# MY472 – Week 1: Introduction

Thomas Robinson

Course website: lse-my472.github.io

# What is this course about?

The 80/20 rule of data science:
80% data manipulation, 20% data analysis


It is about the 80%

# In more detail

Course tries to provide "data science literacy"

- What is data?
  - Basic data types and structures
- How to collect data?
  - How to scrape data from the internet
  - How to work with APIs
- How to clean and process data?
  - How to format, organize, and reshape data
  - Cloud computing to process very large datasets
- How to store and query data?
  - How to create and use databases
  - How to create and manage (online) databases

# Plan for today

- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- Markdown in brief
- git and Github for version control

# Course outline

1. Introduction to data
2. The shape of data
3. Data visualisation
4. Textual data
5. HTML, CSS, and scraping static pages
6. (*Reading week*)
7. XML, RSS, and scraping non-static pages
8. Working with APIs
9. Creating and managing databases
10. Interacting with online databases
11. Cloud computing

# Prerequisites and software

- Introductory course – no prerequisites (only completion of R preparatory course required!)
- Lab computers are available, but we strongly recommend bringing your own laptop
- Software:
  - R 4.3.1 – Install from https://www.r-project.org/
  - RStudio – Install from https://www.rstudio.com/products/rstudio/download/
  - GitHub Desktop – Install from https://desktop.github.com/

  → *Please install before lab session this week*
- Mirrors similar tool usage and learning in other Methodology courses

# About me

- Assistant Professor at the London School of Economics
  - PhD in Politics, University of Oxford, 2020
- Research:
  - Use of machine learning in experimental designs
  - Corruption, money in politics, voter behavior
  - Causal inference methodology: treatment effect heterogeneity, missing data etc.
- Contact:
  - `t.robinson7@lse.ac.uk`
  - `https://ts-robinson.com`

# Your turn

1. Name?
2. MSc/PhD Programme?
3. Previous experience with R?
4. Why are you interested in this course and what would you most like to learn?

# Course philosophy

How to learn the techniques in this course?

- Lecture approach: not ideal for learning how to code
- You can only learn by doing

→ We will cover each concept three times during each week

1. Introduction to the topic in lecture
2. Guided coding session in lecture and lab
3. Course assignments

- We will move relatively fast

# Readings

Course webpage: https://lse-my472.github.io/

- Mixed set of readings, very specific to each week
  - Often freely available online, otherwise, available for purchase (often in electronic versions)
  - Some books are (freely) available online and in print, and the online version may be more recent
- Please do the readings!

# Course meetings

- Weekly lectures
- Ten one-hour classes ("labs") *starting this week*
  - Group 1: Thursdays 13:00–14:00 (CKK.2.13)
  - Group 3: Thursdays 14:00–15:00 (CKK.2.18)
  - Group 2: Thursdays 17:00–18:00 (CKK.1.09)
- No lecture/class in Week 6
- Office hours (book via StudentHub)

# Assessment

- 1 practise problem set
  - Opportunity to practise format and style of response
  - Due Thursday 12 October, 16:00
- 2 further problem sets will be assessed (50% in total)
  - Submitted via Moodle
  - Only "knitted" R-markdown assignments in HTML accepted
  - Due 2 November and 7 December 2023, 16:00
- Take-home assessment (50% in total)
  - A collaborative project undertaken over winter holidays
  - Deadline: 10 January 2024, 16:00

# A note on collaboration

- All assignments are individual unless we instruct you otherwise
- For individual assignments:
  - You can discuss solutions with peers
  - However, you are not allowed to copy-paste someone else's code
  - Submissions with identical code (where we shouldn't expect to see it) will be considered plagiarism
- You can use online resources but always give credit in comments if you borrow code/solutions

# ChatGPT (and other generative assistants)

We will **allow** ChatGPT to be used for assignments

- Ignoring the presence/possibilities of ChatGPT is unwise
- An opportunity to *learn* how to integrate these tools into your workflow

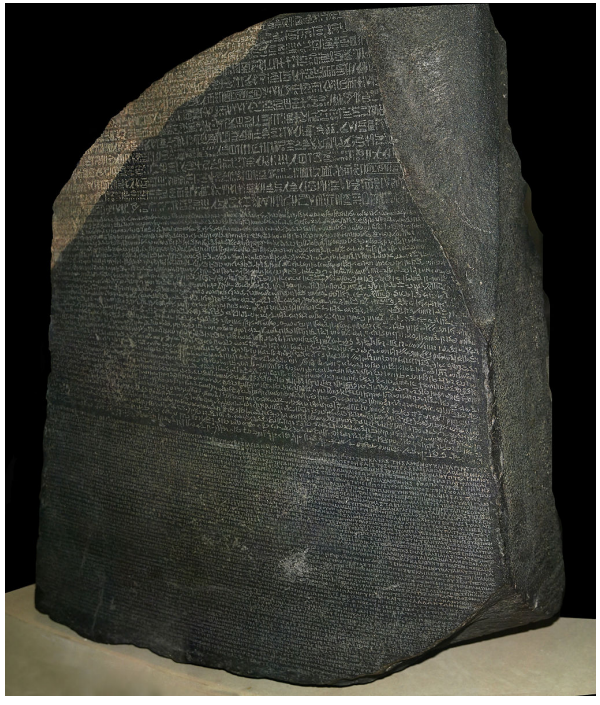
But beware:

- Often you need some proficiency to recognise "good" code or fix broken code
- We are *all* learning about these tools, and there remains huge uncertainty
- The leading models are proprietary

# Plan for today

- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- Markdown in brief
- git and GitHub for version control

# History of data



Rosetta Stone, British Museum

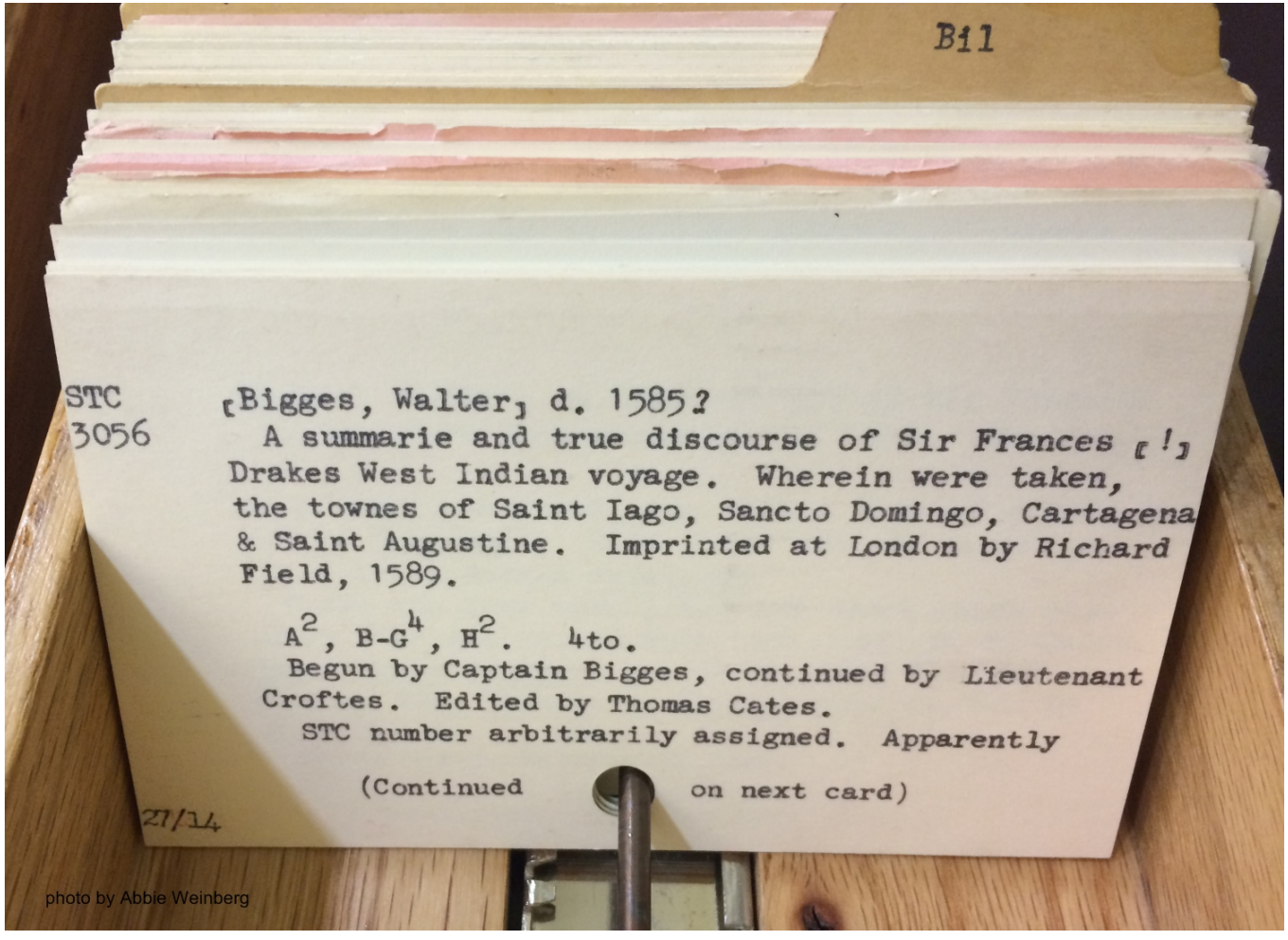
- Great book on the history of information and data: The Information by James Gleick (not on the formal reading list)
- Early example of database often government records: Who is paying taxes and how much, census of citizens, etc.

# Early example of a database index

Index cards used in a library catalog of books

- Initially developed to catalog species by botanist Carl Linnaeus (19th century)
- Units (species, books) are a record; records are *indexed* using a specific reference / sorting system

Each record looked like this:

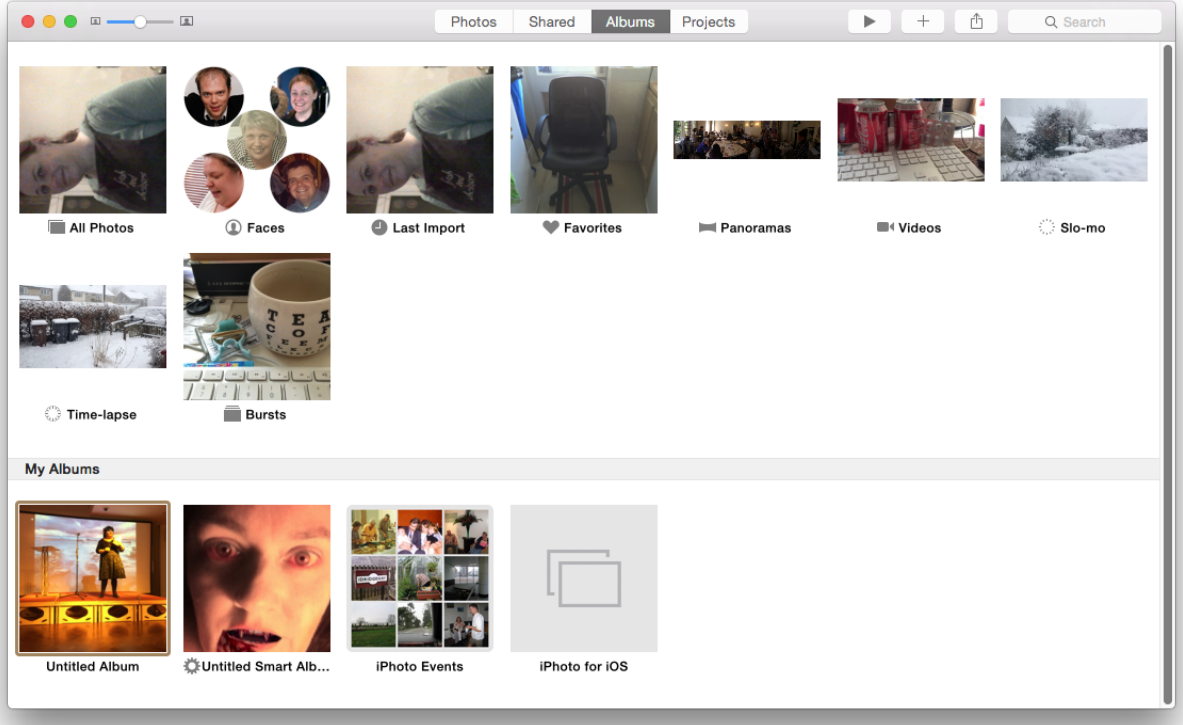


# Dewey decimal system

- A proprietary library classification system first published in the United States by Melvil Dewey in 1876
- Scheme is made up of ten classes, each divided into ten divisions, each having ten sections
- The system's notation uses Arabic numbers, with three whole numbers making up the main classes and sub-classes and decimals creating further divisions
- Example:

```
500 Natural sciences and mathematics
   510 Mathematics
      516 Geometry
         516.3 Analytic geometries
            516.37 Metric differential geometries
               516.375 Finsler Geometry
```

- Problem: Cards only sorted in one way. Re-referencing literally a manual operation
- Contrast with the idea of electronic indices, where assets are stored once and many indexing and referencing systems can be applied



# Relational databases

- Codd, E.F. (1970) "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*.

**School Table**

| ID | Name |
|---|---|
| **S001** | University of Technology |
| **S002** | University of Applied Science |

**Student Table**

| School ID | ID | Name | DOB |
|---|---|---|---|
| **S001** | UT-1000 | Tommy | 05/06/1995 |
| **S001** | UT-1000 | Better | 16/04/1995 |
| **S002** | UAS-1000 | Linda | 02/09/1995 |
| **S002** | UAS-1000 | Jonathan | 22/06/1995 |

# Recent developments in data storage/management

- **NoSQL**: beyond relational structure; flexible; more scalable & compatible with distributed cloud storage (Big Data)

# Trying to define Big Data

1. Volume: Around 8 billion mobile phones, around 2.5 billion Facebook users, 500+ million tweets per day...
2. Velocity: How quickly is data flowing? Personal, spatial and temporal granularity
3. Variability: Images, networks, long and short text, geographic coordinates, streaming...

Dumbill (2012), Monroe (2013)

Big Data: Data that is so large, complex, and/or variable that some new tools to understand it must be created

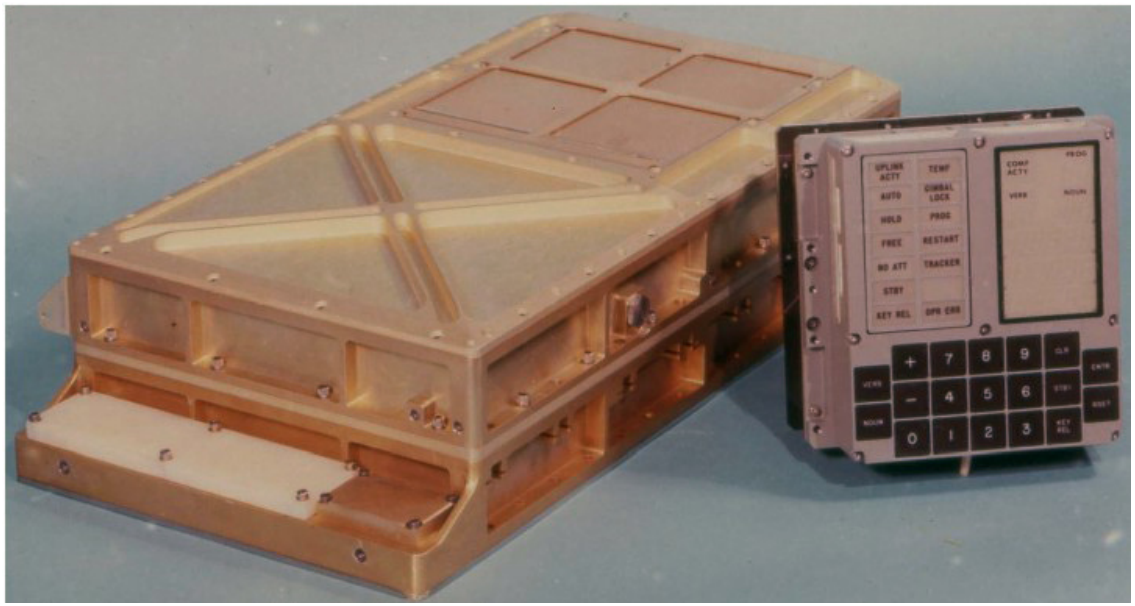
# Plan for today

- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- Markdown in brief
- git and Github for version control

# Changes in the world of data

- Volume of data in the modern world: Very large fraction of the world's data has been generated in the last *two years*
- Facebook processes 500+ terabytes of data each day
- Square Kilometer Array (SKA) telescope
  - Southern hemisphere radio telescope with a total of $1\text{km}^2$ of data sensors
  - Will generate 1 exabyte *daily* $= 10^{18}$ bytes

- Compare this with the Apollo Guidance Computer (1966), which guided the first humans to the moon:
  - Magnetic core memory: 16-bit word length, 2048 words RAM = 4KB
  - Core rope memory: 36,864 words. 73KB



# Basic units of data

- Bits
  - Smallest unit of storage; a 0 or 1
  - With $n$ bits, can store $2^n$ patterns
- Bytes
  - 8 bits = 1 byte (why 1 byte can store 256 patterns)
  - "eight bit encoding" - used to represent characters, such as `A` represented as 65 = 01000001

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

# Basic units of data

Multi-byte units:

| unit | abbreviation | total bytes | nearest decimal equivalent |
|---|---|---|---|
| kilobyte | KB | 1,024^1 | 1000^1 |
| megabyte | MB | 1,024^2 | 1000^2 |
| gigabyte | GB | 1,024^3 | 1000^3 |
| terabyte | TB | 1,024^4 | 1000^4 |
| petabyte | PB | 1,024^5 | 1000^5 |
| exabyte | EB | 1,024^6 | 1000^6 |
| zettabyte | ZB | 1,024^7 | 1000^7 |
| yottabyte | YB | 1,024^8 | 1000^8 |

# Plan for today

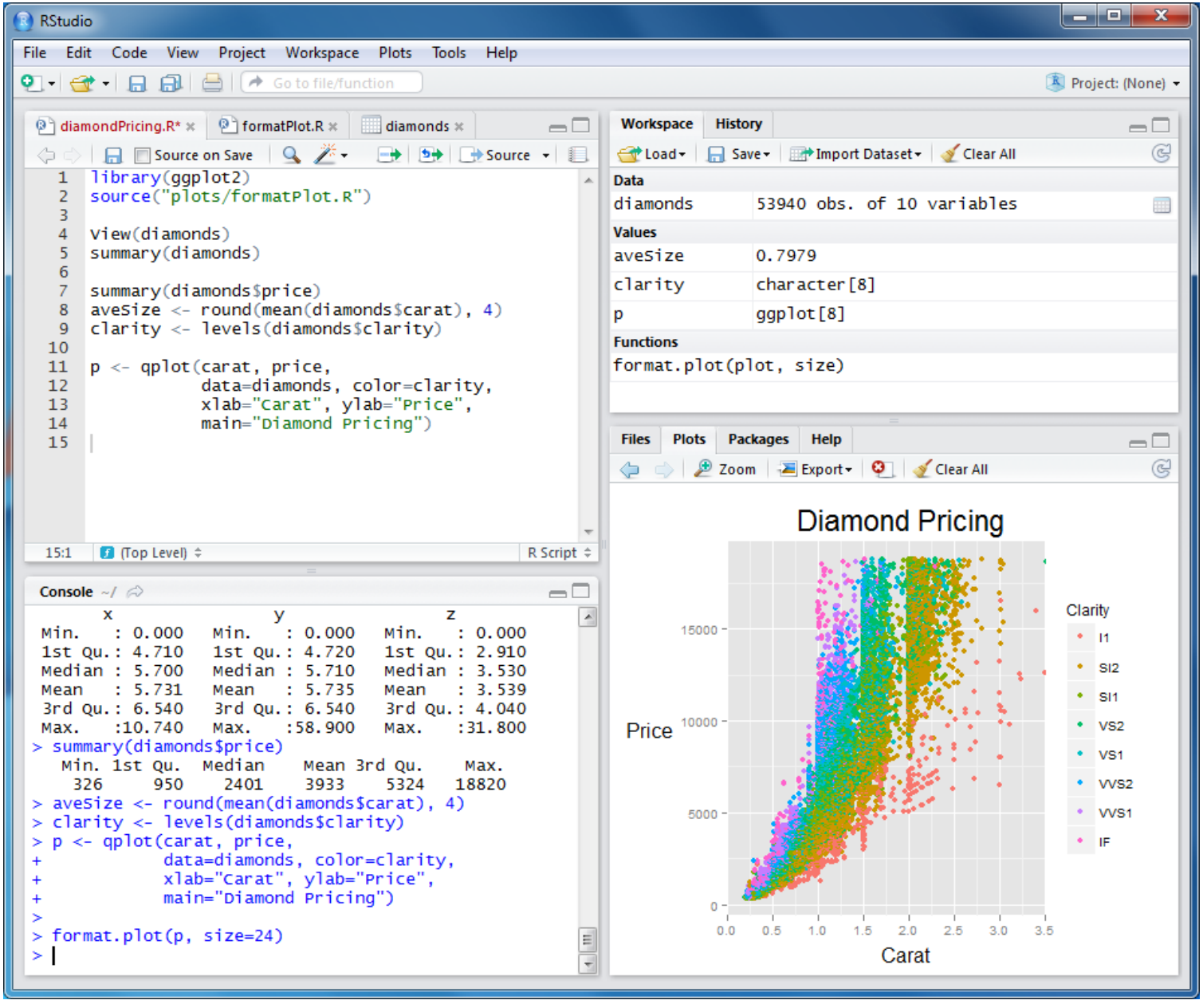
- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- Markdown in brief
- git and Github for version control

# Why we're using R

- Becoming *lingua franca* of statistical analysis in academia
- Often demanded by employers in the private sector
- It's free and open-source
- Flexible and extensible through *packages* (over 10,000 and counting!)
- Powerful tool to conduct automated text analysis, social network analysis, and data visualization, with packages such as quanteda, igraph or ggplot2
- Command-line interface and scripts favor reproducibility
- Excellent documentation and online help resources

R is also a full programming language; once you understand how to use it, you can learn other languages too.

# RStudio


RStudio
File Edit Code View Project Workspace Plots Tools Help
Go to file/function
Project: (None)
diamondPricing.R*
formatPlot.R
diamonds
Source on Save
Source
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 View(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
15:1 (Top Level) R Script
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
>
> format.plot(p, size=24)
>
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
0
5000
10000
15000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

# Plan for today

- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- R markdown in brief
- git and Github for version control

# Exemplary markdown files

`01-RMarkdown.Rmd`

`02-vectors-lists-dfs.Rmd`

# Plan for today

- Administration and logistics
- On the history of data and databases
- Data types and storage units
- Introduction to R
- Markdown in brief
- git and Github for version control

# Introduction to git/GitHub

Git is a type of version control system or VCS

- A VCS keeps records of your changes: It helps track who made changes when
- Possibility of reverting any changes and go back to previous state
- Distributed (entire code and history on each machine) – allows for collaborative development
- Git: Created by Linus Torvalds in 2005 to facilitate Linux kernel development
- Other options: Mercurial, Subversion
- GitHub allows you to host repositories and adds extra functionalities (UI, documentation, issues, user profiles...)

# Basic concepts of git

- Code lives in a repository: Collection of all files (and history)
- Every time you make changes, you need to make a commit:
  - Creates a snapshot of your code
  - Informs how files have changed
  - You need to add a message explaining changes
- After you commit, you need to push the changes to the repository on GitHub so that others can see them
- Note – you also need to pull first to receive changes from other people
- When you start from a repository someone created, you will have to first fork it (create a copy on GitHub) and then clone it (download) to your computer