sequence clustering



Trying to make sense of proteins using sequence clustering

Making more trees than I can count



A circular portrait of a man with long, dark, wavy hair and a full beard. He is wearing dark-rimmed glasses and a pinkish-red sweater over a collared shirt. The background is blurred, showing what appears to be a stone wall or building.

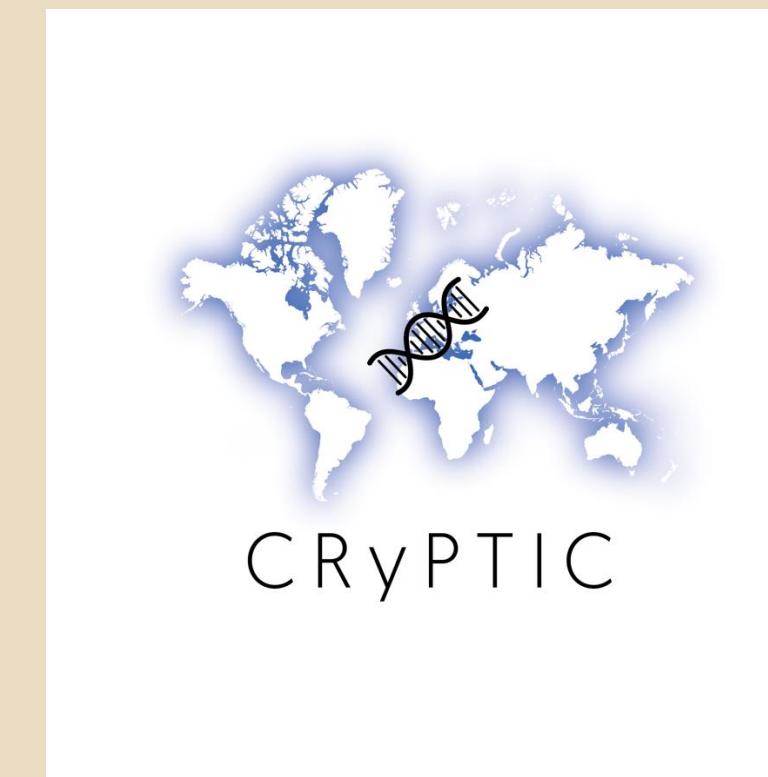
A circular portrait of a young woman with long, dark, wavy hair. She is smiling warmly at the camera. She is wearing a dark-colored top and a thin necklace with a small pendant. The background is a soft-focus green foliage.



A portrait of a woman with dark hair, smiling, overlaid with a circular frame containing the text "mutation from nucleotide".

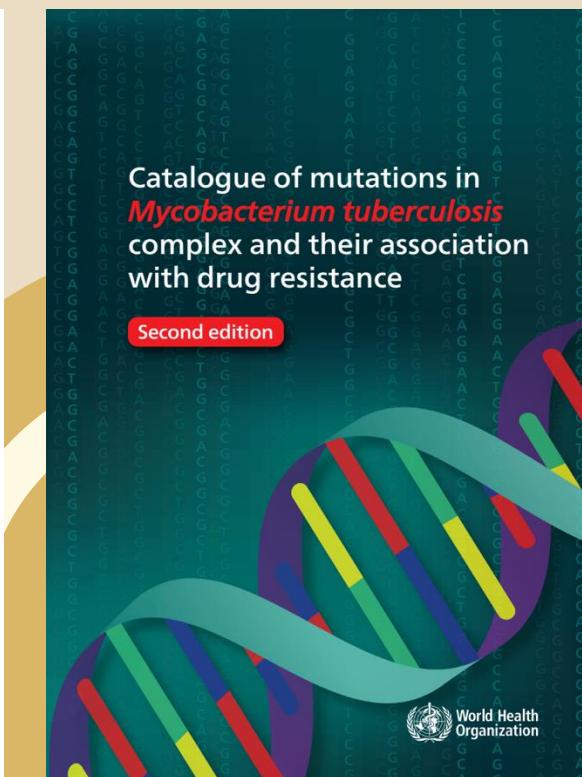


Step 2: Fix the old, add the new



Data from >30,000 MTB patient samples

30,000 MTB mutations



16

Step 3: Write the code (finally)

The screenshot shows a code editor interface with a dark theme. The left sidebar contains a file tree with numerous Python files under the 'rqi_total_resistome' directory, including VariantModel.py, Base.py, Database.py, and ConvertRGIJsonToTSV.py. The main editor area displays the VariantModel.py code, which defines a Variant class that inherits from BaseModel. The code handles protein and DNA sequence extraction, JSON parsing, and alignment processing. A SonarLint analysis is running, with a progress bar at the top indicating the status. On the right side, there is a vertical panel showing detailed SonarLint issue locations and a large, dense tree diagram representing the code's dependency graph.

```
VariantModel.py rqi5.1.1dev0 · app VariantModel.py Variant (f) run
VariantModel.py rqi_total_resistome > app > VariantModel.py Variant (f) run

class Variant(BaseModel):
    def run(self):
        predicted_genes_dict = self.get_orf_dna_sequence(self.input_sequence, self.input_type)
        predicted_genes_dict_protein = self.get_orf_protein_sequence(self.input_sequence, self.input_type)

        if self.input_type == "protein":
            submitted_proteins_dict = (self.get_submitted_protein_sequence(self.input_sequence))

        with open(os.path.join(self.data, "card.json")) as json_file:
            json_data = json.load(json_file)

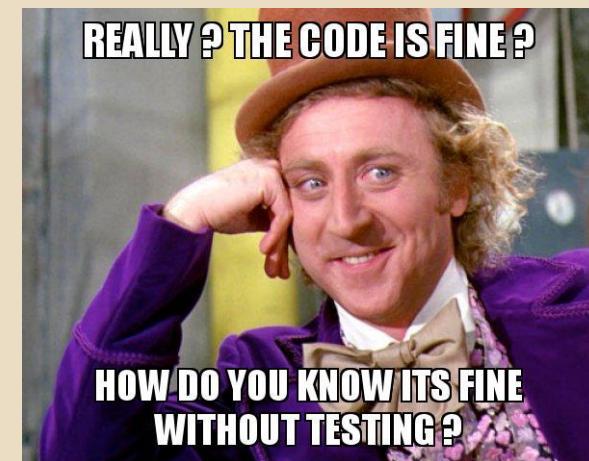
        with open(self.xml_file, 'r') as result_handle:
            blast_records = NCBIXML_parse(result_handle)
            for blast_record in blast_records:
                perfect = {}
                strict = {}
                loose = {}
                for alignment in blast_record.alignments:
                    align_title = alignment.title
                    orf_info = blast_record.query.encode('ascii', 'replace')

                    c = 0
                    barc = 0
                    for eachc in orf_info:
                        if barc >= 6:
                            break
                        elif eachc == '|':
                            barc += 1
                            c += 1
                        else:
                            c += 1
                    orf_from = orf_info[c:]

                    model_type_id = self.extract_nth_bar(align_title, 0)
                    # logger.info("model_type_id: {}".format(model_type_id))
                    space_pos = align_title.index(' ')
                    hit_id = align_title[0:space_pos]
                    hit_id = hit_id.encode('ascii', 'replace')
                    model_descript = align_title[align_title.index(' ') + 1:]
                    underscore_in_MD = model_descript.index('_')
                    model_id = model_descript[0:underscore_in_MD]
                    seq_in_model = model_descript[underscore_in_MD + 1: model_descript.index(' ')]
                    pass_value = self.extract_nth_bar(alignment.title, 1)
                    # logger.info("pass value: {}".format(pass_value))
```

Almost there...

- RGI is **production-level code**
 - All **object-oriented Python**
 - New modules need accompanying **testing code**



17

Step 4: Validate the code with data

Real datasets

```
>my_query
MRAHFRRKYQLSQNAKDSAQLRAVGGKVFSGDLARMVRPDAEKDDGFNLIRAFQGLKF
SGPMFTGSKHSRADKTNEEPCRVIRRTDAQDTLFFSSQCSVCIEISGMSFSCAGSVC
ARPHLCVRASCVRPLHQSSRSAGAARLFQEIPDTGSNGWMNLLRFWRDGRFSRMHKHME
ESFRRLGPIREHLGSQSSVNIMLPMDTGERLPRRMTLQPWATHRETRRHSGKV
FLKNGTEWRADRLNNREVMSVSSVHRLPLDEVAQDFCRSLRRVQADGFEKAQHTL
TLDPSDPLRFALEASCHVLYGERIGLFFSCPDESERFISAVERMATTPLLYLPPRL
LLRLRASLWTHATAWDIFSHAEQRQSYQRQARPSSAAPDCSFPGVLGKLMEAQQLS
LELIRANITELMAGGVDTAVPLQFALFELARNPDVQECVRAQVLSWWQASGDPLKALQ
GAPLLGTTKETLRLYPVGITVQRYPVRDILVQNYHVPAGTLVQVCLYPLGRSAEVFSRP
ECFDPRSWSADADAGSAGGRSLAFGFSRQCVGRRIAENEMQLLMHILRTFKLTVST
EELSTKYTLIQLQPECPPRITFSTLTHQ
```

```
>my_query
MRAHFRRKYQLSQNAKDSAQLRAVGGKVFSGDLARMVRPDAEKDDGFNLIRAFQGLKF
SGPMFTGSKHSRADKTNEEPCRVIRRTDAQDTLFFSSQCSVCIEISGMSFSCAGSVC
ARPHLCVRASCVRPLHQSSRSAGAARLFQEIPDTGSNGWMNLLRFWRDGRFSRMHKHME
ESFRRLGPIREHLGSQSSVNIMLPMDTGERLPRRMTLQPWATHRETRRHSGKV
FLKNGTEWRADRLNNREVMSVSSVHRLPLDEVAQDFCRSLRRVQADGFEKAQHTL
TLDPSDPLRFALEASCHVLYGERIGLFFSCPDESERFISAVERMATTPLLYLPPRL
LLRLRASLWTHATAWDIFSHAEQRQSYQRQARPSSAAPDCSFPGVLGKLMEAQQLS
LELIRANITELMAGGVDTAVPLQFALFELARNPDVQECVRAQVLSWWQASGDPLKALQ
GAPLLGTTKETLRLYPVGITVQRYPVRDILVQNYHVPAGTLVQVCLYPLGRSAEVFSRP
ECFDPRSWSADADAGSAGGRSLAFGFSRQCVGRRIAENEMQLLMHILRTFKLTVST
EELSTKYTLIQLQPECPPRITFSTLTHQ
```

```
>my_query
MRAHFRRKYQLSQNAKDSAQLRAVGGKVFSGDLARMVRPDAEKDDGFNLIRAFQGLKF
SGPMFTGSKHSRADKTNEEPCRVIRRTDAQDTLFFSSQCSVCIEISGMSFSCAGSVC
ARPHLCVRASCVRPLHQSSRSAGAARLFQEIPDTGSNGWMNLLRFWRDGRFSRMHKHME
ESFRRLGPIREHLGSQSSVNIMLPMDTGERLPRRMTLQPWATHRETRRHSGKV
FLKNGTEWRADRLNNREVMSVSSVHRLPLDEVAQDFCRSLRRVQADGFEKAQHTL
TLDPSDPLRFALEASCHVLYGERIGLFFSCPDESERFISAVERMATTPLLYLPPRL
LLRLRASLWTHATAWDIFSHAEQRQSYQRQARPSSAAPDCSFPGVLGKLMEAQQLS
LELIRANITELMAGGVDTAVPLQFALFELARNPDVQECVRAQVLSWWQASGDPLKALQ
GAPLLGTTKETLRLYPVGITVQRYPVRDILVQNYHVPAGTLVQVCLYPLGRSAEVFSRP
ECFDPRSWSADADAGSAGGRSLAFGFSRQCVGRRIAENEMQLLMHILRTFKLTVST
EELSTKYTLIQLQPECPPRITFSTLTHQ
```

Synthetic datasets

tseemann/injecta

Insert genes into genomes to aid synthetic test data generation



1
Contributor

1
Issue

4
Stars

0
Forks

