Course Outline

Simon Rayner

simon.rayner@medisin.uio.no

https://github.com/IzmirKCU/IKCU_1

Focus on

- Install Ubuntu for Windows
- Introduction to Python programming
- how to write clean and reusable Python code
- How to debug code and report errors
- How to maintain and collaborate on code
- How to document code
- How to keep an electronic lab book

Course files

You can download from github at

If you are familiar with github, you can pull it from

https://github.com/orgs/CBGOUS/repositories

Need java 11. Use SDKMAN to handle java versions (https://sdkman.io/)

```
curl -s "https://get.sdkman.io" | bash
```

Note: you need to restart terminal after installation (or just logout and in again)
or run

```
source "$HOME/.sdkman/bin/sdkman-init.sh”
```

To install Corretto 17:

```
sdk install java 17.0.6-amzn
```

To install Corretto 8:

```
sdk install java 8.0.442-amzn
```

hairpin.fa

From mirbase.org, a list of all hairpin sequences, for all species

Question 1

```
How many sequences in the file?
```

Question 2

```
How many species in the file?
```

Question 3

```
How many human sequences in the file?
```

Install Ubuntu on Windows

Install Ubuntu on Windows

Install Python

Install Ubuntu on Windows

Install Java (sdkman)

Install Java (sdkman)

Need java 11. Use SDKMAN to handle java versions (https://sdkman.io/)

```
curl -s "https://get.sdkman.io" | bash
```

Note: you need to restart terminal after installation (or just logout and in again)
or run

```
source "$HOME/.sdkman/bin/sdkman-init.sh”
```

To install Corretto 17:

```
sdk install java 17.0.6-amzn
```

To install Corretto 8:

```
sdk install java 8.0.442-amzn
```

Some basic Linux commands

pwd

ls

cd

wc –l

grep

sed

awk

find

xargs

piping commands using the ‘|’ character

Some basic Linux commands

Install Java (sdkman)

GC Calculation

Find the GC content of the fasta file `hairpin.fa`

cat hairpin.fa|more

```
>cel-let-7 MI0000001 Caenorhabditis elegans let-7 stem-loop
UACACUGUGGAUCCGGUGAGGUAGUAGGUUGUAUAGUUUGGAAUAUUACCACCGGUGAAC
UAUGCAAUUUUCUACCUUACCGGAGACAGAACUCUUCGA
>cel-lin-4 MI0000002 Caenorhabditis elegans lin-4 stem-loop
AUGCUUCCGGCCUGUUCCCUGAGACCUCAAGUGUGAGUGUACUAUUGAUGCUUCACACCU
GGGCUCUCCGGGUACCAGGACGGUUUGAGCAGAU
>cel-mir-1 MI0000003 Caenorhabditis elegans miR-1 stem-loop
AAAGUGACCGUACCGAGCUGCAUACUUCCUUACAUGCCCAUACUAUAUCAUAAAUGGAUA
UGGAAUGUAAAGAAGUAUGUAGAACGGGGUGGUAGU
>cel-mir-2 MI0000004 Caenorhabditis elegans miR-2 stem-loop
UAAACAGUAUACAGAAAGCCAUCAAAGCGGUGGUUGAUGUGUUGCAAAUUAUGACUUUCA
UAUCACAGCCAGCUUUGAUGUGCUGCCUGUUGCACUGU
```

Let’s modify the file so that we have one sequence/line

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);} END {printf("\n");}’ \
    < hairpin.fa > hairpin_flat.fa
```

cat hairpin_flat.fa|more

```
>cel-let-7 MI0000001 Caenorhabditis elegans let-7 stem-loop
UACACUGUGGAUCCGGUGAGGUAGUAGGUUGUAUAGUUUGGAAUAUUACCACCGGUGAACUAUGCAAUUUUCUACCUUACCGGAGACAGAACUCUUCGA
>cel-lin-4 MI0000002 Caenorhabditis elegans lin-4 stem-loop
AUGCUUCCGGCCUGUUCCCUGAGACCUCAAGUGUGAGUGUACUAUUGAUGCUUCACACCUGGGCUCUCCGGGUACCAGGACGGUUUGAGCAGAU
>cel-mir-1 MI0000003 Caenorhabditis elegans miR-1 stem-loop
AAAGUGACCGUACCGAGCUGCAUACUUCCUUACAUGCCCAUACUAUAUCAUAAAUGGAUAUGGAAUGUAAAGAAGUAUGUAGAACGGGGUGGUAGU
>cel-mir-2 MI0000004 Caenorhabditis elegans miR-2 stem-loop
UAAACAGUAUACAGAAAGCCAUCAAAGCGGUGGUUGAUGUGUUGCAAAUUAUGACUUUCAUAUCACAGCCAGCUUUGAUGUGCUGCCUGUUGCACUGU
```

Let’s modify the file so that we have one sequence/line

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);} END {printf("\n");}’ < hairpin.fa
```

We are going to work with the code and data in the folder
../BINF_M612/day1/GCCalculation/software

This folder contains two java files CalcGC.jar and GCCalc.jar

They both calculate average GC content for an input file of fasta sequences

We can find out how to run the program by typing

```
$ java -jar day1/GCCalculation/software/GCcalc.jar -h
```

```
GCCalc
initializing
parse arguments
   ==================================================================\
   | GCCalc  :                                           \
   |   Java code to calculate GC percentage in a FastA file             \
   ==================================================================\
usage: command line options
 -f,--sequence file <arg>   sequence file in FASTA format
 -h,--help                  view help
```

So, we just need to specify a fasta file. Let's start with the file data/GCtest.fa

(This corresponds to an alignment of sequences for the 3'UTR of the Lysine Methyltransferase 5B gene)

We’ve tried with a small test dataset. Let’s repeat with a real one.

```
$ java -jar day1/GCCalculation/software/GCCalc.jar
-f day1/GCCalculation/data/GCtest.fa
```

```
GCCalc
initializing
parse arguments
fasta input file is <day1/GCCalculation/data/ENSG00000110066___ENST00000441488___2___KMT5B__uniq_aln.fa>
read <16> sequences from file
average GC value of all sequences is <43.35%>
```

And repeat using CalcGC.jar

```
$ java -jar day1/GCCalculation/software/CalcGC.jar
-f day1/GCCalculation/data/GCtest.fa
```

```
CalcGC
initializing
parse arguments
fasta input file is <day1/GCCalculation/data/ENSG00000110066___ENST00000441488___2___KMT5B__uniq_aln.fa>
read <16> sequences from file
average GC value of all sequences is <43.35%>
```

So, two programs give the same results, which is encouraging

We’ve checked the programs using a simple test dataset.
Now let’s try running the programs against the hairpin.fa file

What do you find?

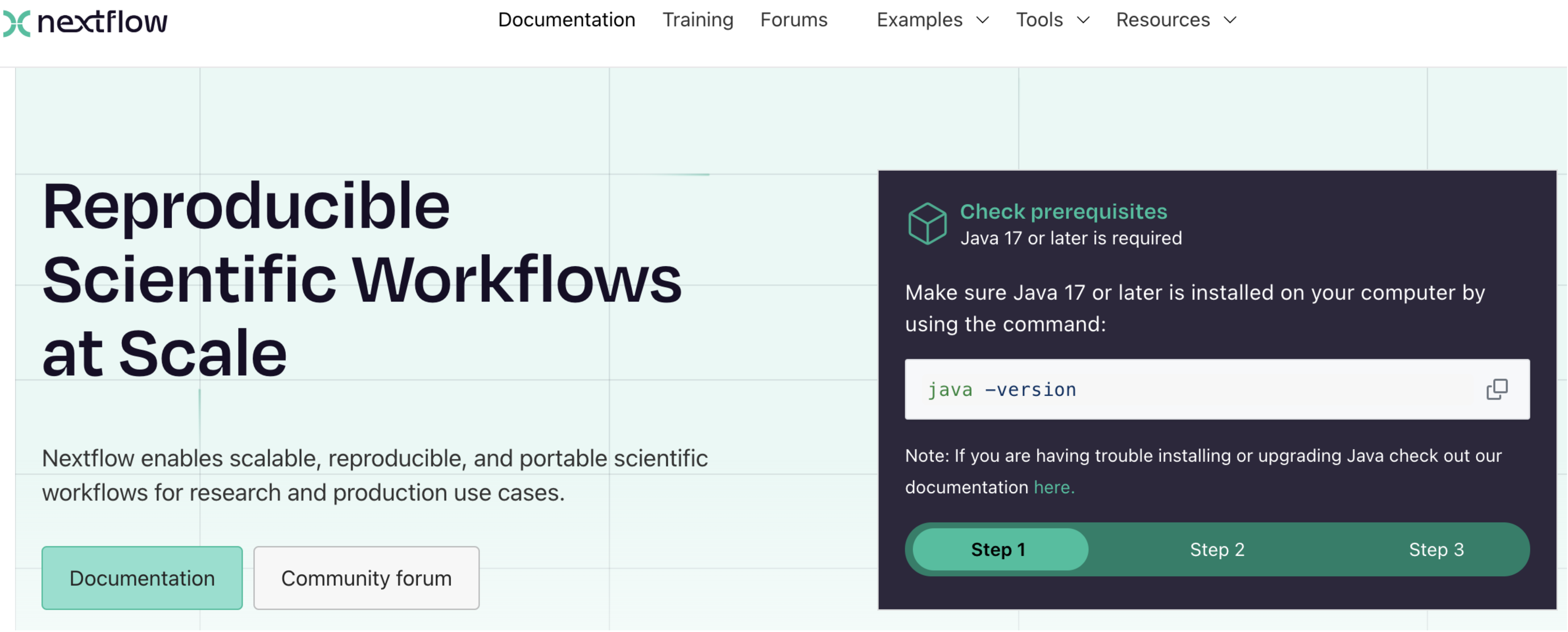
How can you figure out what is going on?

Data set is very large – so let’s use a simpler test set

Programming in Python

We started with calculating the average GC content of all the sequences in the hairpin.fa file using two different Java programs

We installed Java using SDKMAN
This allows us to run different versions of Java, which is quite handy if you are running programs you have downloaded from other sites



Reproducible Scientific Workflows at Scale

Nextflow enables scalable, reproducible, and portable scientific workflows for research and production use cases.

Check prerequisites

Java 17 or later is required

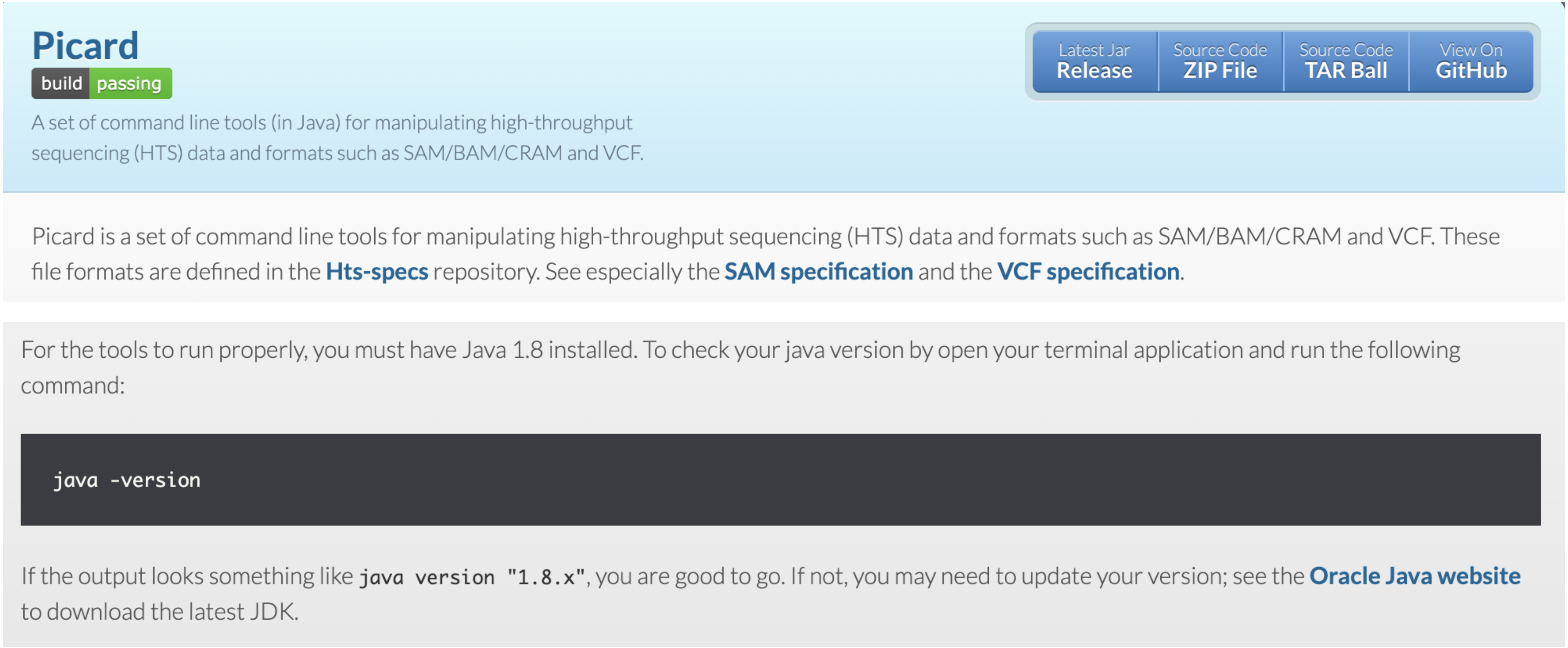
Make sure Java 17 or later is installed on your computer by using the command:

```
java -version
```

Note: If you are having trouble installing or upgrading Java check out our documentation here.

For example, Nextflow requires Java 17

But Picard, another popular tool only requires Java 8



Picard

A set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF.

Picard is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF. These file formats are defined in the **Hts-specs** repository. See especially the **SAM specification** and the **VCF specification**.

For the tools to run properly, you must have Java 1.8 installed. To check your java version by open your terminal application and run the following command:

```
java -version
```

If the output looks something like `java version "1.8.x"`, you are good to go. If not, you may need to update your version; see the **Oracle Java website** to download the latest JDK.

We found that running the two different Java programs sometimes returned different values for GC %

We didn’t have the source code, but we found the error by creating a simple test dataset to put into the programs
i.e., rather than calculating the GC% for 38000 sequences, we created a test file containing the sequence

```
>test1
AACCGGTT
```

Now we are going to do the same thing by writing some Python code

We will do this by starting with a simple Python program and run in two different ways

First of all, we will run it from inside a terminal window
Then we will run it inside Jupyter

Programming in Python

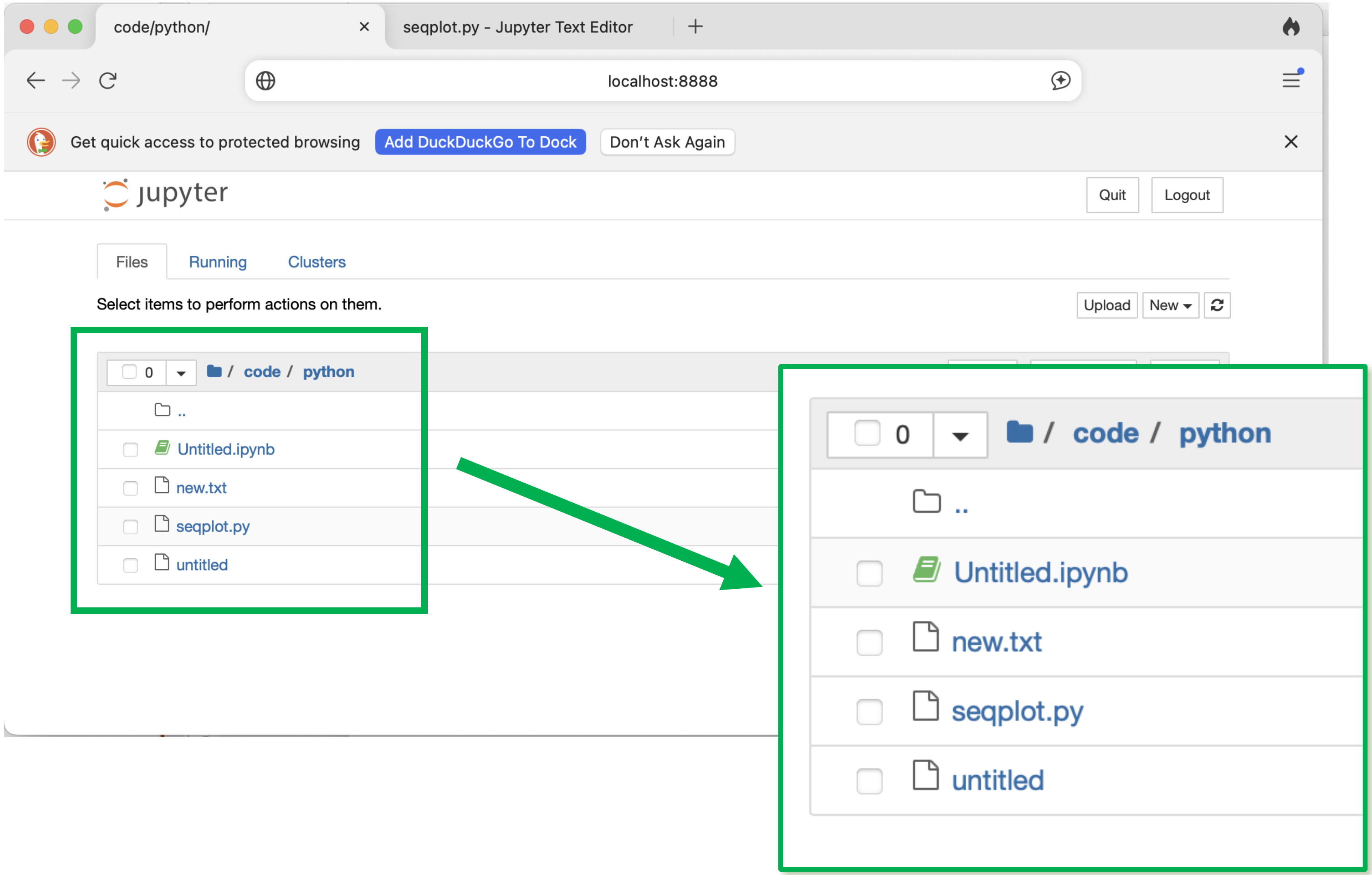
Writing code inside Jupyter Notebook

Open a command window
(in Windows you can do this by typing
`<CTRL>+<SHIFT>+P`)

When a window opens, type

```
jupyter notebook
```

After a while (depending on your computer's speed), a webpage will open that looks something like this



Select items to perform actions on them.

□ 0 / code / python

..

□ Untitled.ipynb

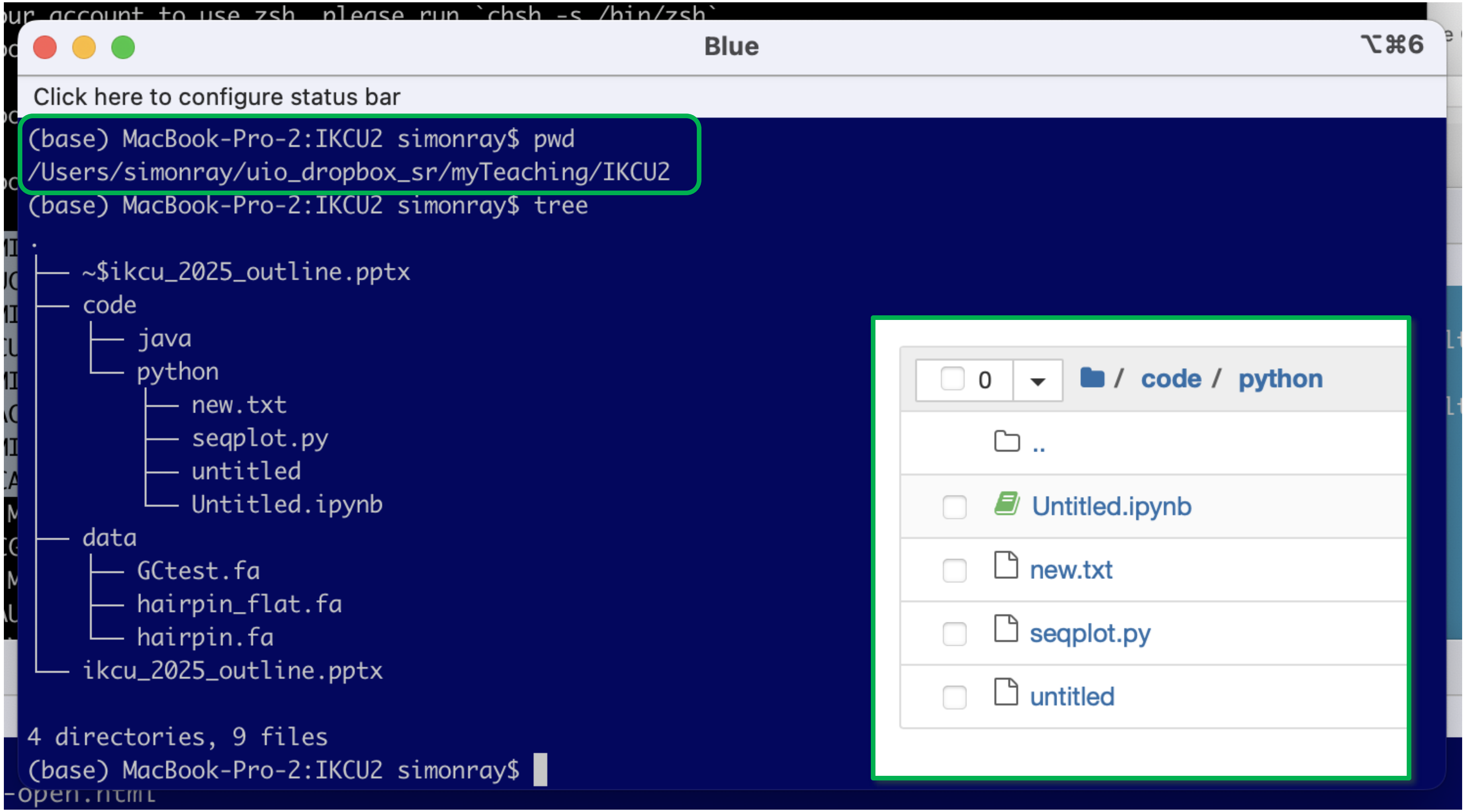
□ new.txt

□ seqplot.py

□ untitled

It won't look exactly the same, because you need to move to the folder where you downloaded the code

For example, before i started **jupyter**, i changed the directory to the folder where i downloaded the code



```
(base) MacBook-Pro-2:IKCU2 simonray$ pwd
/Users/simonray/uio_dropbox_sr/myTeaching/IKCU2
(base) MacBook-Pro-2:IKCU2 simonray$ tree
.
├── ~$ikcu_2025_outline.pptx
├── code
│   ├── java
│   └── python
│       ├── new.txt
│       ├── seqplot.py
│       ├── untitled
│       └── Untitled.ipynb
├── data
│   ├── GCtest.fa
│   ├── hairpin_flat.fa
│   └── hairpin.fa
└── ikcu_2025_outline.pptx

4 directories, 9 files
(base) MacBook-Pro-2:IKCU2 simonray$
```

□ 0 ▾ / code / python

..

□ Untitled.ipynb

□ new.txt

□ seqplot.py

□ untitled

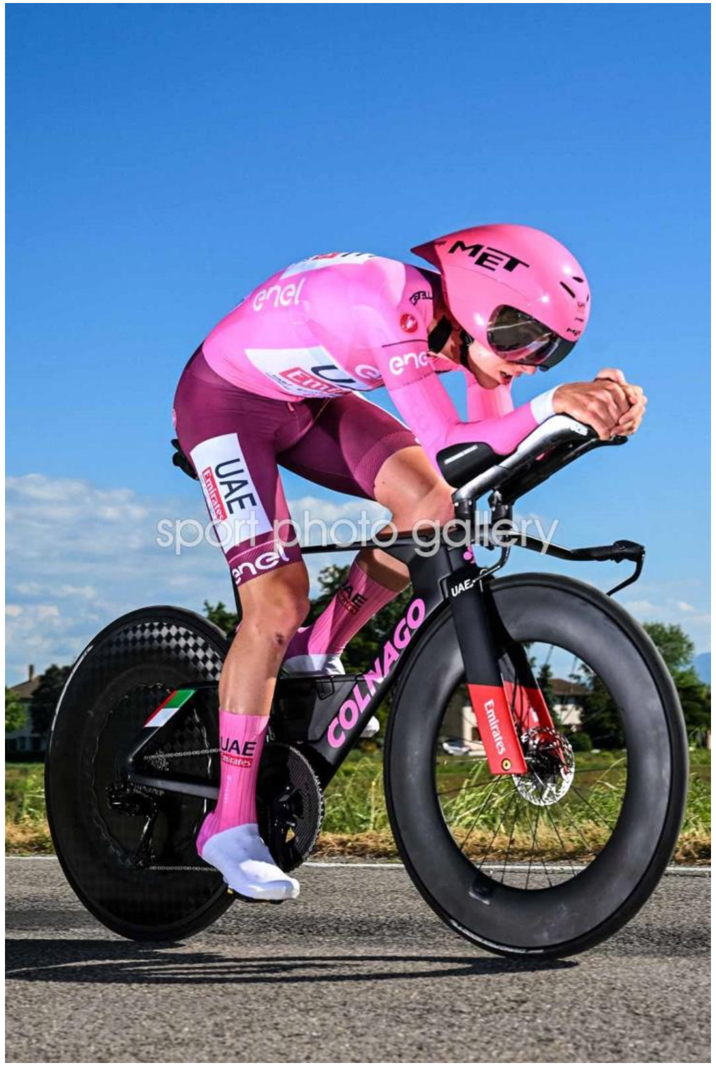
Programming in Python

Running code inside a virtual environment

How would you describe a bicycle?

- Two wheels
- Handlebar
- Saddle
- frame

Here is a bike



Here is another bike

The parts are incompatible (e.g., you can’t change the wheels

This is a bit like the problem you face with Python

```
import sklearn
import tensorflow as tf
```

They both use Pandas, but may require different versions of the Pandas package

What about Anaconda or MiniConda?

Anaconda gives you whatever version happens to be in the package. So it tries to make sure the are no dependency issues, but it may break other programs you already have installed

Also, Anaconda gives you many other packages you may never use

Virtual Environments

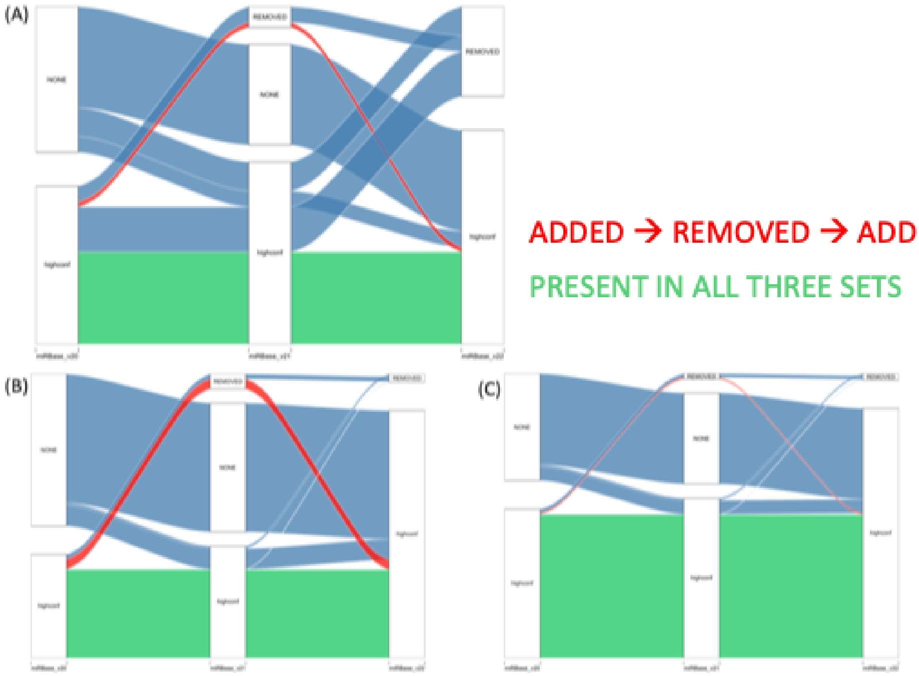
Virtual Enviroments give you a way to create a custom environment to run your code

VE1

VE 2

Reproducible Research

miRBASE ANNOTATION : HIGH CONFIDENCE SETS



ADDED → REMOVED → ADDED

PRESENT IN ALL THREE SETS

TO ADDRESS SOME OF THESE ANNOTATION PROBLEMS miRBASE RELEASED A HIGH CONFIDENCE ANNOTATION SET

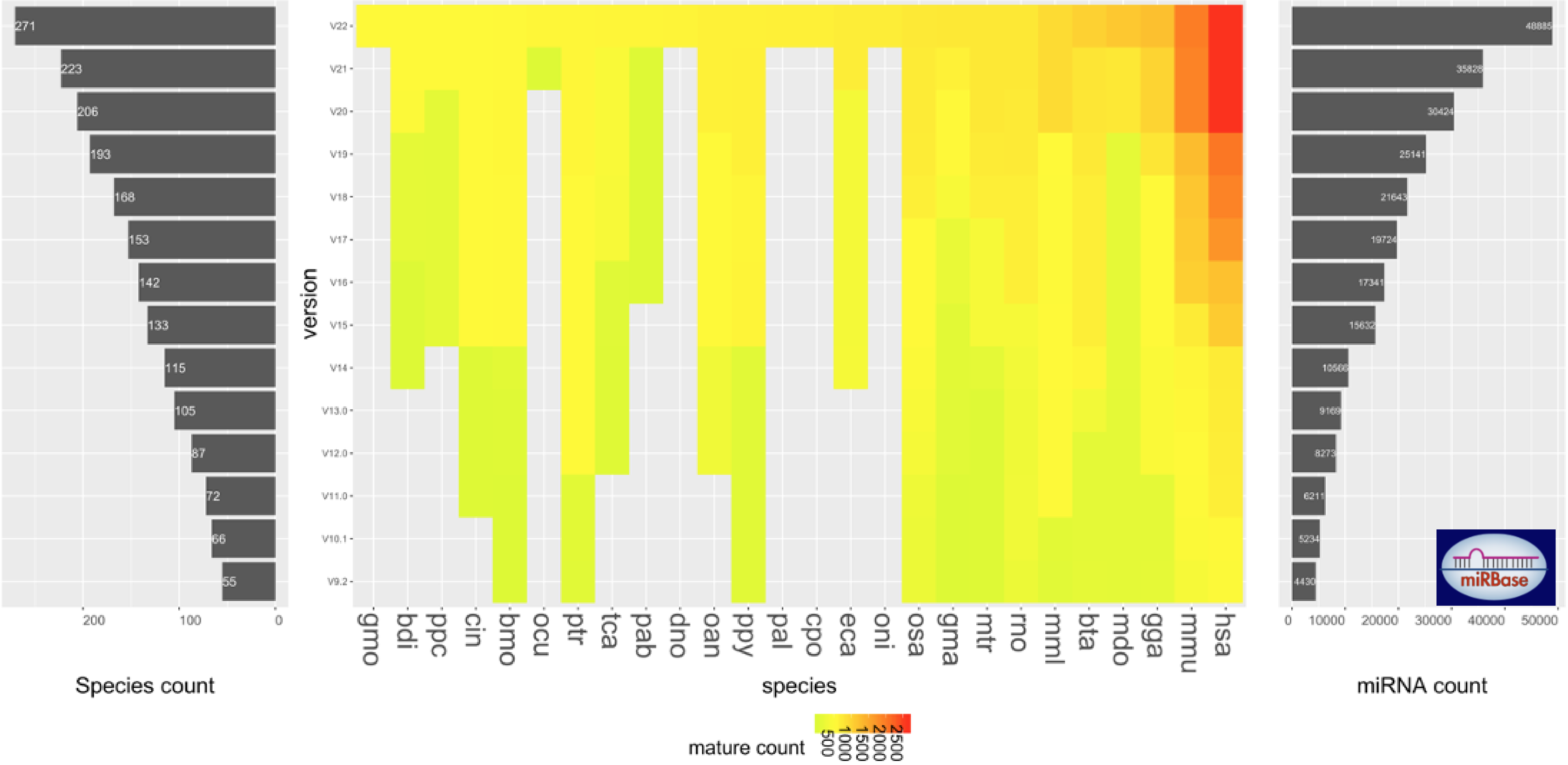
3298 DISTINCT HAIRPIN PRECURSORS ACROSS THE THREE RELEASES,
ONLY 925 (231 HUMAN HAIRPINS) ARE PRESENT ACROSS ALL THREE RELEASES

We saw that the high confidence miRBase set changed between each version

How does the variation among different miRBase releases affect an analysis?

Variation in miRBase Annotation

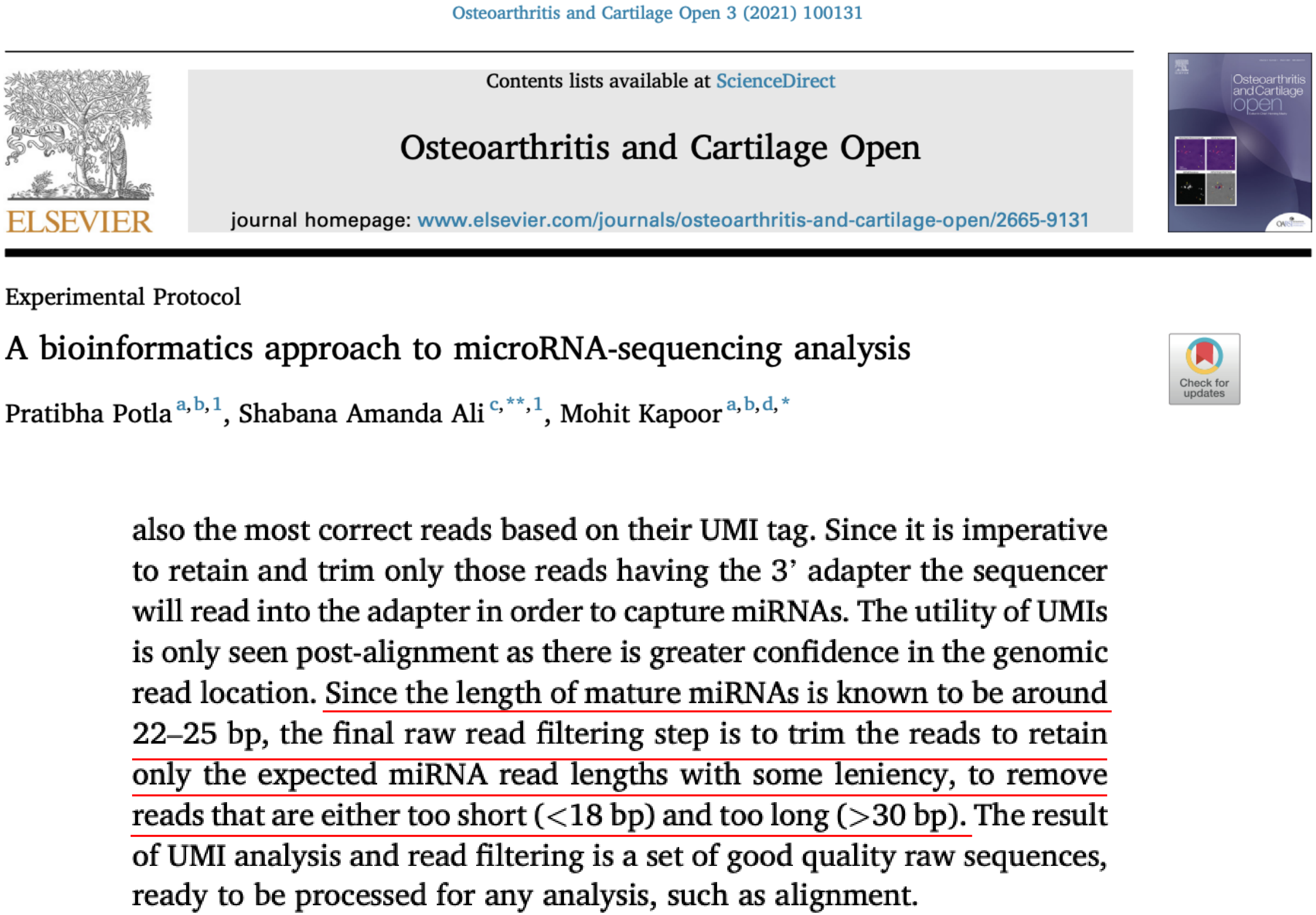
miRBase: Growth



We are going to look at the impact of using different annotation and parameter values on the analysis results

To do this, we are going to run the mapping part of a NGS analysis using smallRNA seq data using the `bowtie` mapping tool

To do the analysis, we need some reads, and a reference genome
It will take too long to work with a whole NGS dataset + Reference Genome, so we are going to use a test dataset, `smallRNA_reads.fa`
And a shorter reference genome containing only chr 10 and chr X.



Osteoarthritis and Cartilage Open 3 (2021) 100131

Contents lists available at ScienceDirect

Osteoarthritis and Cartilage Open

journal homepage: www.elsevier.com/journals/osteoarthritis-and-cartilage-open/2665-9131

Experimental Protocol

A bioinformatics approach to microRNA-sequencing analysis

Pratibha Potla[a,b,1], Shabana Amanda Ali[c,**,1], Mohit Kapoor[a,b,d,*]

also the most correct reads based on their UMI tag. Since it is imperative to retain and trim only those reads having the 3' adapter the sequencer will read into the adapter in order to capture miRNAs. The utility of UMIs is only seen post-alignment as there is greater confidence in the genomic read location. Since the length of mature miRNAs is known to be around 22–25 bp, the final raw read filtering step is to trim the reads to retain only the expected miRNA read lengths with some leniency, to remove reads that are either too short (<18 bp) and too long (>30 bp). The result of UMI analysis and read filtering is a set of good quality raw sequences, ready to be processed for any analysis, such as alignment.

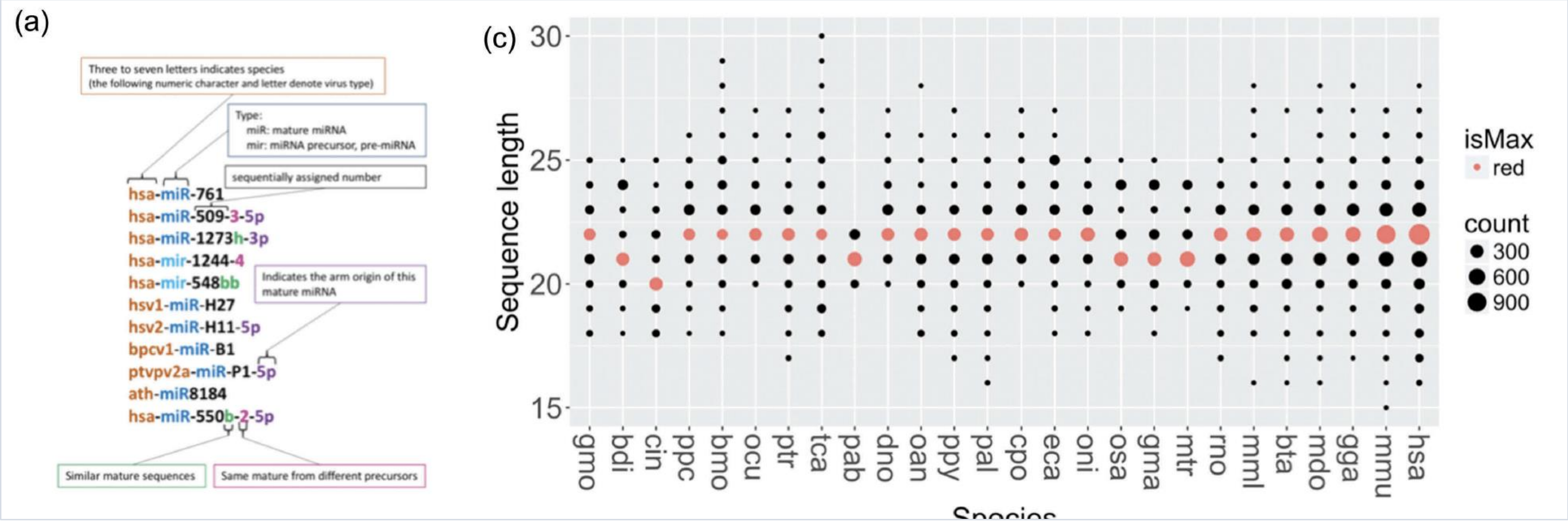
RNA BIOLOGY
2019, VOL. 16, NO. 11, 1534–1546
https://doi.org/10.1080/15476286.2019.1637680

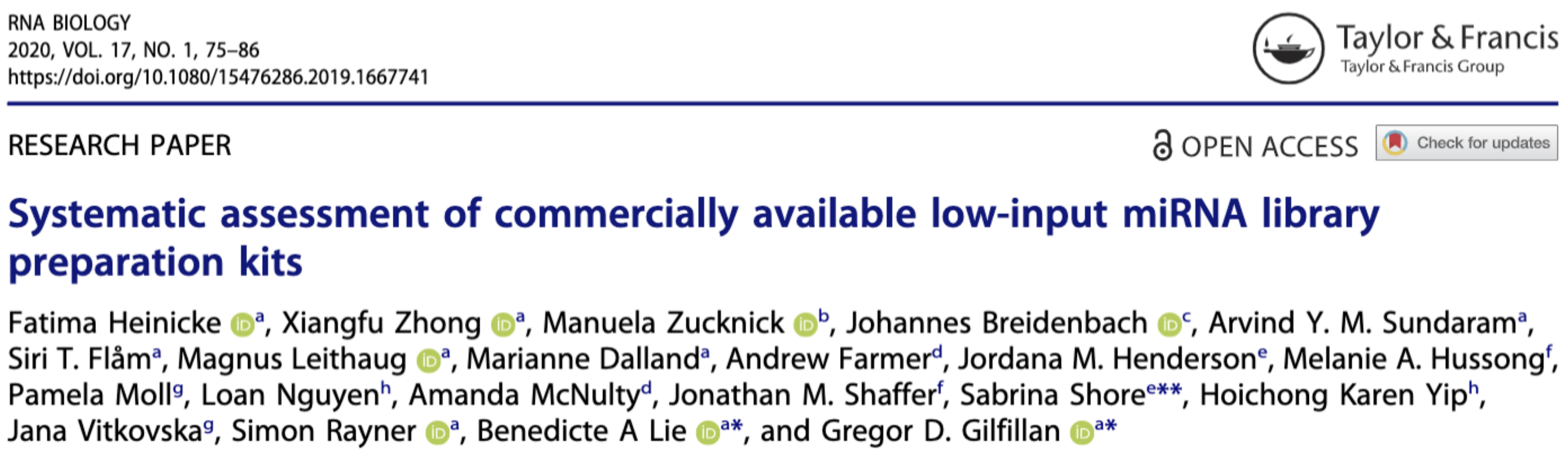
RESEARCH PAPER

miRBaseMiner, a tool for investigating miRBase content

Xiangfu Zhong, Fatima Heinicke, and Simon Rayner*

Department of Medical Genetics, Oslo University Hospital and University of Oslo, Oslo, Norway





RNA BIOLOGY
2020, VOL. 17, NO. 1, 75–86
https://doi.org/10.1080/15476286.2019.1667741

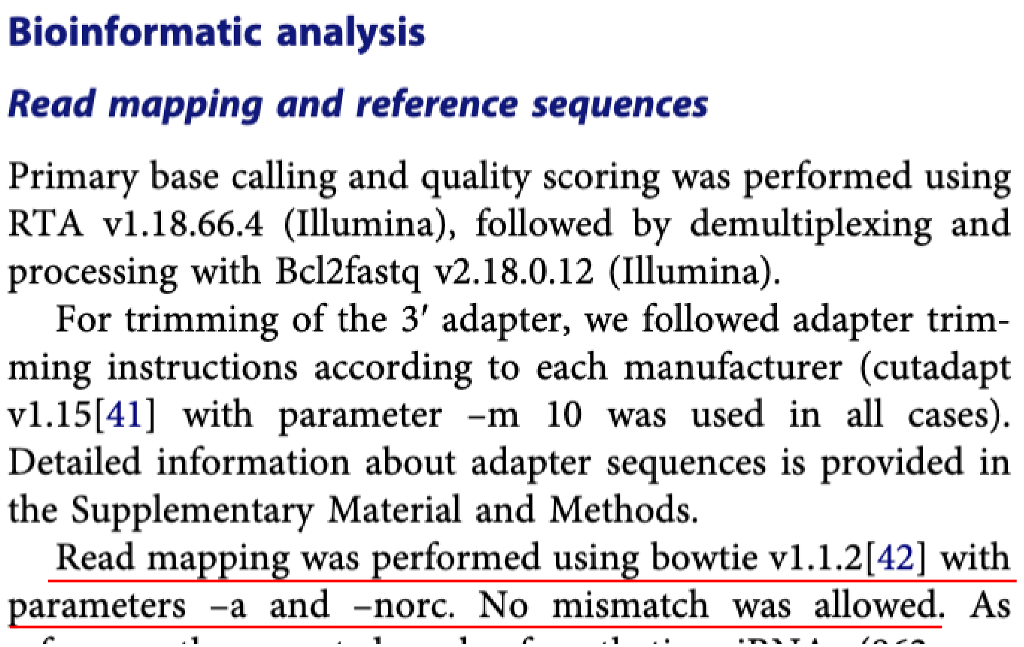
Taylor & Francis
Taylor & Francis Group

RESEARCH PAPER

OPEN ACCESS Check for updates

Systematic assessment of commercially available low-input miRNA library preparation kits

Fatima Heinicke[a], Xiangfu Zhong[a], Manuela Zucknick[b], Johannes Breidenbach[c], Arvind Y. M. Sundaram[a], Siri T. Flåm[a], Magnus Leithaug[a], Marianne Dalland[a], Andrew Farmer[d], Jordana M. Henderson[e], Melanie A. Hussong[f], Pamela Moll[g], Loan Nguyen[h], Amanda McNulty[d], Jonathan M. Shaffer[f], Sabrina Shore[e**], Hoichong Karen Yip[h], Jana Vitkovska[g], Simon Rayner[a], Benedicte A Lie[a*], and Gregor D. Gilfillan[a*]



Bioinformatic analysis

Read mapping and reference sequences

Primary base calling and quality scoring was performed using RTA v1.18.66.4 (Illumina), followed by demultiplexing and processing with Bcl2fastq v2.18.0.12 (Illumina).

For trimming of the 3′ adapter, we followed adapter trimming instructions according to each manufacturer (cutadapt v1.15[41] with parameter –m 10 was used in all cases). Detailed information about adapter sequences is provided in the Supplementary Material and Methods.

Read mapping was performed using bowtie v1.1.2[42] with parameters –a and –norc. No mismatch was allowed. As

```
bowtie -x bowtie/hsa_chr10 -f ngsdata/smallRNA_reads.fa -n 0
```

We are going to look at the impact of using different annotation and parameter values on the analysis results

To do this, we are going to run the mapping part of a NGS analysis using smallRNA seq data using the `bowtie` mapping tool

To do the analysis, we need some reads, and a reference genome
It will take too long to work with a whole NGS dataset + Reference Genome, so we are going to use a test dataset, `smallRNA_reads.fa`
And a shorter reference genome containing only chr 10 and chr X.

Let’s see how this affects read mapping ….

What is the length distribution of human miRNAs?

We need the following software

bowtie https://sourceforge.net/projects/bowtie-bio/files/bowtie/1.3.1/

samtools

bedtools

bgzip

Installed using apt-get install

c:\User\simonrat\Downloads —> /mnt/c/Users/simonrat/Downloads

1. Map the reads to the reference genome

```
bowtie -x bowtie/hsa_chr10 -f ngsdata/smallRNA_reads.fa
bowtie -x bowtie/hsa_chr10 -f ngsdata/smallRNA_reads.fa -S 10.sam
```

2. Map the reads to the reference genome and write to 10.sam

```
samtools view -bo 10.bam 10.sam
```

3. Convert the alignment results to binary format (less space and faster to process)

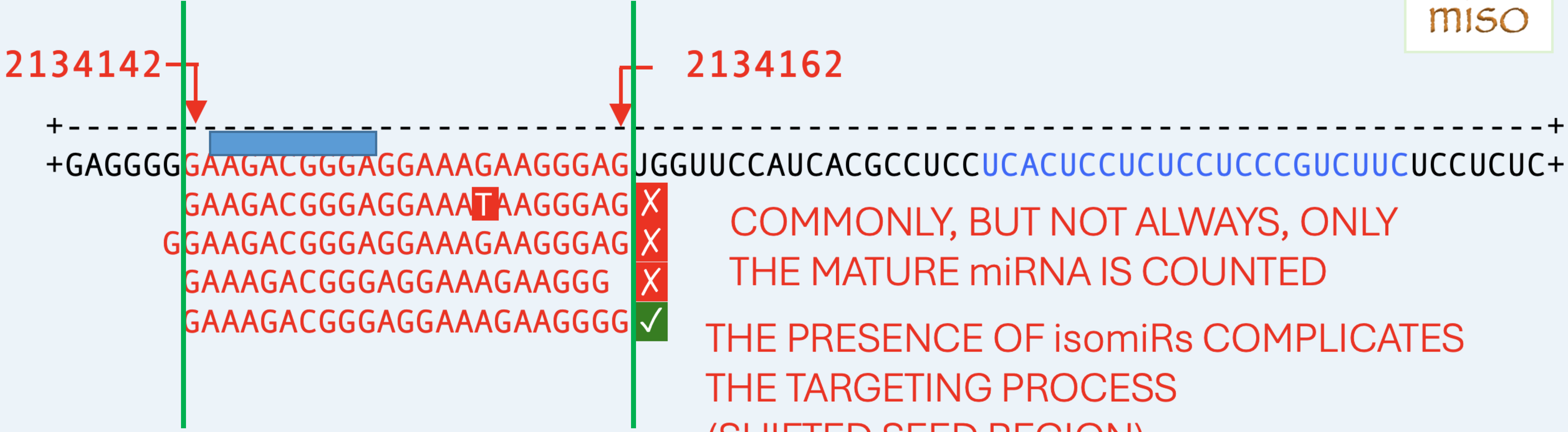
```
bedtools intersect -a mirbase/21/hsa_s.gff3 -b 10.bam
```

4. Find which reads overlap the features in the gff file

```
chr11           .               miRNA_primary_transcript        2134134         2134209         .               -               .
                ID=MI0002467;Alias=MI0002467;Name=hsa-mir-483
chr11           .               miRNA           2134181         2134202         .               -               .
                ID=MIMAT0004761;Alias=MIMAT0004761;Name=hsa-miR-483-5p;Derives_from=MI0002467
chr11           .               miRNA           2134142         2134162         .               -               .
                ID=MIMAT0002173;Alias=MIMAT0002173;Name=hsa-miR-483-3p;Derives_from=MI0002467
```

```
chr11    .      miRNA_primary_transcript  2134134  2134209  .
         -
chr11    .      miRNA     2134181  2134202  .        -        .

chr11    .      miRNA     2134142  2134162  .        -        .
```

IN REALITY, THERE CAN BE MULTIPLE ISOFORMS (isomiRs) OF AN miRNA

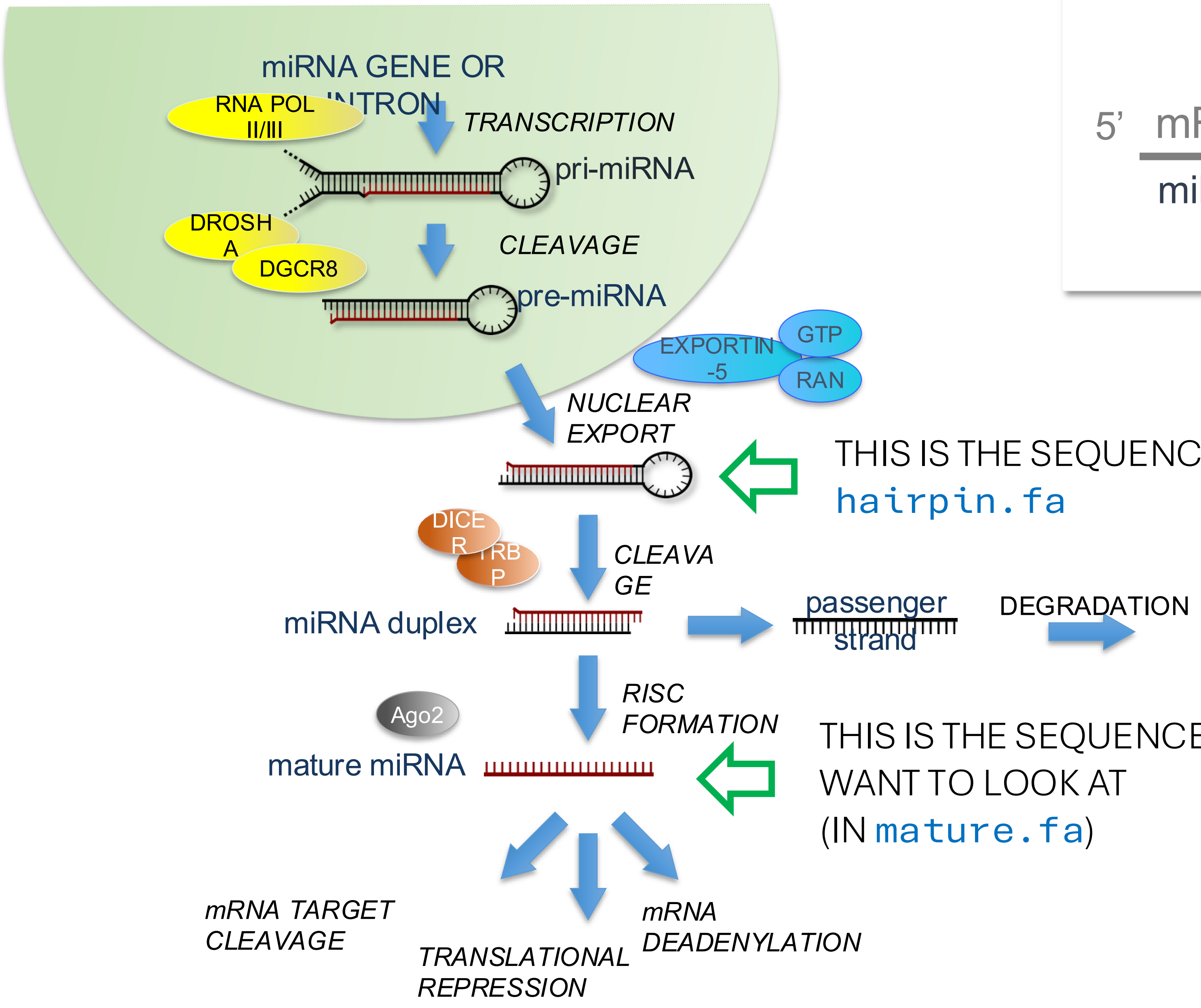


```
2134142                                       2134162
 +-----------------------------------------------------------------------------+
 +GAGGGGGAAGACGGGAGGAAAGAAGGGAGUGGUUCCAUCACGCCUCCUCACUCCUCUCCUCCCGUCUUCUCCUCUC+
       GAAGACGGGAGGAAATAAGGGAG ✗
      GGAAGACGGGAGGAAAGAAGGGAG ✗
       GAAAGACGGGAGGAAAGAAGGG  ✗
       GAAAGACGGGAGGAAAGAAGGGG ✓
```

COMMONLY, BUT NOT ALWAYS, ONLY THE MATURE miRNA IS COUNTED

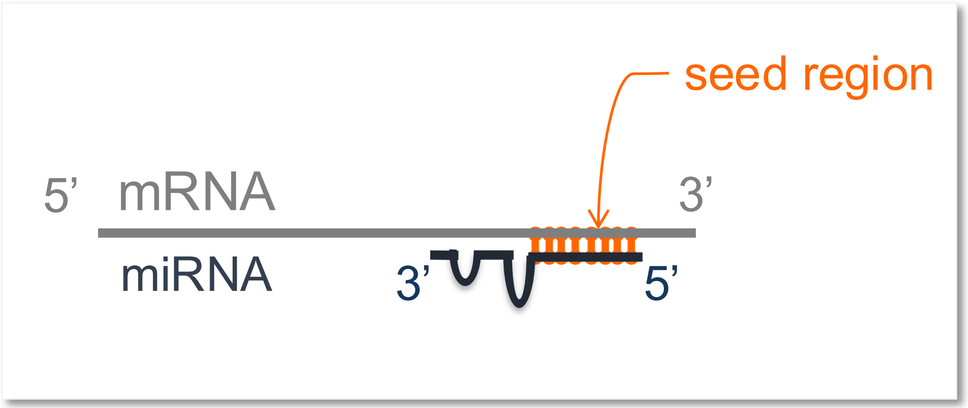
THE PRESENCE OF isomiRs COMPLICATES THE TARGETING PROCESS (SHIFTED SEED REGION)

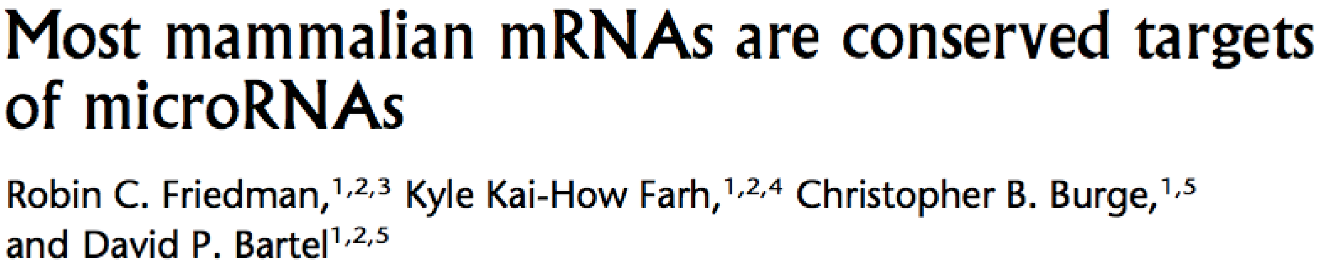
So far, we have been working with the hairpin.fa file. Actually, what we are really interested in are the miRNAs that are generated from the hairpin sequence



THIS IS THE SEQUENCE IN `hairpin.fa`

THIS IS THE SEQUENCE WE WANT TO LOOK AT (IN `mature.fa`)

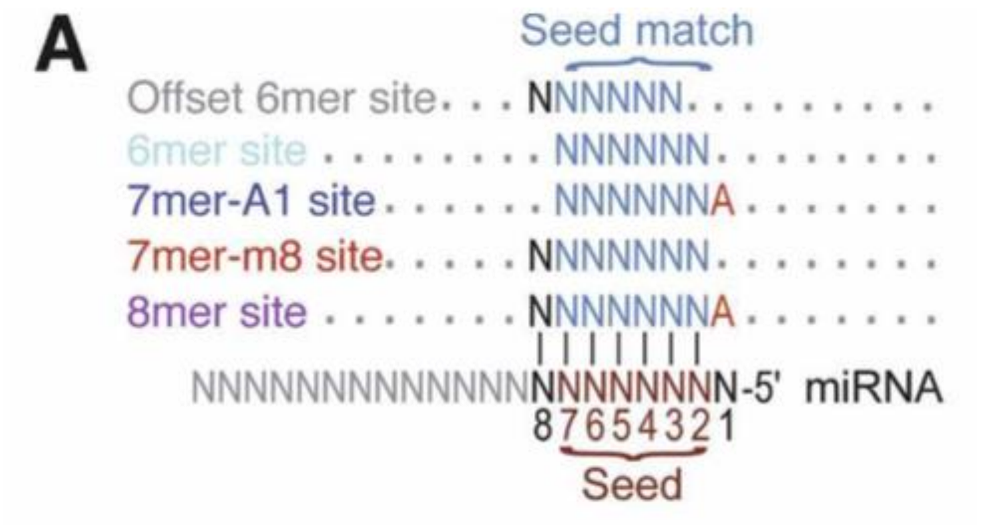


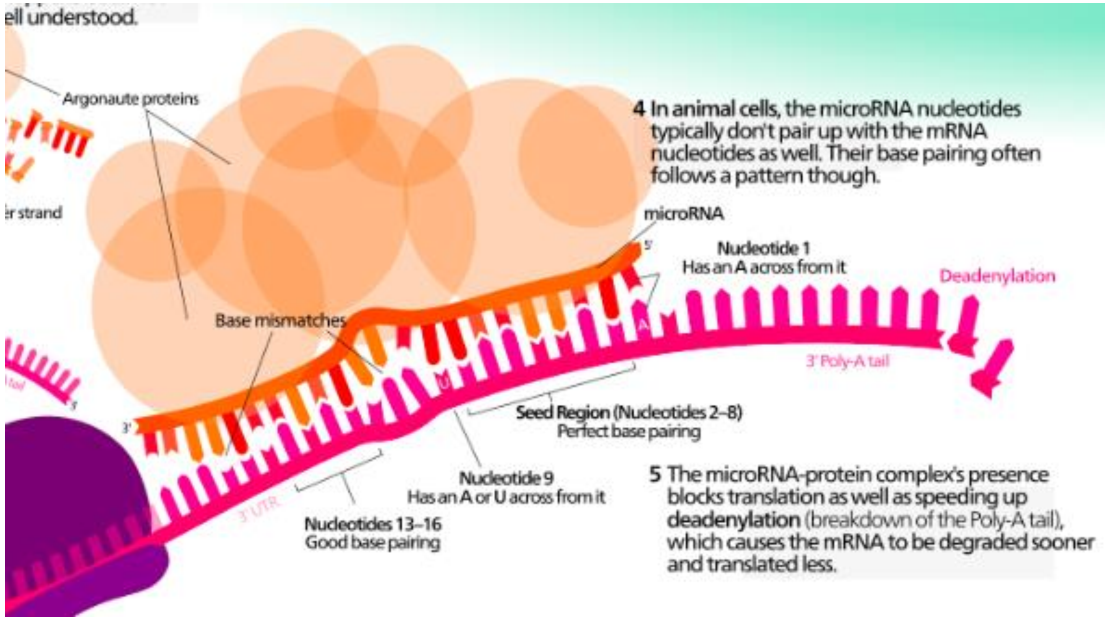


Most mammalian mRNAs are conserved targets of microRNAs

Robin C. Friedman,[1,2,3] Kyle Kai-How Farh,[1,2,4] Christopher B. Burge,[1,5]
and David P. Bartel[1,2,5]

Targeting model


A
Seed match
Offset 6mer site . . . NNNNNN. . . . .
6mer site . . . . NNNNNN. . . . .
7mer-A1 site . . . . NNNNNNA . . . .
7mer-m8 site. . . . . NNNNNNN. . . . .
8mer site . . . . NNNNNNNA . . . .
NNNNNNNNNNNNNNNNNNNNN-5' miRNA
87654321
Seed



4 In animal cells, the microRNA nucleotides typically don't pair up with the mRNA nucleotides as well. Their base pairing often follows a pattern though.

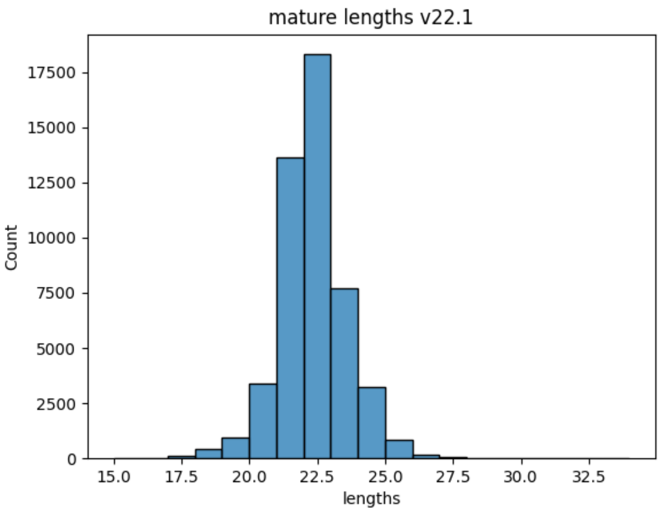
5 The microRNA-protein complex's presence blocks translation as well as speeding up **deadenylation** (breakdown of the Poly-A tail), which causes the mRNA to be degraded sooner and translated less.

TARGET SITE IDENTIFICATION

2009: 10.1101/gr.082701.108

So, let’s begin by looking at the mature sequences in miRBase
These are stored in

```
reproducible_research/data/mirbase
├── 21
│   ├── hsa_s.gff3
│   ├── hsa.gff3
│   └── mature.21.fa
└── 22.1
    ├── hairpin.fa
    ├── hsa.gff3
    └── mature.fa
```



```
from Bio import SeqIO
import pandas as pd

records = list(SeqIO.parse("22.1/mature.fa", "fasta"))

seqLens = []
i=0
while i < len(records):
    seqLens.append(len(records[i].seq))
    i = i + 1
dfseqLens = pd.DataFrame(seqLens, columns=['lengths'])

from matplotlib import pyplot as plt
import seaborn as sns

histplot=sns.histplot(data=dfseqLens, x="lengths",
binwidth=1)
fig=histplot.get_figure()
fig.savefig("22.1/out.png")
```

This is the plot for all mature sequences.
Next,
generate plots for

1. human and mouse, and for 21 and 22
2. Human and mouse, for seed regions, for 21 and 22

Give your files sensible names so you know which file is which

You need to get the mature hsa sequences only. You can do this using `grep`

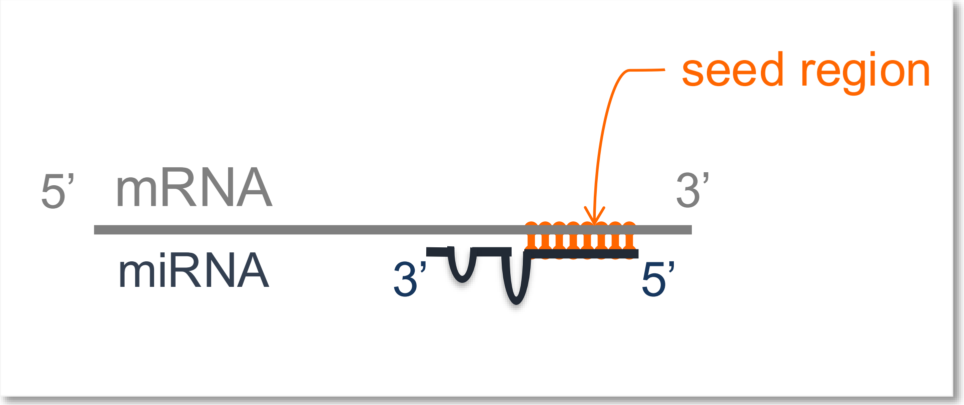
I haven’t put in the full file path to save space

```
grep -A 1 hsa mature.fa | grep -v "^--" > hsa_mature_v21.fa
grep -A 1 hsa mature.fa | grep -v "^--" > hsa_mature_v22.fa
```

Day 5

Human miRNAs seem to be shorter than mouse miRNAs

But we saw that it is the seed region that is most important for how the miRNA chooses a target



On day 1, when you generated the logoplot, you also

When you generated the logoplot for the seed region, you also wrote out a list of all the unique sequences

We can use this information to see how the seed region varies between human and mouse

gccalc/entrypoint.py

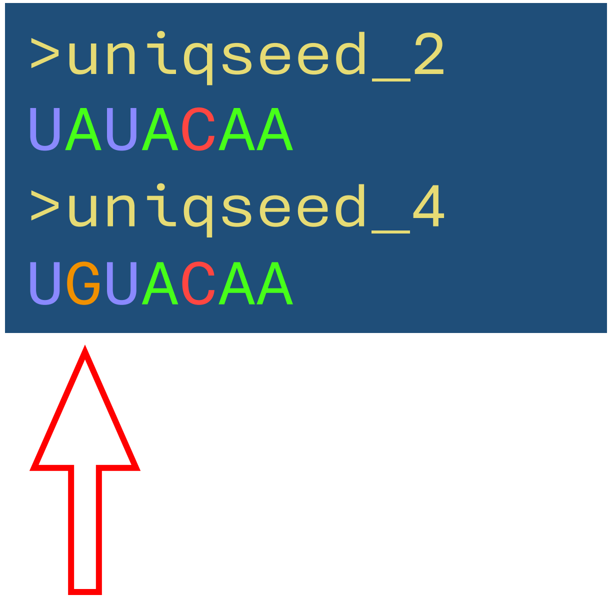
```
def main(argv=None): # IGNORE:C0111

    if argv is None:
        argv = sys.argv

    parseArgs(argv)

    n = readFastaFile(fastaFile)
    uSeqs = getUniqueSeedSequences()

    writeUniqSeqs(uSeqs)      line 338

    dfNTFrequencies = getNucleotideFrequencyMatrix(uSeqs)
    generateLogoPlot(dfNTFrequencies)
```

Distance

Distance measures

How can we measure the distance between two sequences?

For example, in the list of unique human seed sequences we have the following two sequences

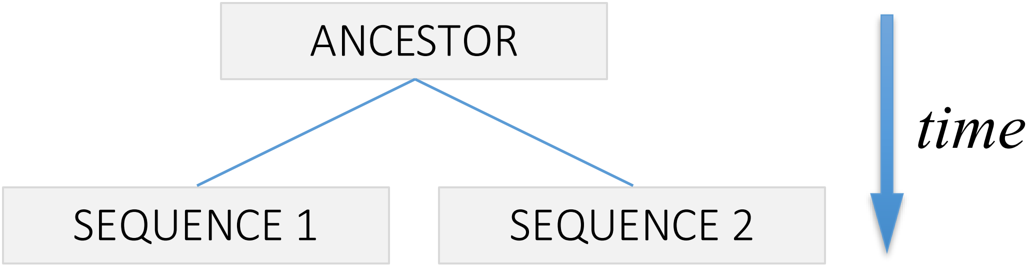


```
>uniqseed_2
UAUACAA
>uniqseed_4
UGUACAA
```

We have a single change in position 2 from A→G

So we can just count the differences between each pair of sequences

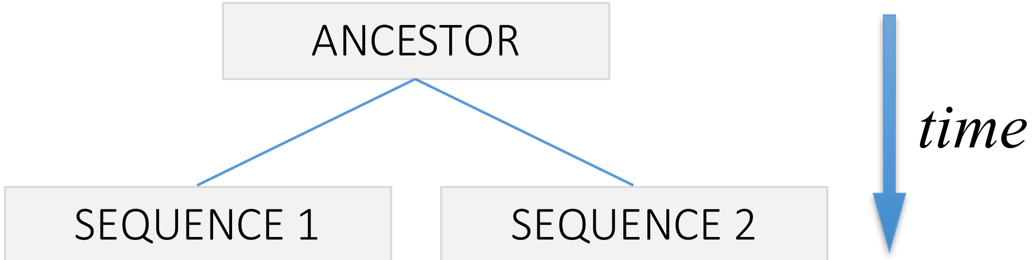
This is probably fine for this application, as the sequences are short and we just want to get a general idea of the differences between the sets of seed regions for human and mouse



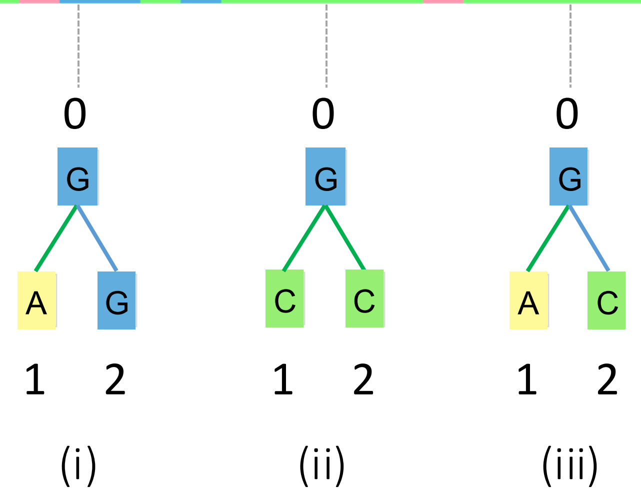
C	T	A	G	G	G	G	C	C	C	C	T	C	C	A	C	C	C	C	G	C	C	T	1
C	T	G	G	C	G	C	C	C	C	C	T	C	C	C	C	C	C	A	A	A	C	T	2
		*		*		*								*				*	*	*			D = 7/23

FOR SMALL NUMBERS OF CHANGES, CAN USE *D* AS A DISTANCE MEASURE

But its not always so simple. When we are comparing two sequences, we are effectively assuming they are coming from a common ancestor that we don’t know



?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0
C	T	A	G	G	G	G	C	C	C	C	T	C	C	A	C	C	C	C	G	C	C	T	1
C	T	G	G	C	G	C	C	C	C	C	T	C	C	C	C	C	C	A	A	A	C	T	2



(i) ONE SUBSTITUTION/ONE VISIBLE

(ii) TWO SUBSTITUTIONS/NONE VISIBLE

(iii) TWO SUBSTITUTIONS/ONE VISIBLE

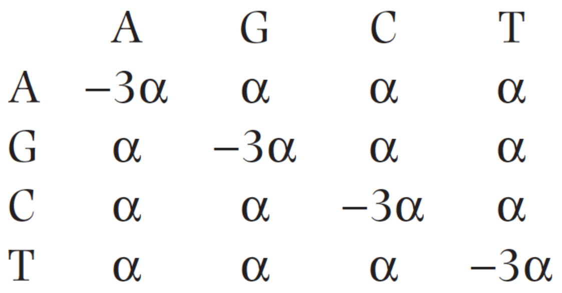
NO LONGER A LINEAR RELATIONSHIP BETWEEN *D* AND TIME

In such cases (for example a rapidly mutating virus) we need to use nucleotide substitution models

In the simplest case, there is an equal probability of one nucleotide being substituted for another

NEED BETTER MODEL OF SEQUENCE EVOLUTION

JUKES-CANTOR



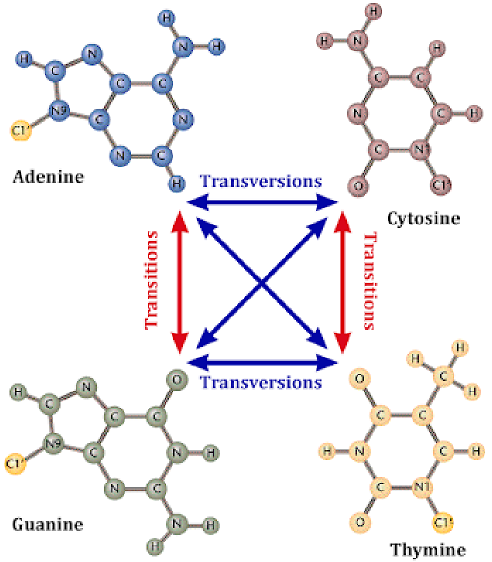
$$\begin{array}{ccccc} & A & G & C & T \\ A & -3\alpha & \alpha & \alpha & \alpha \\ G & \alpha & -3\alpha & \alpha & \alpha \\ C & \alpha & \alpha & -3\alpha & \alpha \\ T & \alpha & \alpha & \alpha & -3\alpha \end{array}$$

EQUAL PROBABILITY OF SUBSTITUTION BETWEEN ANY TWO BASES AT ANY SITE

For example, transitions occur more frequently than transversions (pyrimidine <-> purine), so a better model would include this

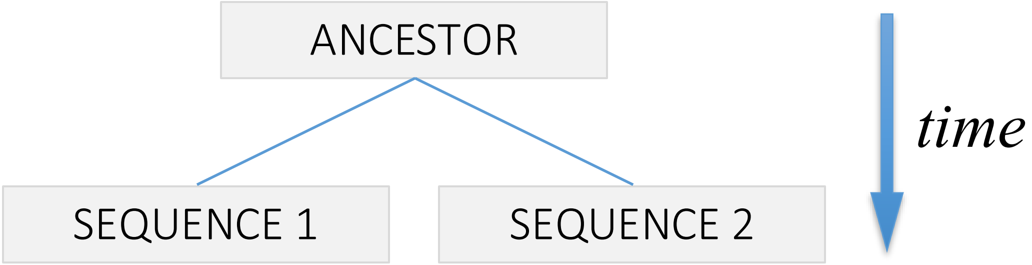
JUKES-CANTOR

$$\begin{array}{ccccc}
 & A & G & C & T \\
A & -\alpha\pi_G - \beta(\pi_C + \pi_T) & \alpha\pi_G & \beta\pi_C & \beta\pi_T \\
G & \alpha\pi_A & -\alpha\pi_A - \beta(\pi_C + \pi_T) & \beta\pi_C & \beta\pi_T \\
C & \beta\pi_A & \beta\pi_G & -\alpha\pi_T - \beta(\pi_A + \pi_G) & \alpha\pi_T \\
T & \beta\pi_A & \beta\pi_G & \alpha\pi_C & -\alpha\pi_C - \beta(\pi_A + \pi_G)
\end{array}$$



EQUAL PROBABILITY OF SUBSTITUTION BETWEEN ANY TWO BASES AT ANY SITE

But here, we will stick to the simple measure of distance by counting the number of differences between each sequence pair



C	T	A	G	G	G	G	C	C	C	C	T	C	C	A	C	C	C	C	G	C	C	T	1
C	T	G	G	C	G	C	C	C	C	C	T	C	C	C	C	C	C	A	A	A	C	T	2
		*		*		*								*				*	*	*			D = 7/23

FOR SMALL NUMBERS OF CHANGES, CAN USE *D* AS A DISTANCE MEASURE

Tasks

1. If you haven't already done this, run `gccalc/entrypoint.py` and generate a file containing a list of the unique seed regions for human and mouse.
   Generate one file for each species, and use v22.1 of miRBase
   Give your files sensible names so you know which file is which.

2. Run `networks/levenstein.py` and generate a histogram for each species

3. What do you find? How do you intepret?

4. What is the median and upper and lower quartile for each distribution?

Code notes