HKU SPACE and Edinburgh Napier University
Msc Biomedical Science
Scientific Skills Module

Data handling on Bioinformatics I

GU Haogao
The University of Hong Kong
guhaogao@hku.hk

2024-11-01

About me

www.guhaogao.com

- Currently a Research Assistant Professor in School of Public Health, HKU
- Research focuses: computational analysis on viral genome, studying virus evolution at host and population level.
- Daily work: NGS data processing, public data collection, data cleaning, model building/fitting, debugging, high-performance computing, etc..

The Learning Objective

Data handling on Bioinformatics

- Analyse and interpret DNA and protein sequence data using bioinformatics.
  - Knowing how DNA and protein sequences are stored as data
  - Knowing how to create and manipulate such data
  - How to use existing tools to boost your analysis
  - Standard practice and advices for bioinformatics projects
- In these two lessons, we will learn:
  - **General data skills** that applicable to a wide range of (bio)informatics projects.

Rundown

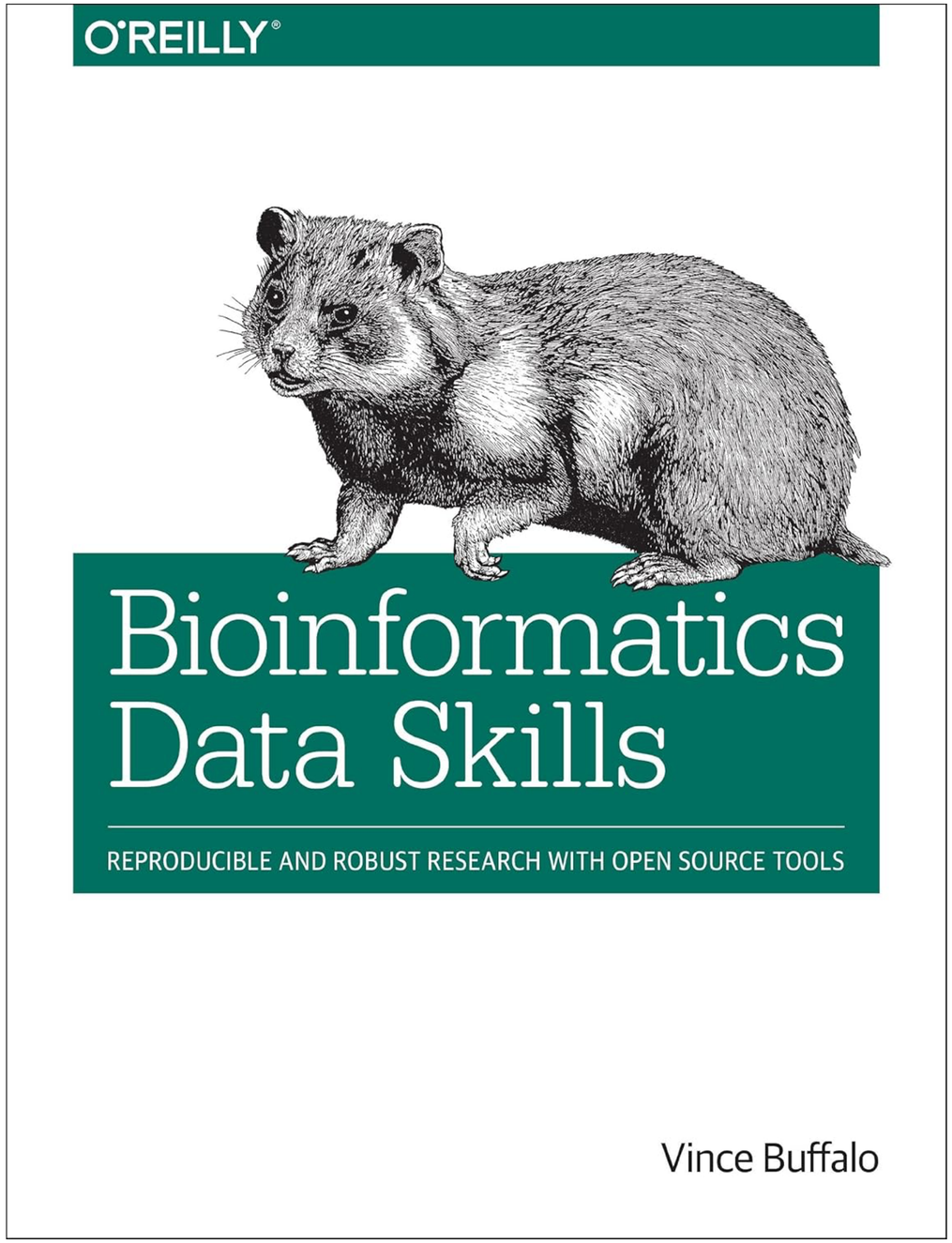
Lesson 1

- Ideology: Data skills in Robust and reproducible bioinformatics (15 mins)
- Practical skills for managing a bioinformatic project (25 mins)
  - Working folder
  - Documentation with Markdown
  - Data integrity
- Break (10 mins)
- A rapid introduction to the R language (40 mins)
  - Google Colab
  - R
- Break (10 mins)
- Storing sequence data (40 mins)
  - Fasta format
  - Fastq format

Acknowledgment

Reference book

- Bioinformatics Data Skills by Vince Buffalo
- My favourite introductory book to Bioinformatics
  - Useful
  - Funny
  - Easy to follow

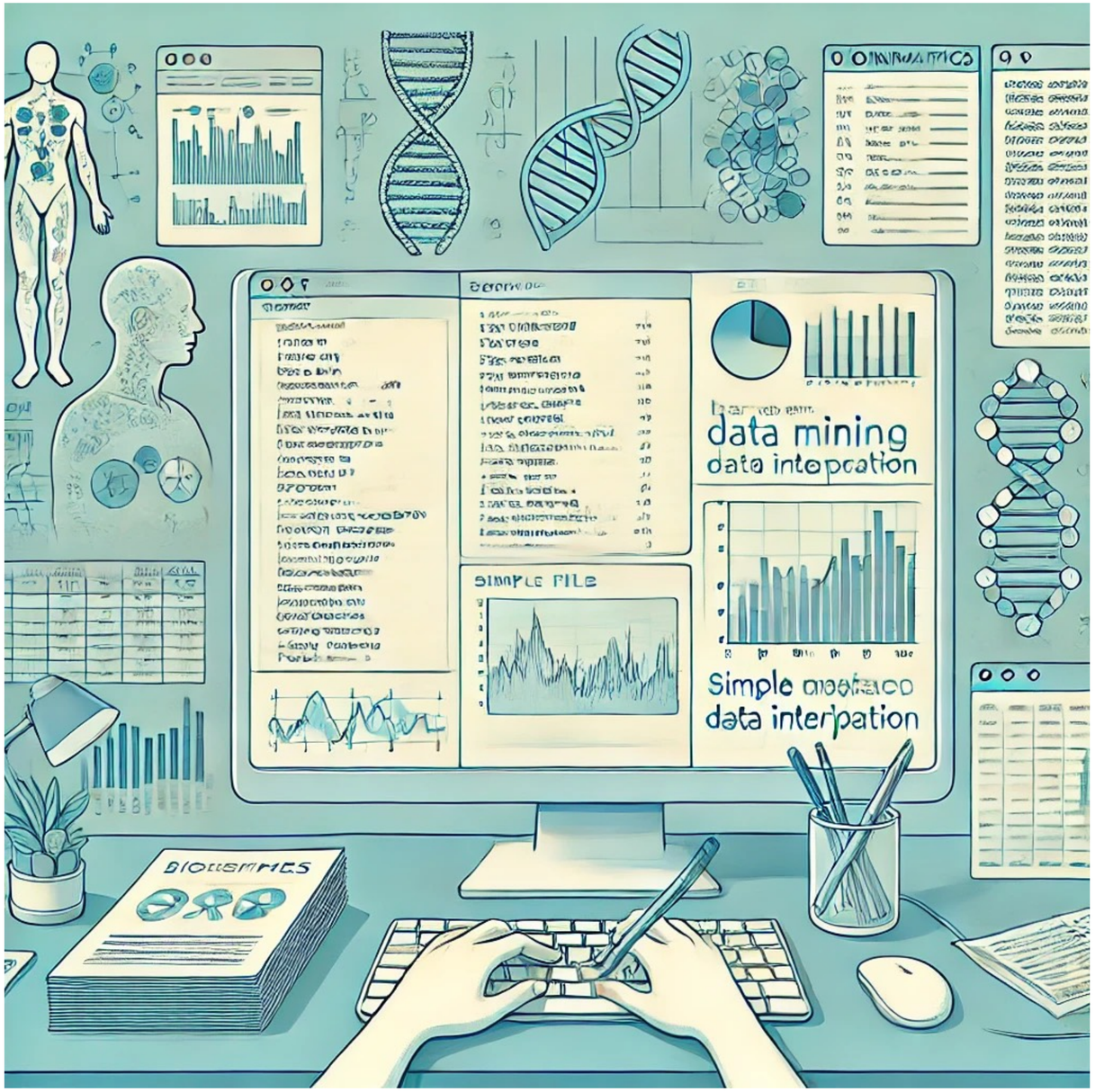


Ideology:
Data skills in Robust and reproducible bioinformatics

Bioinformatics

What is it?

- What is Bioinformatics?
  - Definition: Bioinformatics is an interdisciplinary field combining biology, computer science, mathematics, and statistics to manage, analyze, and interpret large-scale biological data.
  - Core Goal: Transform raw biological data into insights about life at molecular, cellular, and ecosystem levels.
- Fields of Application:
  - Genomics, Proteomics, Transcriptomics, Metagenomics, Phylogenetics, Structural Biology, and more.

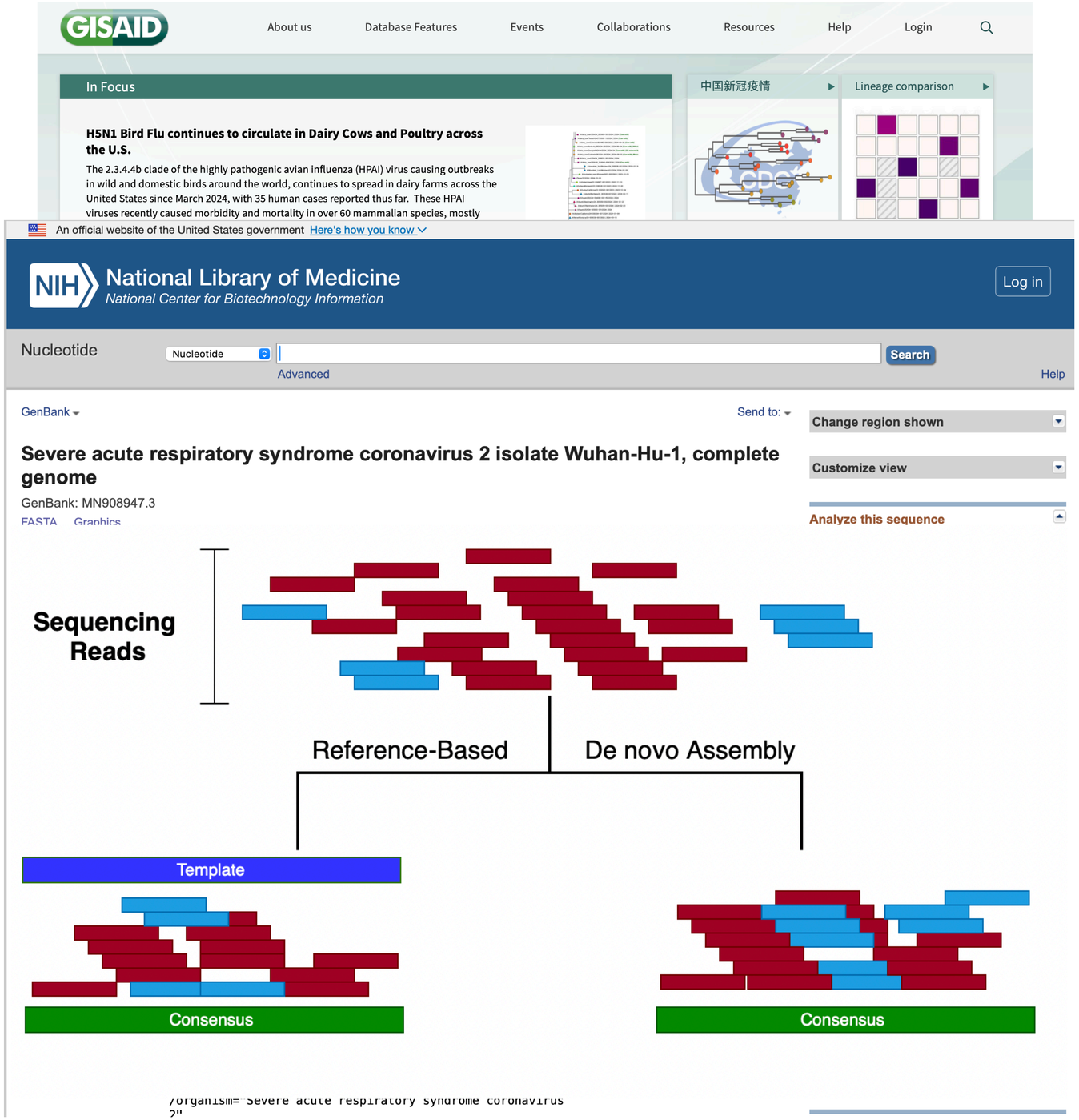


Images were generated by DALL·E

Key Focus Areas in Bioinformatics

All about biology and informatics

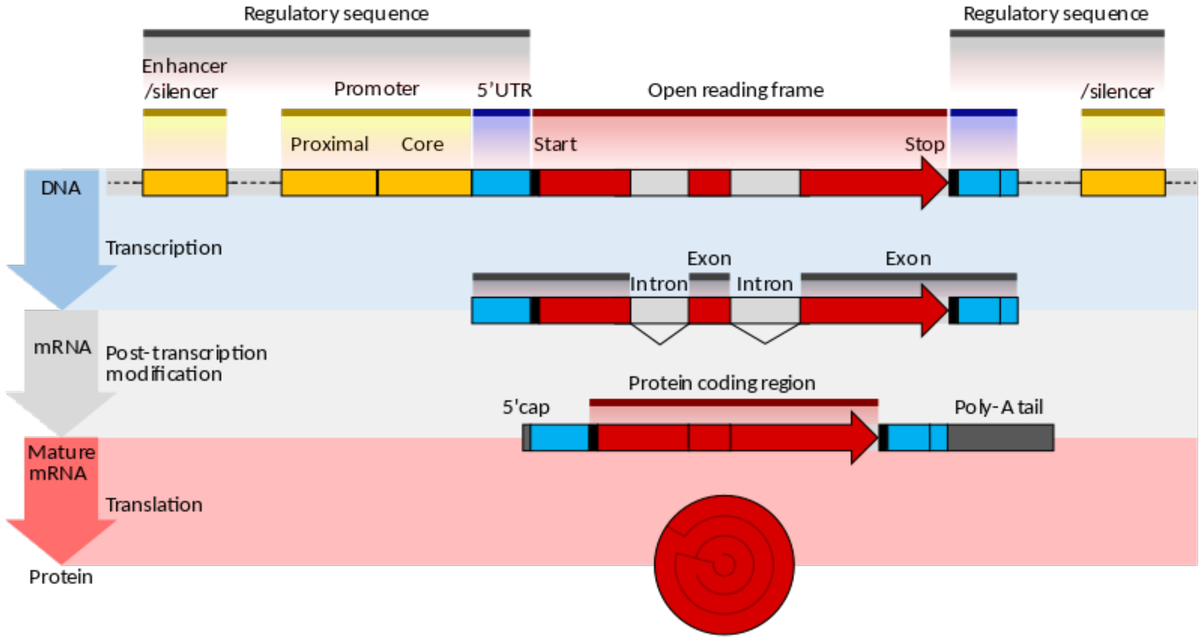
1. **Data Collection, Curation, and Storage:**
   - Gathering data from sequencing, imaging, and clinical sources.
   - Storing vast amounts of sequence data in organized, accessible databases (e.g., NCBI, EMBL-EBI).
   - Building and curating reference datasets and repositories for ongoing and future analyses.
2. **Data Cleaning and Quality Control:**
   - Filtering raw sequencing data, removing contaminants, correcting errors.
   - Ensuring high-quality data for accurate downstream analyses.
3. **Sequence Alignment and Assembly:**
   - Alignment: Comparing DNA/RNA/protein sequences to identify similarities and evolutionary relationships.
   - Assembly: Reconstructing entire genomes from sequencing data, whether de novo (from scratch) or using reference genomes.



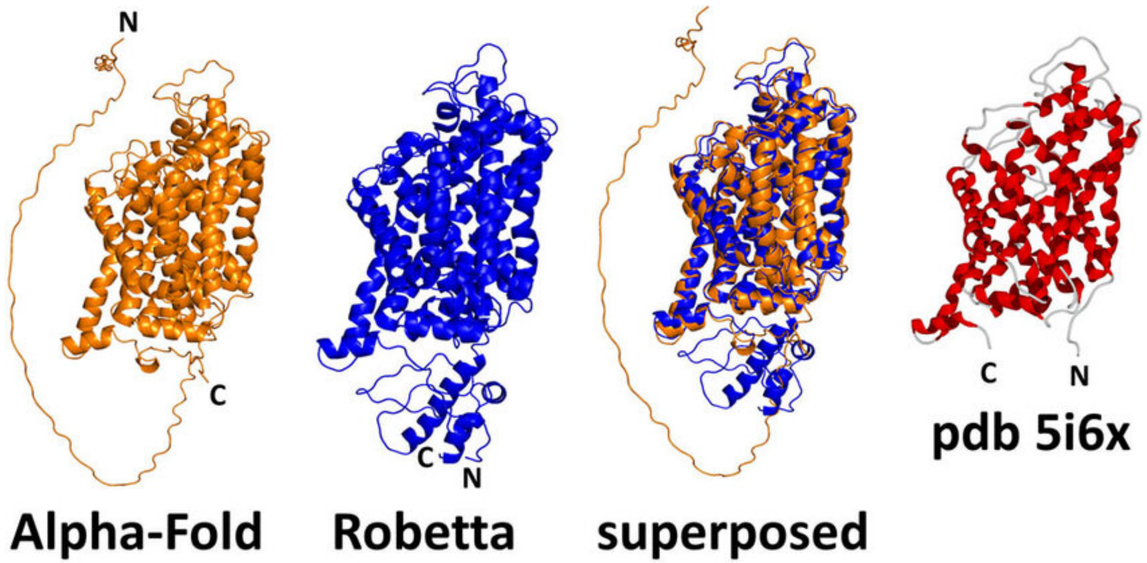
Key Focus Areas in Bioinformatics

All about biology and informatics

4. Functional Annotation:
   - Annotating genes, proteins, and functional regions of sequences.
   - Understanding protein functions, gene pathways, and interactions within biological systems.
5. Structural and Comparative Genomics:
   - Structural Genomics: Studying the 3D structures of proteins and other biomolecules.
   - Comparative Genomics: Comparing genomes across species to understand evolutionary changes, gene family expansions, and structural variations.



Sodium-dependent serotonin transporter (P31645)

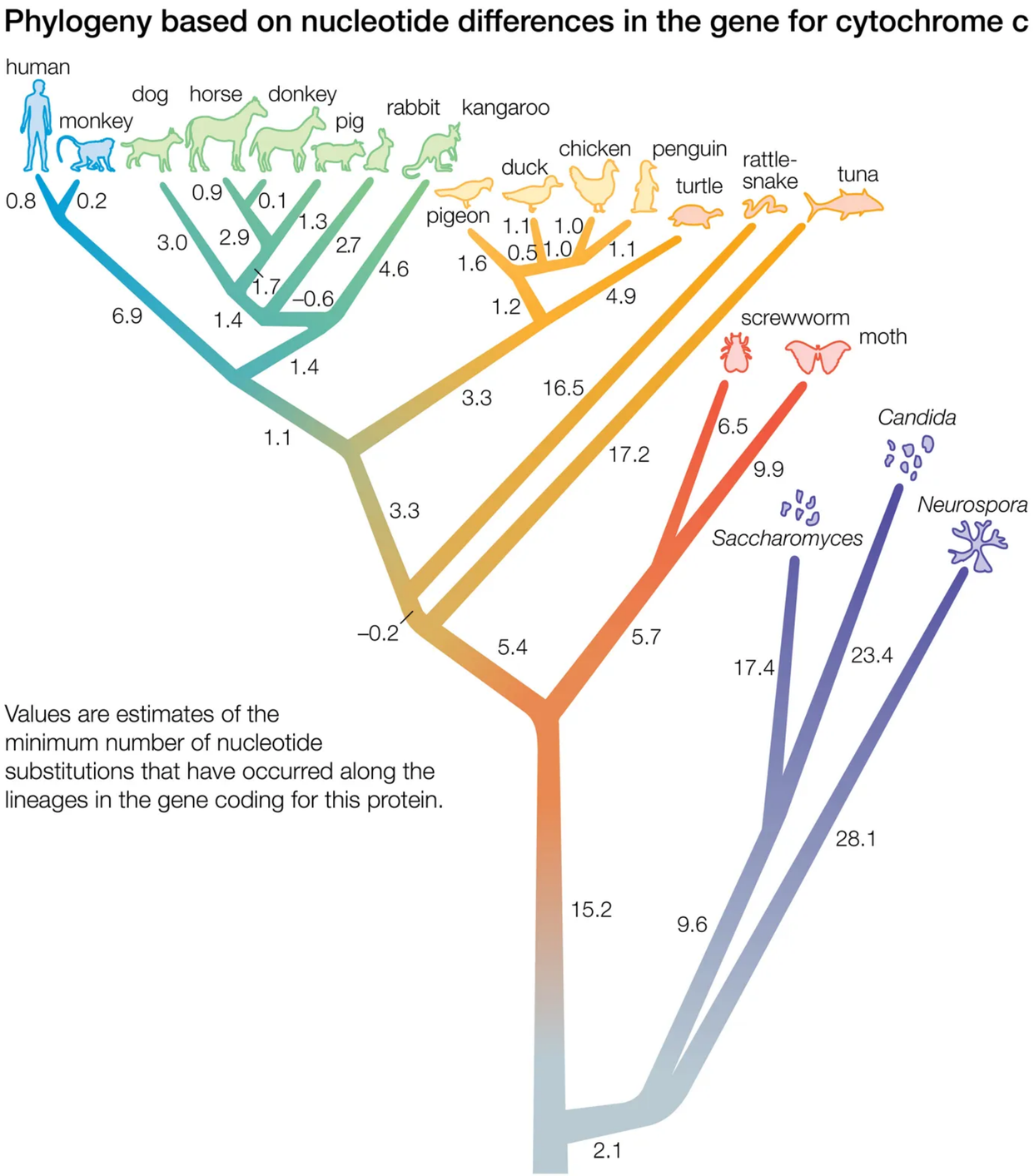


Alpha-Fold **Robetta** **superposed** **pdb 5i6x**

Key Focus Areas in Bioinformatics

All about biology and informatics

6. Systems Biology and Network Analysis:
   - Mapping gene and protein interactions to understand the bigger picture in biological systems.
   - Modeling metabolic and signaling pathways to investigate complex diseases like cancer, and immune responses.
7. Phylogenetics and Evolutionary Biology:
   - Phylogenetics: Analyzing evolutionary relationships between organisms based on genetic information.
   - Evolutionary Biology: Investigating evolutionary changes within species, e.g., mutations that drive viral evolution.
   - Applications: Tracking viral strains (e.g., COVID-19 variants), studying genetic drift in populations.
8. Others ...



Bioinformatics is trending

Growing data

- Sequencing cost dropped.
- Biological data grows.
- Moore's Law:
  - The number of transistors in computer chips doubles ~every two years.
- New tools are continuously being created.
- Challenges:
  - Poor reliability
  - Poor documentation
  - Poor generalisability

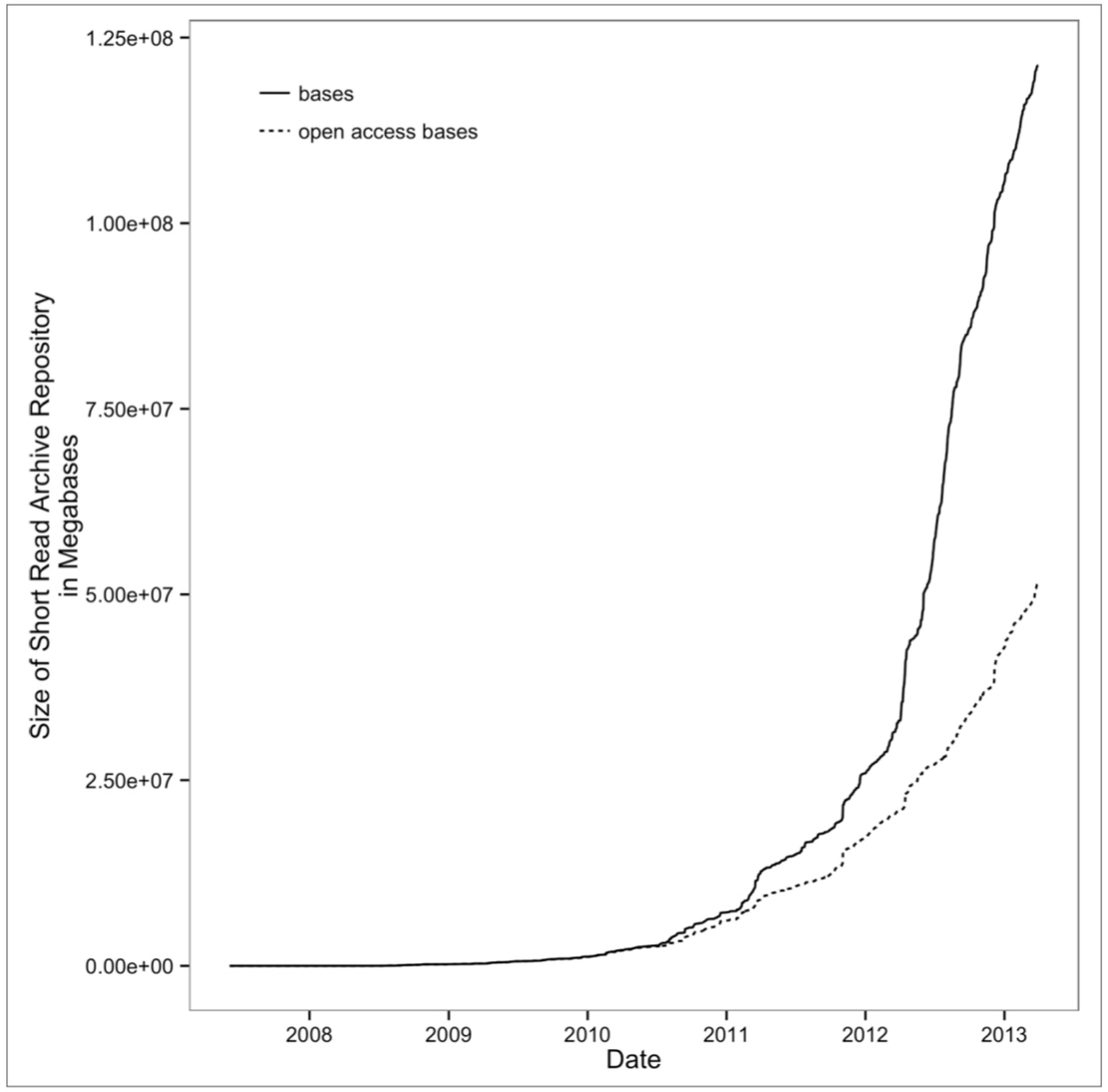


Figure 1-2. Exponential growth of the Short Read Archive; open access bases are SRA submissions available to the public (figure reproduced and data downloaded from the NIH)

Reproducible and Robust Research

New challenges

- Increasing complexity of bioinformatic analyses makes findings error-prone.
- To make things worse, some analyses are usually only run once before publication.
- Some analyses may rely on very specific versions of all software/systems used.

- Healthy skepticism: never trust your tools (or data).
- Keeping the analyses reproducible and robust.

Reproducible and Robust Research

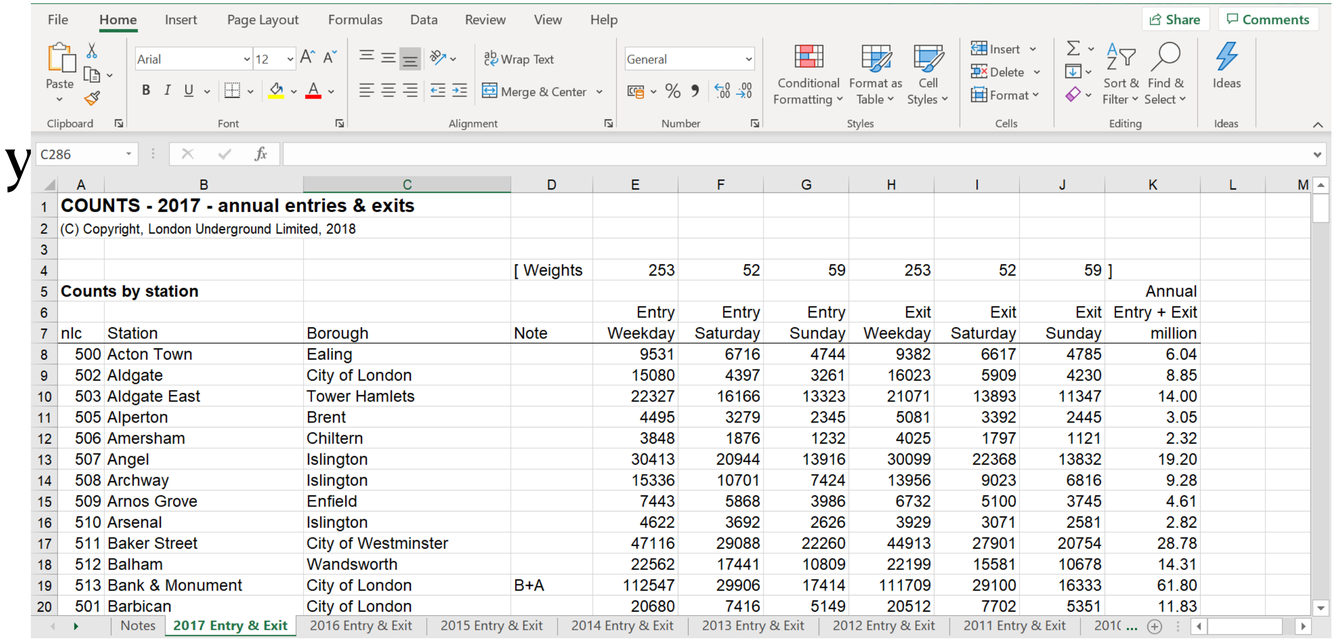
New challenges

- Every conclusion/claim should be **reproducible**:
  - Karl Popper, *The Logic of Scientific Discovery*: "*non-reproducible single occurrences are of no significance to science*" (1959)
- Adopting Robust and Reproducible Practices Will **Make Your Life Easier**, Too:
  - You will almost certainly have to **rerun** an analysis more than once, possibly with new or changed data.
  - In the future, you will almost certainly **forget** the details. Without writing down key facts (e.g., where you downloaded data from, when you downloaded it, and what steps you ran), you'll certainly forget them. Documenting your computational work is equivalent to keeping a detailed lab notebook—an absolutely crucial part of science.

Reproducible and Robust Research

Recommendation for robust research

- Pay attention to **experimental design**.
- Write **code** for humans, write **data** for computers.
  - Google style guide for coding
  - Consistent data format
- Let your **computer** do the work for y
- **Test** your code.
- Use existing **well vetted libraries**.
- Treat data as **Read-only**.



Reproducible and Robust Research

Recommendation for reproducible research

- **Release** your **code and data**.
- **Document everything**.
- Make figures and statistics the **results of scripts**.
- Use **code** as documentation.
- Check **data integrity**:
  - MD5 is a method that turns any piece of data into a unique, fixed-size "fingerprint" or hash, used to verify data integrity.
  - 2024-12-31: a448e7af4416d615121f68d83c954ec5
  - 51fcad48f76a22206b5b5e1dd5df4507 (Guess this one)

Setting Up and Managing a Bioinformatics Project

Project Directories Structures

Foundation of reproducible projects

- Your project directory should be laid out in a consistent and understandable fashion.
- Clear project organization makes it easier for both you and collaborators to figure out exactly where and what everything is.
- Additionally, it's much easier to automate tasks when files are organized and clearly named.
- Naming schemes.
- Leading zeros and sorting.

`snake_case`

Pros: Concise when it consists of a few words.
Cons: Redundant as hell when it gets longer.
`push_something_to_first_queue, pop_what, get_whatever…`

`PascalCase`

Pros: Seems neat.
`GetItem, SetItem, Convert, ...`
Cons: Barely used. (why?)

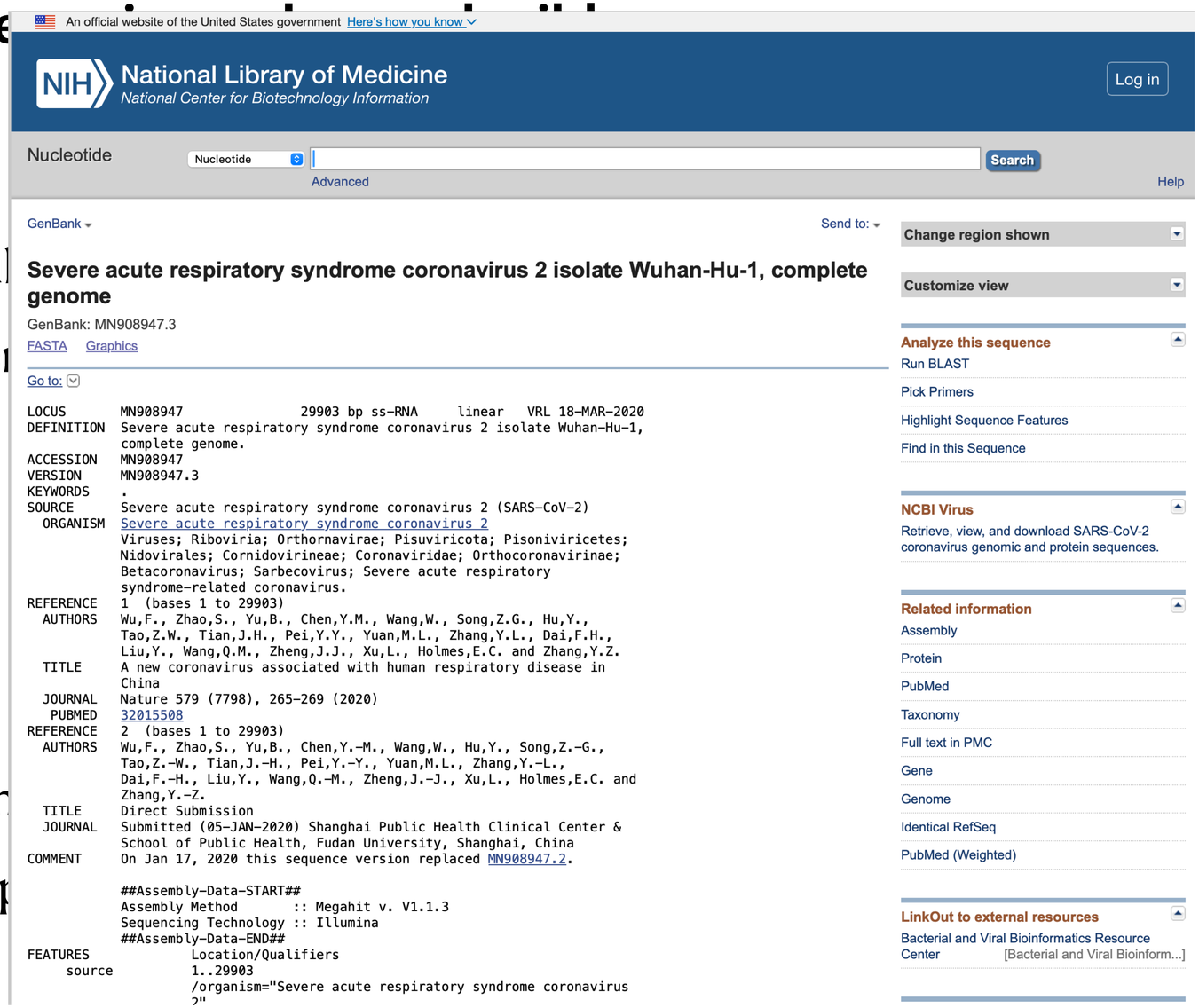
`camelCase`

Pros: Widely used in the programmer community.
Cons: Looks ugly when a few methods are n-worded.
`push, reserve, beginBuilding, ...`

Project documentation

Makes your life

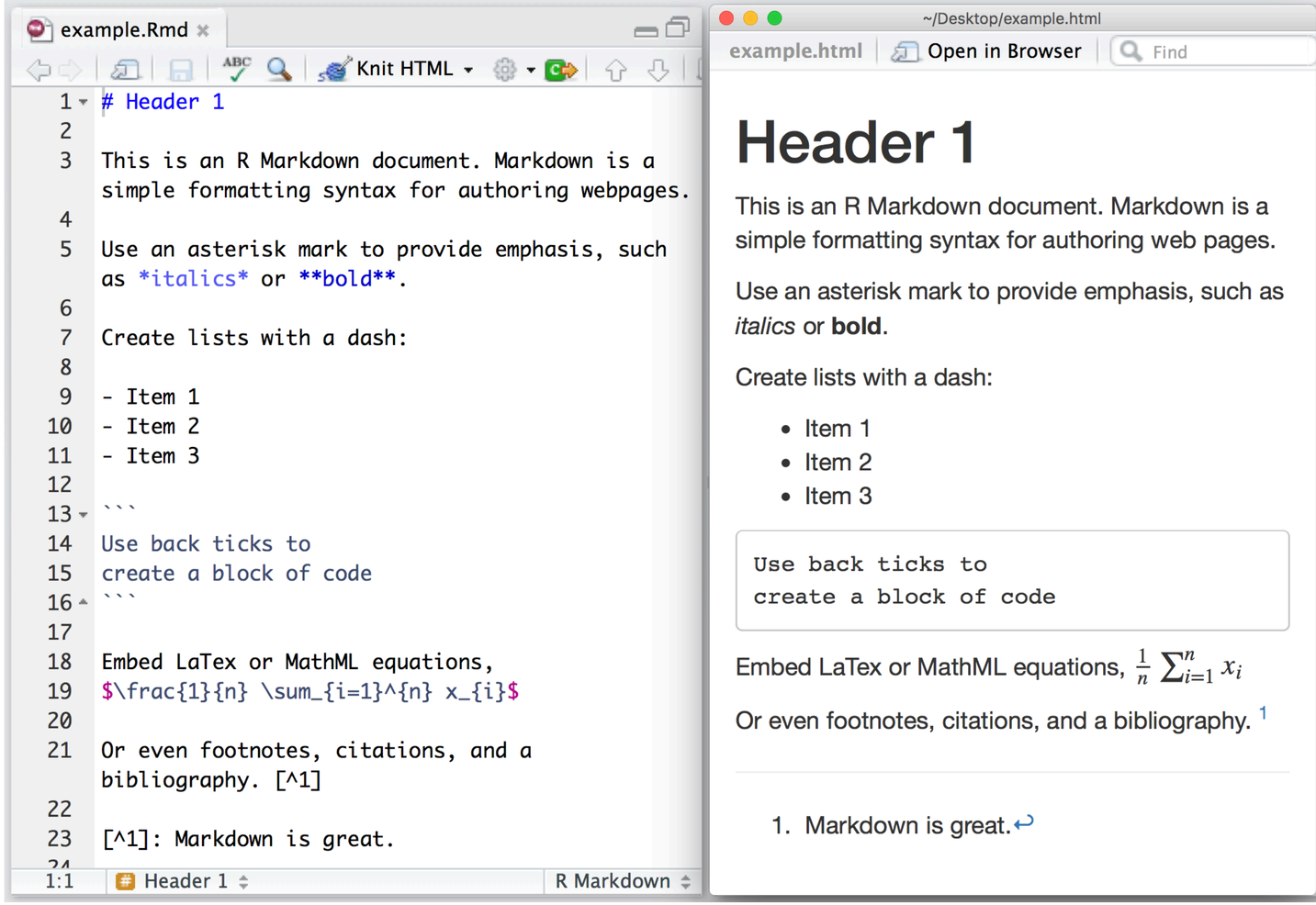
- Document your methods and workflows.
  - Full commands and values (even defaul
- Document the origin of all data in your pr
  - Any data you use to create output
- Document when you downloaded data.
- Record data version information.
- Describe how you downloaded the data.
- Document the versions of the software th
- All of these information is best stored in p



Markdown

For project notebooks

- Lightweight markup language for plain-text format.
- Can be easily rendered to HTML or PDF.
- Just plain text.
- Simple and elegant for notebook.



MD formatting basics

Cheatsheet

- Using text editor on your computer.
- Create a markdown file (regular text file with `md` suffix).
- Google “Markdown cheatsheet” and follow the syntax.
- Some basic syntax will be enough.

Heading	`# H1`
	`## H2`
	`### H3`

Bold	`**bold text**`
Italic	`*italicized text*`
Blockquote	`> blockquote`

Ordered List	`1. First item`
	`2. Second item`
	`3. Third item`

Unordered List	`- First item`
	`- Second item`
	`- Third item`

Code	`` `code` ``
Horizontal Rule	`---`
Link	`[title](https://www.example.com)`
Image	`![alt text](image.jpg)`

MD Trial

Try to replicate the notebook

- Online editor:
  - https://markdownlivepreview.com/
- Logo file:
  - https://hkuspace.hku.hk/assets/img/hkuspace-logo-2x.png
- Online to PDF:
  - https://www.markdowntopdf.com/

My first markdown notebook

Testing

I am **excited** to try *Markdown* for the first time.

Here are some notes:

- `*` is for *italic*.
- `**` is for **bold**.

Conclusion

Overall, Markdown is:

1. Simple;
2. Fun.

My first markdown notebook

Testing

I am **excited** to try *Markdown* for the first time.

Here are some notes:

- `*` is for *italic*.

- `**` is for **bold**.

Conclusion

Overall, Markdown is:

1. Simple;

2. Fun.

Markdown Live Preview Reset Copy Sync scroll

```
# My first markdown notebook

![](https://hkuspace.hku.hk/assets/img/hkuspace-logo-2x.png)

## Testing

I am **excited** to try *Markdown* for the first time.

Here are some notes:
- `*` is for *italic*.
- `**` is for **bold**.

## Conclusion

Overall, Markdown is:

1. Simple;
2. Fun.

```

My first markdown notebook

Testing

I am **excited** to try *Markdown* for the first time.

Here are some notes:

- `*` is for *italic*.
- `**` is for **bold**.

Conclusion

Overall, Markdown is:

1. Simple;
2. Fun.

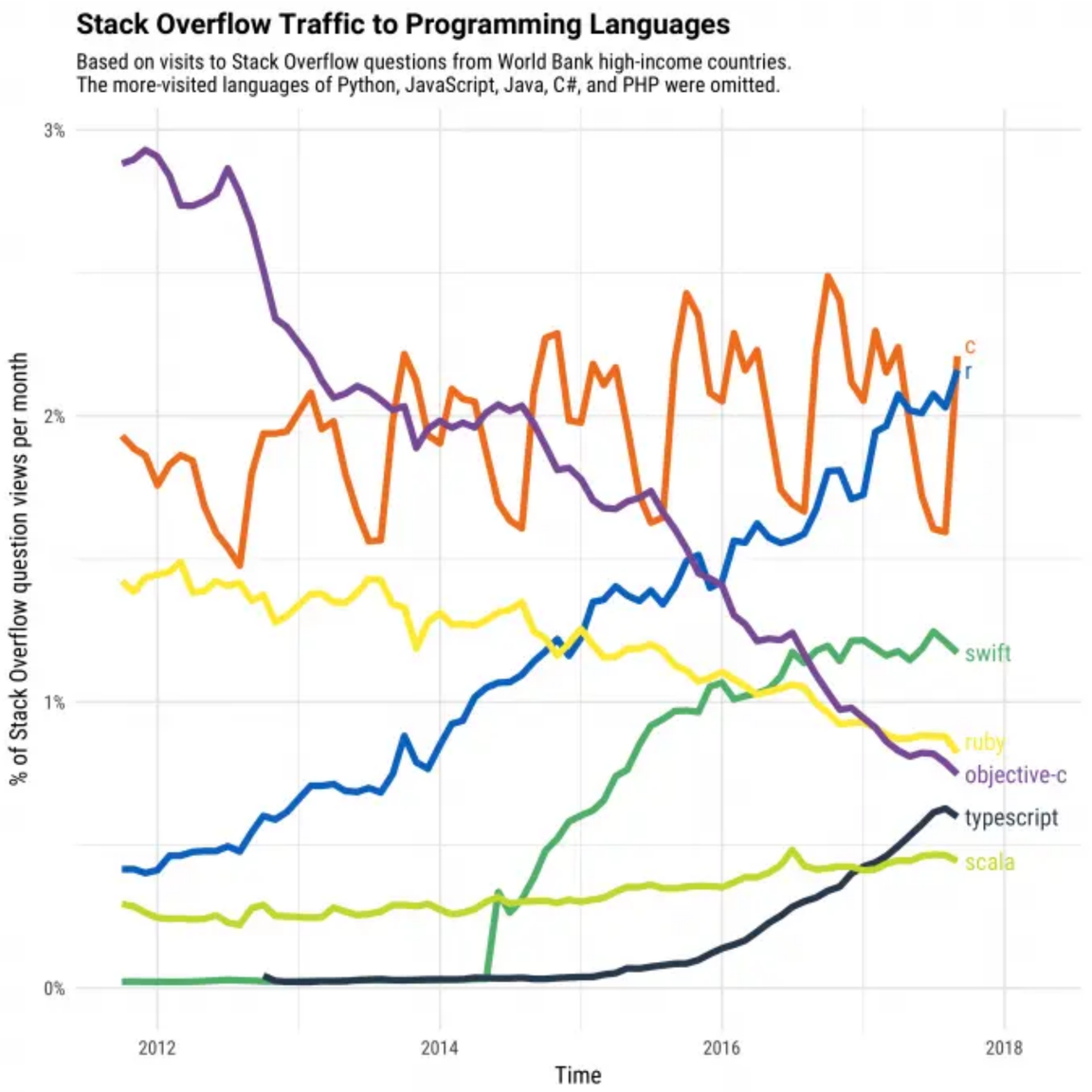
Break (10 mins)

A rapid introduction to the R language

R

Data analysis and visualisation

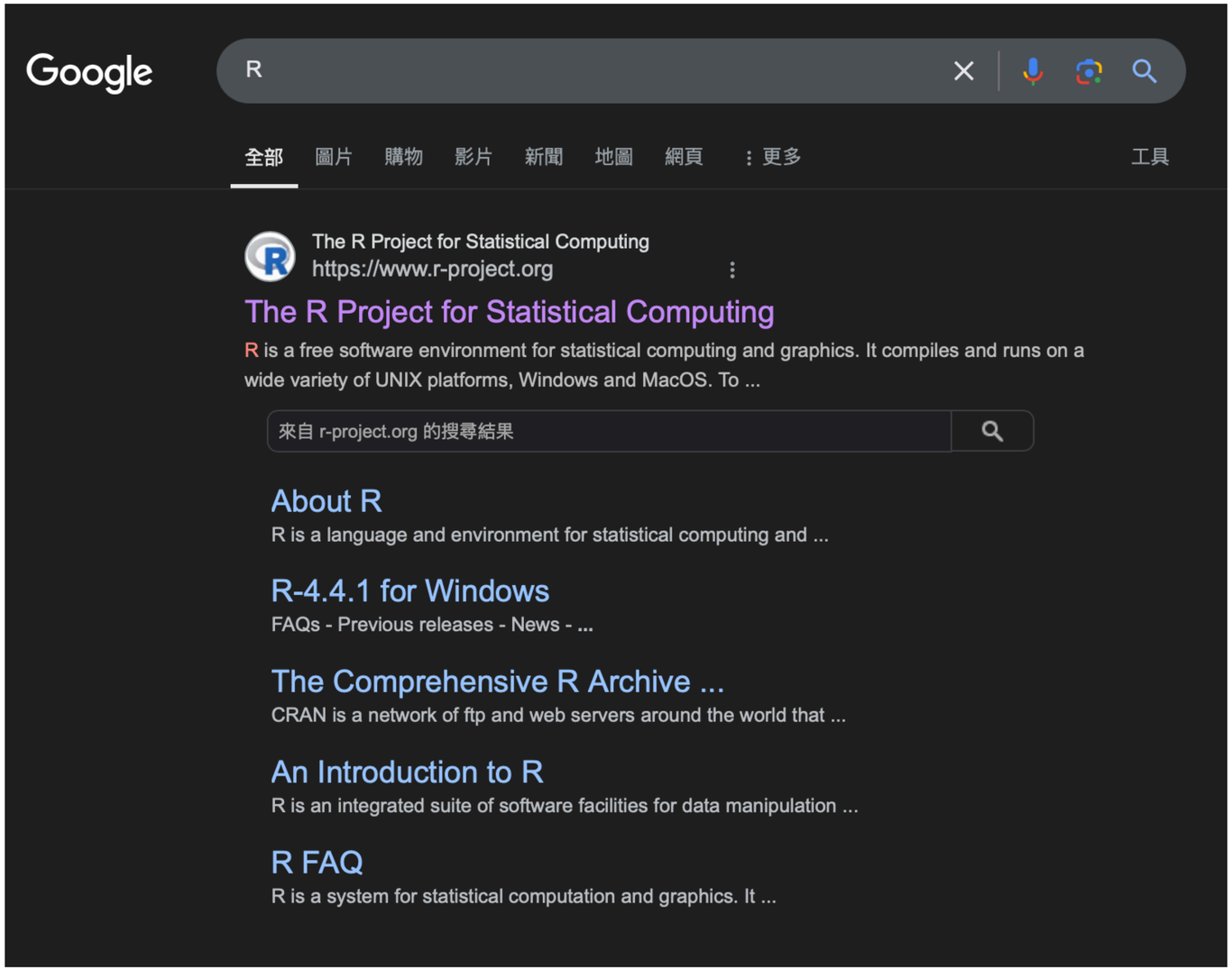
- How do you know R?
- Many bioinformaticians use R.
- Pros:
  - Easy!
  - Many existing packages.
- Cons:
  - Slow (relatively)



Running R

On your local machine

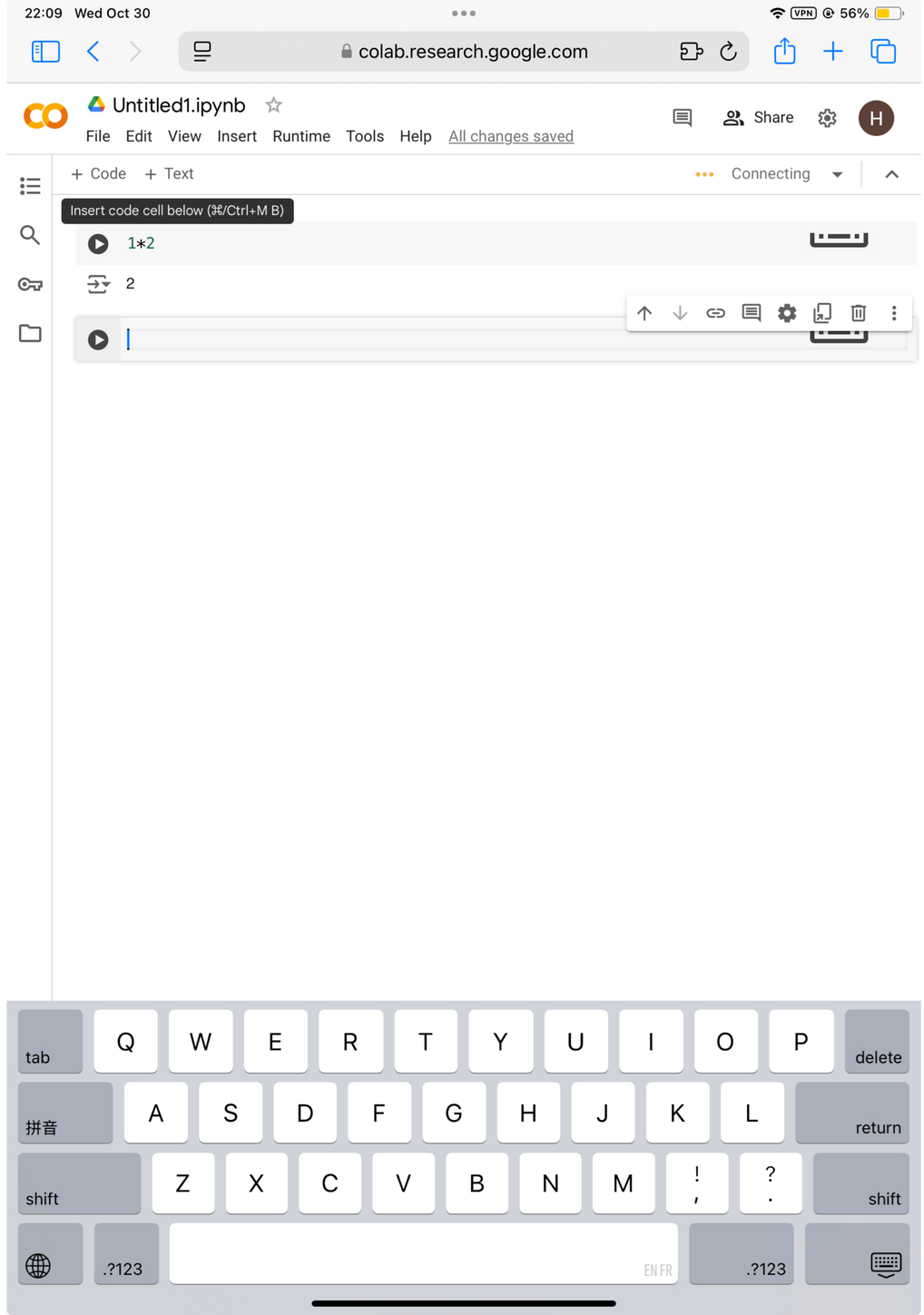
- To install R, go to the official website.
- Optionally you would also like to install the RStudio (an integrated development environment for R).
- Or you can use VSCode (like me).



Running R

Online

- Google colab.
  - Log in your account and create a new notebook.
  - Change the Runtime to R.
  - Raise your hand if you have problems.



R language basics

Simple calculations

Table 8-1. Common mathematic functions

Function Name	Description	Example
`exp(x)`	Exponential function	`exp(1), exp(2)`
`log(x, base=exp(1))`, `log10()`, `log2()`	Natural, base 10, and base 2 logarithms	`log(2)`, `log10(100)`, `log2(16)`
`sqrt(x)`	Square root	`sqrt(2)`
`sin(x)`, `cos(x)`, `tan(x)`, etc.	Trigonometric functions (see `help(sin)` for more)	`sin(pi)`
`abs(x)`	Absolute value	`abs(-3)`
`factorial(x)`	Factorial	`factorial(5)`
`choose(n, k)`	Binomial coefficient	`choose(5, 3)`

+ Code + Text

```
[3] 3+4
```

```
7
```

```
[4] 3-4
```

```
-1
```

```
[5] 4*3
    4/3
```

```
12
1.33333333333333
```

```
[6] 3+4/2
```

```
5
```

```
[7] (3+4)/2
```

```
3.5
```

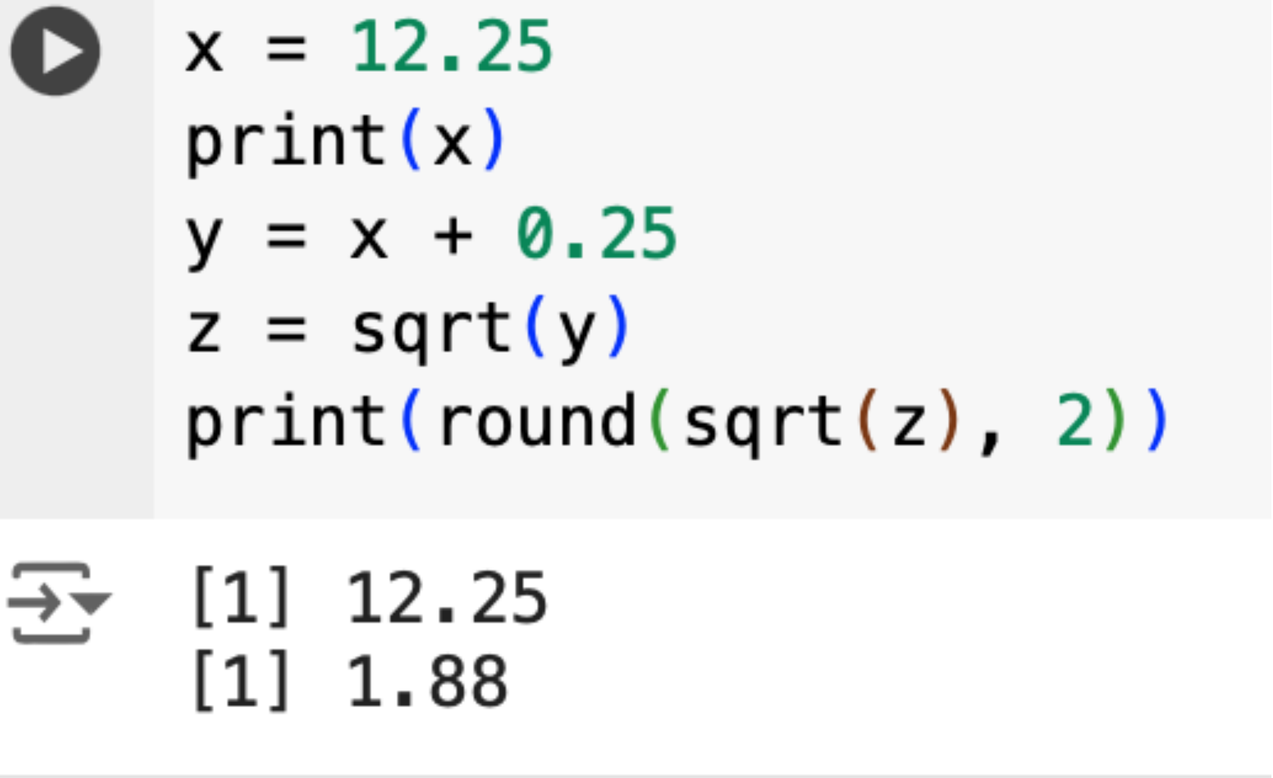
```
sqrt(9)
sqrt(2)
round(sqrt(2), digits = 3)
```

```
3
1.4142135623731
1.414
```

R language basics

Variables and Assignment

- Store a value 12.25 into a variable named `x`.
- Print `x`.
- x + 0.25, then times 2, store it into `y`.
- Square root y and store the result into `z`
- Print square root of `z` and keep 2 digits.



Vectors, Vectorization, and Indexing

R's feature

- Create a vector `x` containing three values (1, 9 and 36) using `c()`.
- Get the square root of `x` and store in `y`.
- Print x + y
- Print y
- Print the third component of y.
- Check whether the elements in y are greater than 4.
- Print the elements in y which is/are less than 5.

Table 8-2. R's comparison and logical operators

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!	Not equal to
&	Elementwise logical AND
|	Elementwise logical OR
!	Elementwise logical NOT
&&	Logical AND (first element only, for `if` statements)
||	Logical OR (first element only, for `if` statements)

Vectors, Vectorization, and Indexing

R's feature

- Create a vector `x` containing three values (1, 9 and 36) using `c()`.
- Get the square root of `x` and store in `y`.
- Print x + y
- Print y
- Print the third component of y.
- Check whether the elements in y are greater than 4.
- Print the elements in y which is/are less than 5.

0s

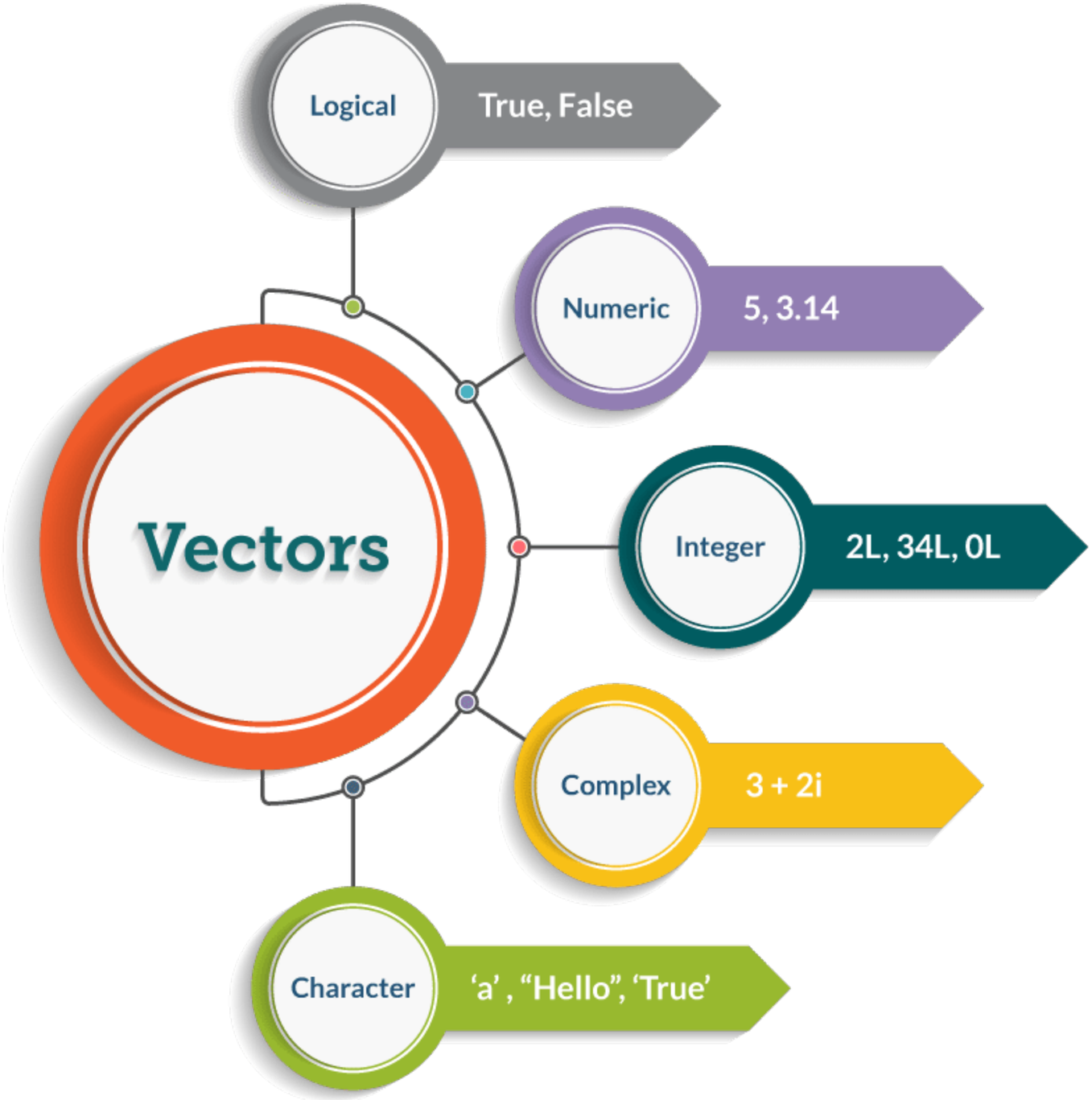
```r
x = c(1, 9, 36)
y = sqrt(x)
print(x + y)
print(y)
print(y[3])
print(y > 4)
print(y[y<5])
```

```
[1]  2 12 42
[1] 1 3 6
[1] 6
[1] FALSE FALSE  TRUE
[1] 1 3
```

Vector types

In R

- Numeric
- Integer
- Character
- Logical
- Factors (ordered levels)
- Special values in R:
  - NA
  - NULL
  - -Inf, Inf
  - NaN



Loading data

Into R

- data <- read_csv("https://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv")
- head()
- nrow()
- ncol()
- dim()
- colnames()

```
[8]  library(tidyverse)
     data <- read_csv("https://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv")

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
Rows: 15441 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (8): Report date, Date of onset, Gender, Age, Hospitalised/Discharged/De...
dbl (1): Case no.
lgl (1): Name of hospital admitted

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

https://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv

Loading data

Into R

- data <- read_csv("https://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv")
- head()
- nrow()
- ncol()
- dim()
- colnames()

```
[10] head(data)
```

A tibble: 6 × 10

Case no.	Report date	Date of onset	Gender	Age	Name of hospital admitted	Hospitalised/Discharged/Deceased	HK/Non-HK resident	Classifica
<dbl>	<chr>	<chr>	<chr>	<chr>	<lgl>	<chr>	<chr>	
1	23/01/2020	21/01/2020	M	39	NA	Discharged	Non-HK resident	Import
2	23/01/2020	18/01/2020	M	56	NA	Discharged	HK resident	Import
3	24/01/2020	20/01/2020	F	62	NA	Discharged	Non-HK resident	Import
4	24/01/2020	23/01/2020	F	62	NA	Discharged	Non-HK resident	Import
5	24/01/2020	23/01/2020	M	63	NA	Discharged	Non-HK resident	Import
6	26/01/2020	21/01/2020	M	47	NA	Discharged	HK resident	Import

Loading data

Into R

- data <- read_csv("https://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv")
- head()
- nrow()
- ncol()
- dim()
- colnames()

```
[11] nrow(data)
     ncol(data)
     dim(data)
```

```
15441
10
15441 · 10
```

```
[12] colnames(data)
```

```
'Case no.' · 'Report date' · 'Date of onset' · 'Gender' · 'Age' · 'Name of hospital admitted' · 'Hospitalised/Discharged/Deceased' · 'HK/Non-HK resident' · 'Classification*' · 'Case status*'
```

Manipulating columns

Change character to date

- Extract a column: `Report date`
- head()
- typeof()

```
typeof(data$`Report date`)
```

Run cell (⌘/Ctrl+Enter)
cell executed since last change

```
data$`Report date`
```

```
'23/01/2020' · '23/01/2020' · '24/01/2020' · '24/01/2020' · '24/01/2020' · '26/01/2020' · '26/01/2020' · '26/01/2020' · '29/01/2020' · '29/01/2020' ·
'30/01/2020' · '30/01/2020' · '31/01/2020' · '01/02/2020' · '01/02/2020' · '04/02/2020' · '04/02/2020' · '04/02/2020' · '05/02/2020' · '05/02/2020' ·
'05/02/2020' · '06/02/2020' · '06/02/2020' · '06/02/2020' · '07/02/2020' · '07/02/2020' · '09/02/2020' · '09/02/2020' · '09/02/2020' · '09/02/2020' ·
'09/02/2020' · '09/02/2020' · '09/02/2020' · '09/02/2020' · '09/02/2020' · '09/02/2020' · '10/02/2020' · '10/02/2020' · '10/02/2020' · '10/02/2020' ·
'10/02/2020' · '10/02/2020' · '11/02/2020' · '11/02/2020' · '11/02/2020' · '11/02/2020' · '11/02/2020' · '11/02/2020' · '11/02/2020' · '12/02/2020' ·
'13/02/2020' · '13/02/2020' · '13/02/2020' · '14/02/2020' · '14/02/2020' · '14/02/2020' · '16/02/2020' · '17/02/2020' · '17/02/2020' · '17/02/2020' ·
'18/02/2020' · '18/02/2020' · '19/02/2020' · '19/02/2020' · '19/02/2020' · '20/02/2020' · '20/02/2020' · '20/02/2020' · '20/02/2020' · '22/02/2020' ·
'23/02/2020' · '23/02/2020' · '23/02/2020' · '23/02/2020' · '24/02/2020' · '24/02/2020' · '24/02/2020' · '24/02/2020' · '24/02/2020' · '24/02/2020' ·
'24/02/2020' · '25/02/2020' · '25/02/2020' · '25/02/2020' · '25/02/2020' · '26/02/2020' · '26/02/2020' · '26/02/2020' · '26/02/2020' · '26/02/2020' ·
'26/02/2020' · '27/02/2020' · '27/02/2020' · '28/02/2020' · '29/02/2020' · '01/03/2020' · '01/03/2020' · '01/03/2020' · '01/03/2020' · '01/03/2020' ·
'02/03/2020' · '04/03/2020' · '04/03/2020' · '04/03/2020' · '04/03/2020' · '06/03/2020' · '06/03/2020' · '06/03/2020' · '07/03/2020' · '07/03/2020' ·
'08/03/2020' · '08/03/2020' · '08/03/2020' · '08/03/2020' · '08/03/2020' · '09/03/2020' · '10/03/2020' · '10/03/2020' · '10/03/2020' · '10/03/2020' ·
'10/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' · '11/03/2020' ·
'12/03/2020' · '12/03/2020' · '13/03/2020' · '13/03/2020' · '13/03/2020' · '13/03/2020' · '13/03/2020' · '13/03/2020' · '14/03/2020' · '14/03/2020' ·
'14/03/2020' · '14/03/2020' · '15/03/2020' · '15/03/2020' · '15/03/2020' · '15/03/2020' · '15/03/2020' · '15/03/2020' · '15/03/2020' · '16/03/2020' ·
'16/03/2020' · '16/03/2020' · '16/03/2020' · '16/03/2020' · '16/03/2020' · '16/03/2020' · '16/03/2020' · '16/03/2020' · '17/03/2020' · '17/03/2020' ·
'17/03/2020' · '17/03/2020' · '17/03/2020' · '17/03/2020' · '17/03/2020' · '17/03/2020' · '17/03/2020' · '17/03/2020' · '18/03/2020' · '18/03/2020' ·
'18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' ·
'18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' · '18/03/2020' ·
'18/03/2020' · '18/03/2020' · '18/03/2020' · '19/03/2020' · '19/03/2020' · '19/03/2020' · '19/03/2020' · '19/03/2020' · '19/03/2020' · '19/03/2020' ·
··· · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '13/01/2022' · '24/01/2022' · '25/01/2022' · '28/01/2022' · '29/01/2022' · '31/01/2022' ·
'31/01/2022' · '01/02/2022' · '01/02/2022' · '05/02/2022' · '05/02/2022' · '05/02/2022' · '05/02/2022' · '05/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' · '06/02/2022' ·
'06/02/2022'
```

```
[14] head(data$`Report date`)
```

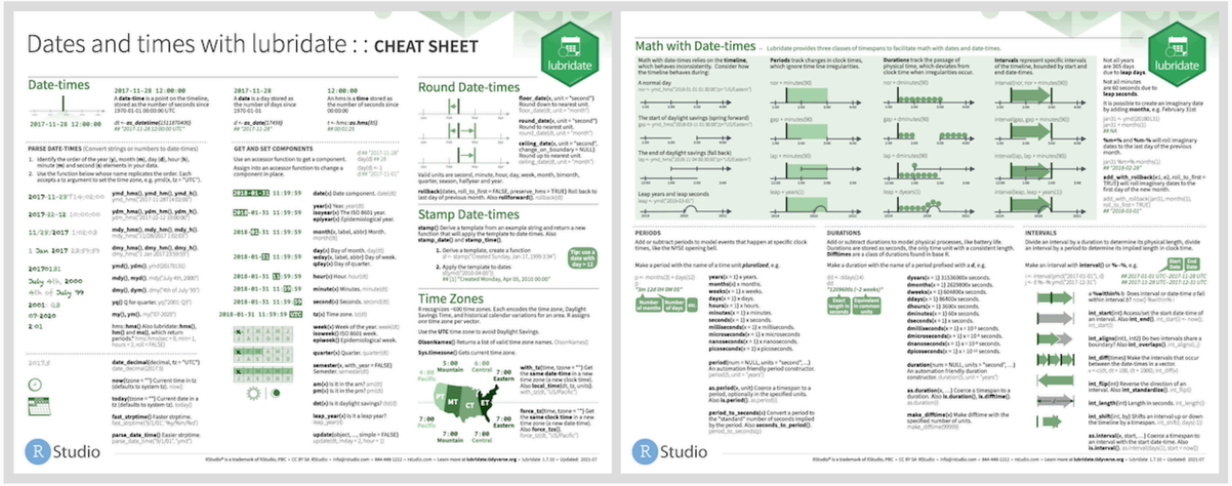
```
'23/01/2020' · '23/01/2020' · '24/01/2020' · '24/01/2020' · '24/01/2020' · '26/01/2020'
```

Manipulating columns

Change character to date

- Change the character type to a date type using lubridate.

Cheatsheet



lubridate 1.9.2 Get started Reference News Search for

lubridate

Overview

Date-time data can be frustrating to work with in R. R commands for date-times are generally unintuitive and change depending on the type of date-time object being used. Moreover, the methods we use with date-times must be robust to time zones, leap days, daylight savings times, and other time related quirks, and R lacks these capabilities in some situations. Lubridate makes it easier to do the things R does with date-times and possible to do the things R does not.

If you are new to lubridate, the best place to start is the date and times chapter in R for data science.

Installation

```
# The easiest way to get lubridate is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just lubridate:
install.packages("lubridate")

# Or the the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/lubridate")
```

LINKS

View on CRAN
Browse source code
Report a bug
Learn more

LICENSE

Full license
GPL (>= 2)

COMMUNITY

Contributing guide
Code of conduct
Getting help

CITATION

Citing lubridate

DEVELOPERS

Vitalie Spinu
Author, maintainer

Garrett Grolemund
Author

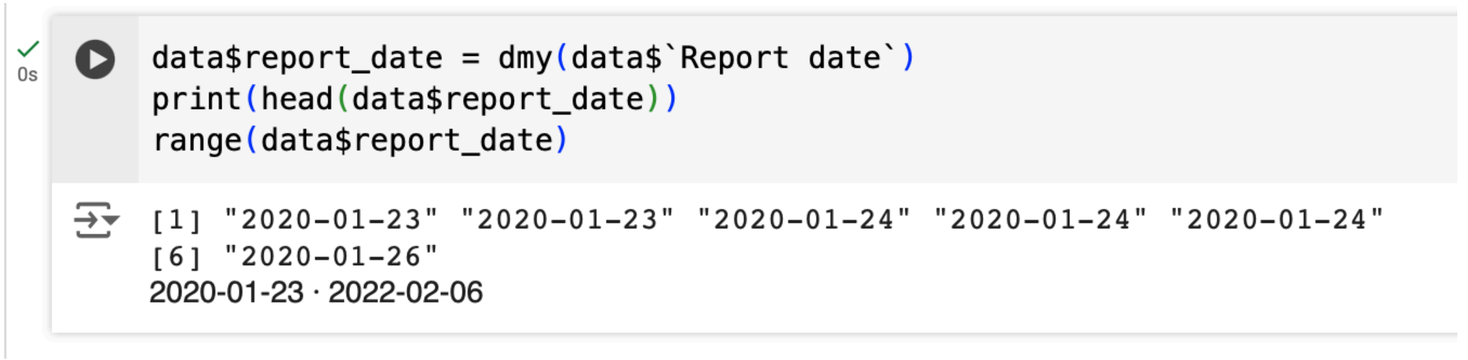
Hadley Wickham
Author

More about authors...

Manipulating columns

Change character to date

- Change the character type to a date type using lubridate.
- Create a new column called `report_date` and assign it with a date-type version of the `Report date` column.
- Hint: ?dmy()
- dmy(): Transforms dates stored in character and numeric vectors to Date



Manipulating columns

Summarising the daily cases

- Group by report_date
- Summarise how many cases per group (per day in this case).
- data %>% group_by(report_dat %>% summarise(total_cases=n(
- data_daily_cases = data %>% group_by(report_date) %>% summarise(total_cases=n())

```
data %>% group_by(report_date) %>% summarise(total_cases=n())
```

A tibble: 687 × 2

report_date	total_cases
<date>	<int>
2020-01-23	2
2020-01-24	3
2020-01-26	3
2020-01-29	2
2020-01-30	2
2020-01-31	1
2020-02-01	2
2020-02-04	3
2020-02-05	3
2020-02-06	3
2020-02-07	2
2020-02-09	10

```
[22] data_daily_cases = data %>% group_by(report_date) %>% summarise(total_cases=n())
```

Data visualisation

In R using ggplot2

ggplot2

10 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

ggplot2 is an open-source data visualization package for the statistical programming language R. Created by Hadley Wickham in 2005, ggplot2 is an implementation of Leland Wilkinson's *Grammar of Graphics*—a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. ggplot2 can serve as a replacement for the base graphics in R and contains a number of defaults for web and print display of common scales. Since 2005, ggplot2 has grown in use to become one of the most popular R packages.[2][3][4]

ggplot2

Original author(s)	Hadley Wickham, Winston Chang
Initial release	10 June 2007; 17 years ago
Stable release	3.5.1[1] / 23 April 2024; 6 months ago
Repository	github.com/tidyverse/ggplot2
Written in	R
License	MIT license
Website	ggplot2.tidyverse.org

Updates [edit]

On 2 March 2012, ggplot2 version 0.9.0 was released with numerous changes to internal organization, scale construction and layers.[5]

On 25 February 2014, Hadley Wickham formally announced that "ggplot2 is shifting to maintenance mode. This means that we are no longer adding new features, but we will continue to fix major bugs, and consider new features submitted as pull requests. In recognition [of] this significant milestone, the next version of ggplot2 will be 1.0.0".[6]

On 21 December 2015, ggplot 2.0.0 was released. In the announcement, it was stated that "ggplot2 now has an official extension mechanism. This means that others can now easily create their [own] stats, geoms and positions, and provide them in other packages."[7]

Data visualization with ggplot2 : : CHEATSHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5**)** Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes

Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

size - integer (in mm for size of points and text)

linewidth - integer (in mm for widths of lines)

shape - integer/shape name or a single character ("a")

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1**)** x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50**))** - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1**))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900**))** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1**))**
b + geom_hline(aes(yintercept = lat**))**
b + geom_vline(aes(xintercept = long**))**

b + geom_segment(aes(yend = lat + 1, xend = long + 1**))**
b + geom_spoke(aes(angle = 1:1155, radius = 1**))**

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin"**)** x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian"**)** x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot() x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy**))** x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES both continuous

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point() x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile() x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)** x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)** x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

f <- ggplot(mpg, aes(class, hwy))

f + geom_col() x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot() x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"**)** x, y, alpha, color, fill, group

f + geom_violin(scale = "area"**)** x, y, alpha, color, fill, group, linetype, size, weight

both discrete

g <- ggplot(diamonds, aes(cut, color))

g + geom_count() x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)** x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d() x, y, alpha, color, group, linetype, size

h + geom_hex() x, y, alpha, color, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv"**)** x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

j + geom_crossbar(fatten = 2**)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width Also **geom_errorbarh()**.

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

Draw the appropriate geometric object depending on the simple features present in the data. aes() arguments: map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf"**))**

ggplot(nc) +
geom_sf(aes(fill = AREA**))**

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z)**)** x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)**)** x, y, alpha, color, fill, group, linetype, size, subgroup

l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE**)** x, y, alpha, fill

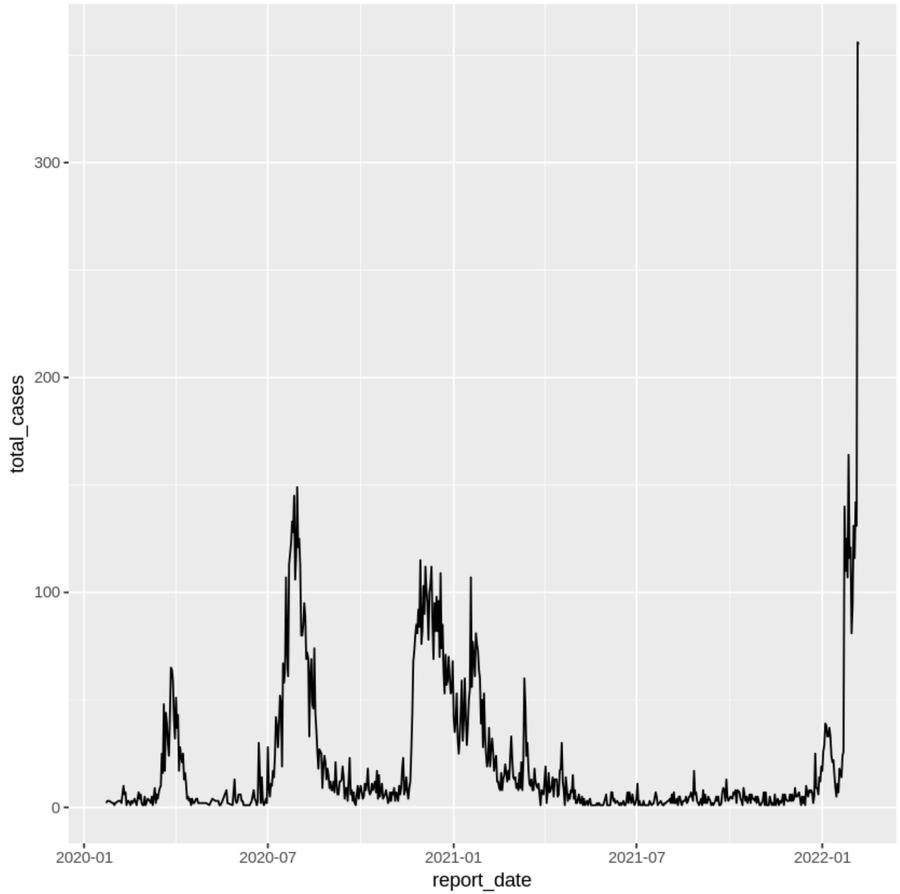
l + geom_tile(aes(fill = z)**)** x, y, alpha, color, fill, linetype, size, width

Data visualisation

In R using ggplot2

- library(ggplot2)
- Line plot showing cases v.s. date
  - X-axis: report_date
  - Y-axis: total_cases
- ggplot(data_daily_cases) + geom_line(aes(x=report_date, y=total_cases))

```
[23] library(ggplot2)
     ggplot(data_daily_cases) + geom_line(aes(x=report_date, y=total_cases))
```



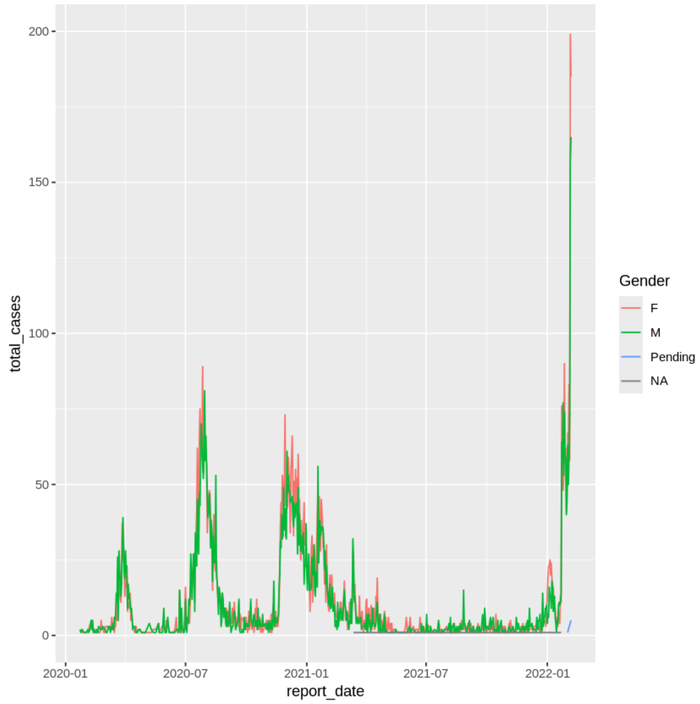
Data visualisation

In R using ggplot2

- data_daily_cases_gender = data %>% group_by(report_date, Gender) %>% summarise(total_cases=n())
- ggplot(data_daily_cases_gender) + geom_line(aes(x=report_date, y=total_cases, color=Gender))

```
[27] data_daily_cases_gender = data %>% group_by(report_date, Gender) %>% summarise(total_cases=n())
     ggplot(data_daily_cases_gender) + geom_line(aes(x=report_date, y=total_cases, color=Gender))
```

```
`summarise()` has grouped output by 'report_date'. You can override using the
`.groups` argument.
```



Data visualisation

In R using ggplot2

- Boxplot showing Age distribution between genders
- The `Age` column is now character type.
- as.numeric()

```
head(data)
```

A tibble: 6 × 11

Case no.	Report date	Date of onset	Gender	Age	Name of hospital admitted	Hospitalised/Discharged/Deceased	HK/No reside
<dbl>	<chr>	<chr>	<chr>	<chr>	<lgl>	<chr>	<ch
1	23/01/2020	21/01/2020	M	39	NA	Discharged	Non-I reside
2	23/01/2020	18/01/2020	M	56	NA	Discharged	I reside
3	24/01/2020	20/01/2020	F	62	NA	Discharged	Non-I reside
4	24/01/2020	23/01/2020	F	62	NA	Discharged	Non-I reside
5	24/01/2020	23/01/2020	M	63	NA	Discharged	Non-I reside
6	26/01/2020	21/01/2020	M	47	NA	Discharged	I reside

Data visualisation

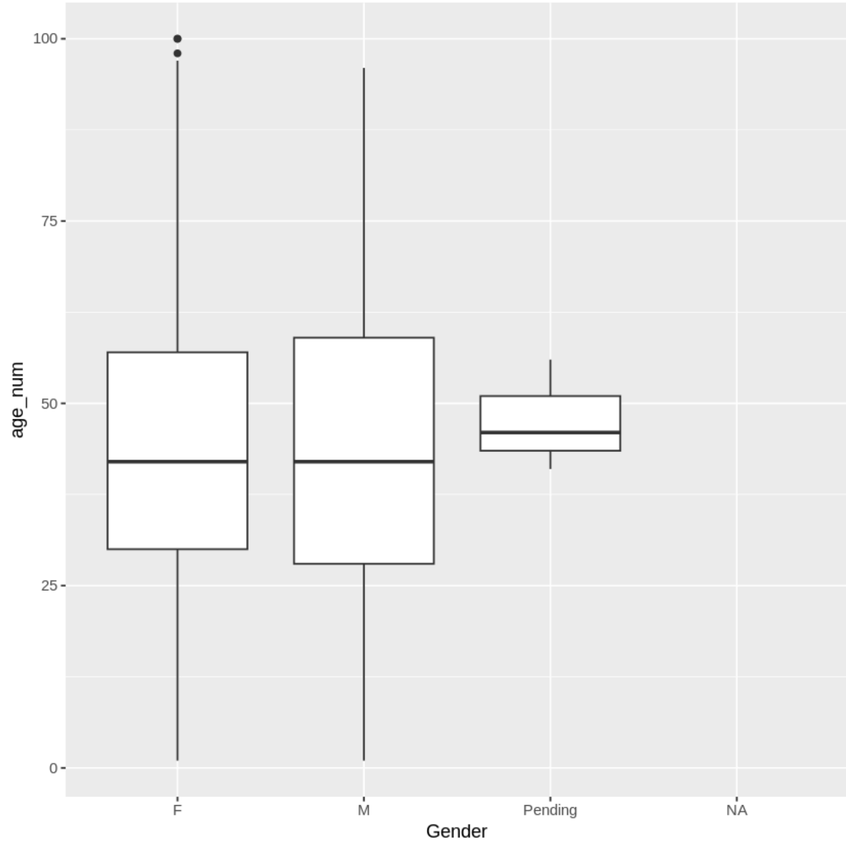
In R using ggplot2

- Boxplot showing Age distribution between genders
- data$age_num = as.numeric(data$Age)
- ggplot(data) + geom_boxplot(aes(x=Gender, y=age_num))

```
data$age_num = as.numeric(data$Age)
ggplot(data) + geom_boxplot(aes(x=Gender, y=age_num))
```

0s

```
Warning message:
"NAs introduced by coercion"
Warning message:
"Removed 94 rows containing non-finite outside the scale range
(`stat_boxplot()`)."
```



Data visualisation

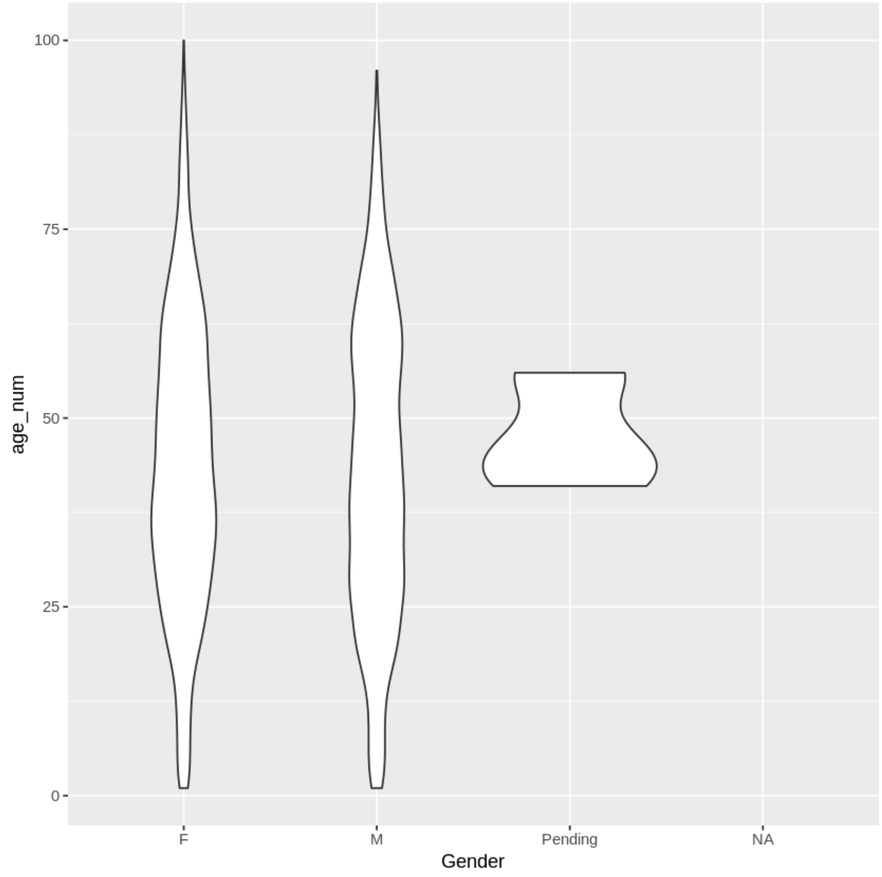
In R using ggplot2

- Boxplot showing Age distribution between genders
- data$age_num = as.numeric(data$Age)
- ggplot(data) + geom_boxplot(aes(x=Gender, y=age_num))

0s

```
data$age_num = as.numeric(data$Age)
# ggplot(data) + geom_boxplot(aes(x=Gender, y=age_num))
ggplot(data) + geom_violin(aes(x=Gender, y=age_num))
```

```
Warning message:
“NAs introduced by coercion”
Warning message:
“Removed 94 rows containing non-finite outside the scale range
(`stat_ydensity()`).”
```



End of lesson I