

Technical Tuesdays

R Scripting

Justin van Dijk
j.t.vandijk@ucl.ac.uk

Technical Tuesdays

Introduction
15 Oct

R Scripting
29 Oct

JavaScript
19 Nov

Version Control
03 Dec

*nix Shell
22 Oct

Python
12 Nov

Databases
26 Nov

Mapping
10 Dec



Technical Tuesdays

Introduction but **not a tutorial**

Tell people what is already there and **what is possible**

Give some **examples** for inspiration

Provide a **minimum viable environment** for
further learning and exploration

Recap - Shell

Managing Input and Output

Managing Filesystem

Executing Programs

Recap - Shell

What is the shell

How to talk to the shell

File system, reading and writing

Installing and executing programs

Passing data through programs (using pipes)

Automating tasks

What is R

Scripting language for data mining and data analysis

From S to R

Packages (CRAN)

Everything is a vector

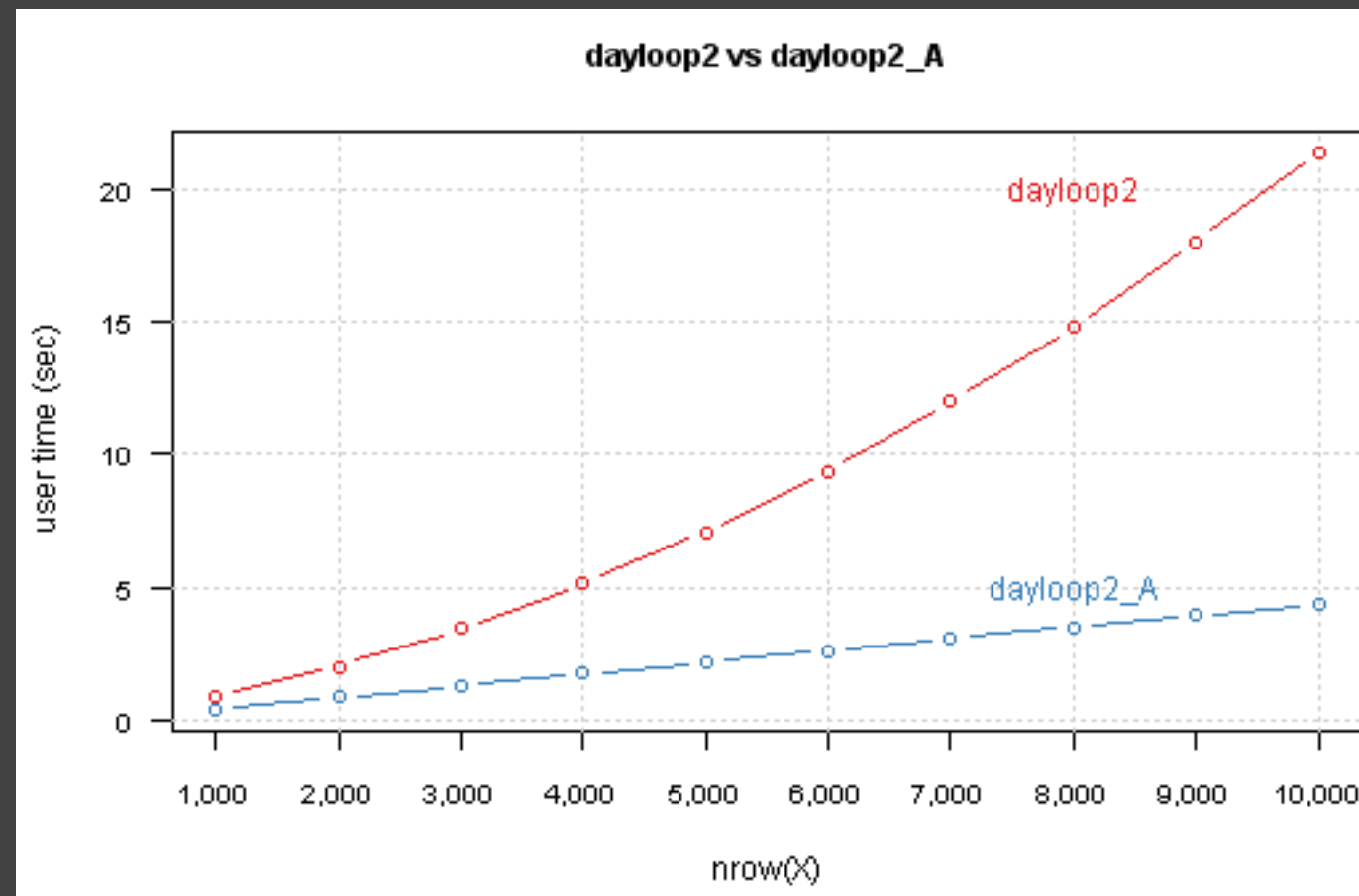
| | Homogeneous | Heterogeneous |
|-----------|---------------|---------------|
| <i>1d</i> | Atomic vector | List |
| <i>2d</i> | Matrix | Dataframe |
| <i>nd</i> | Array | |

Using R

```
dayloop2 <- function(temp){  
  for (i in 1:nrow(temp)){  
    temp[i,10] <- i  
    if (i > 1) {  
      if ((temp[i,6] == temp[i-1,6]) & (temp[i,3] == temp[i-1,3])) {  
        temp[i,10] <- temp[i,9] + temp[i-1,10]  
      } else {  
        temp[i,10] <- temp[i,9]  
      }  
    } else {  
      temp[i,10] <- temp[i,9]  
    }  
  }  
  names(temp)[names(temp) == "V10"] <- "Kumm."  
  return(temp)  
}
```

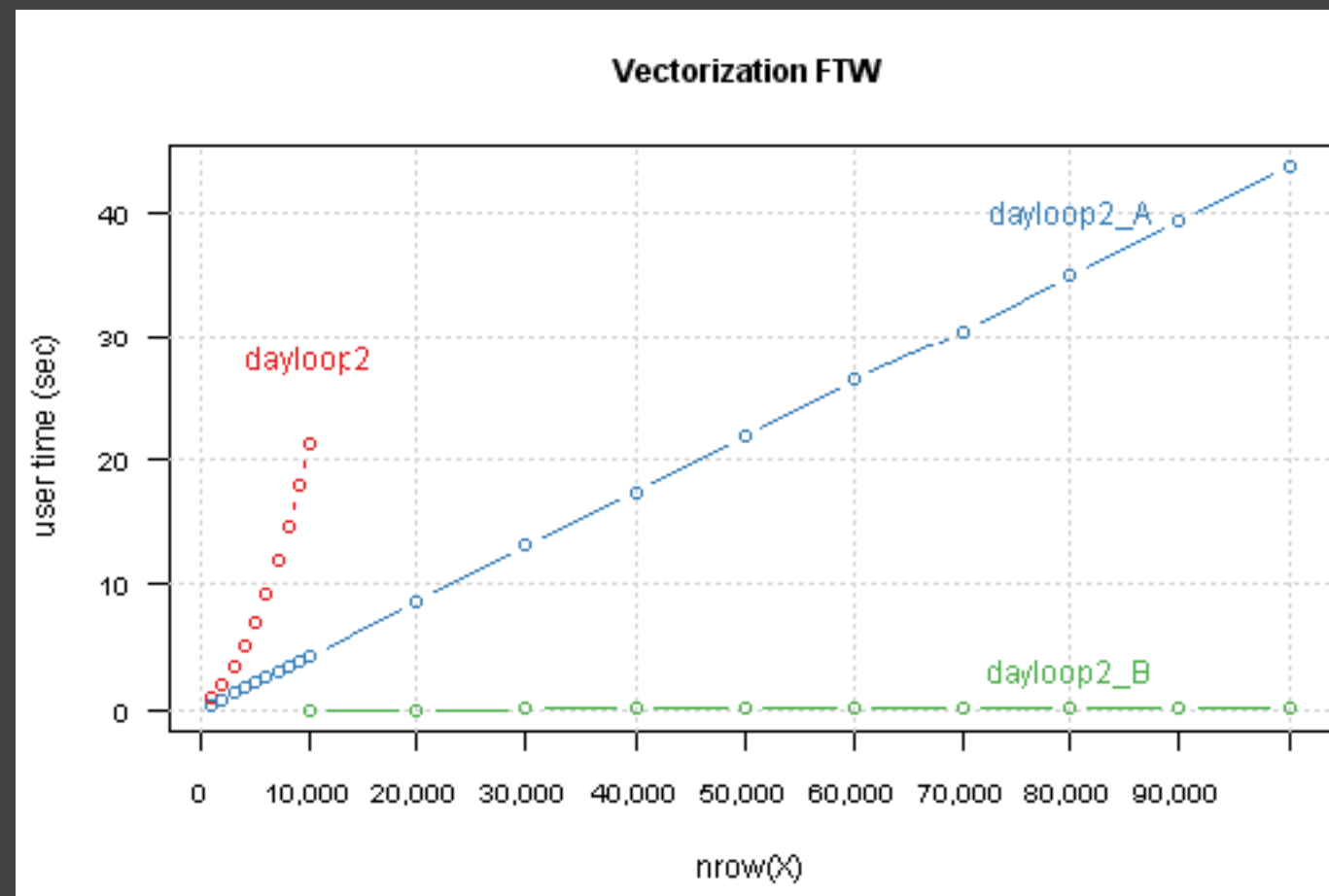
~ 850k rows

Using R



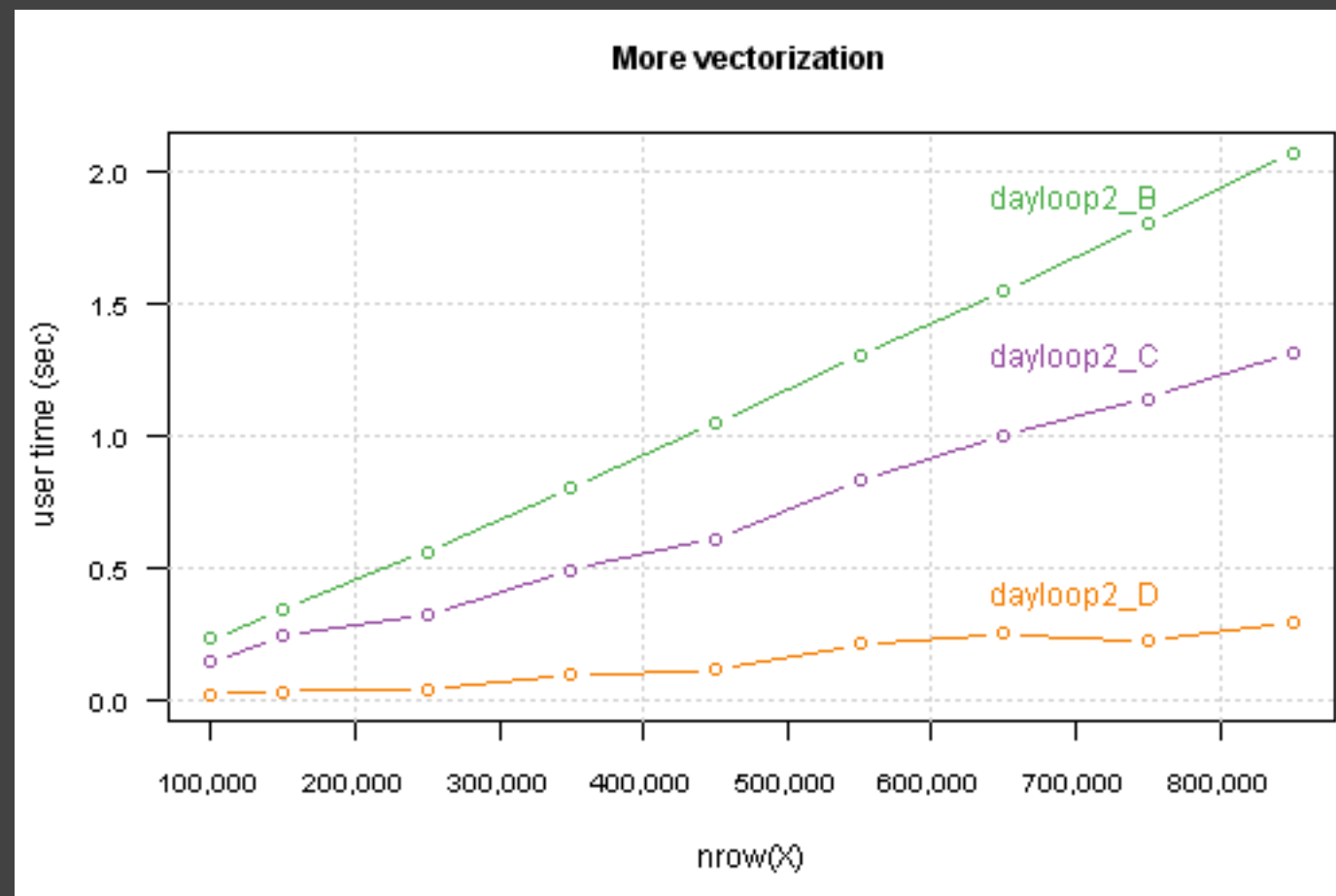
Reduce indexing

Using R



Vectorisation

Using R



More vectorisation

Advantages of R

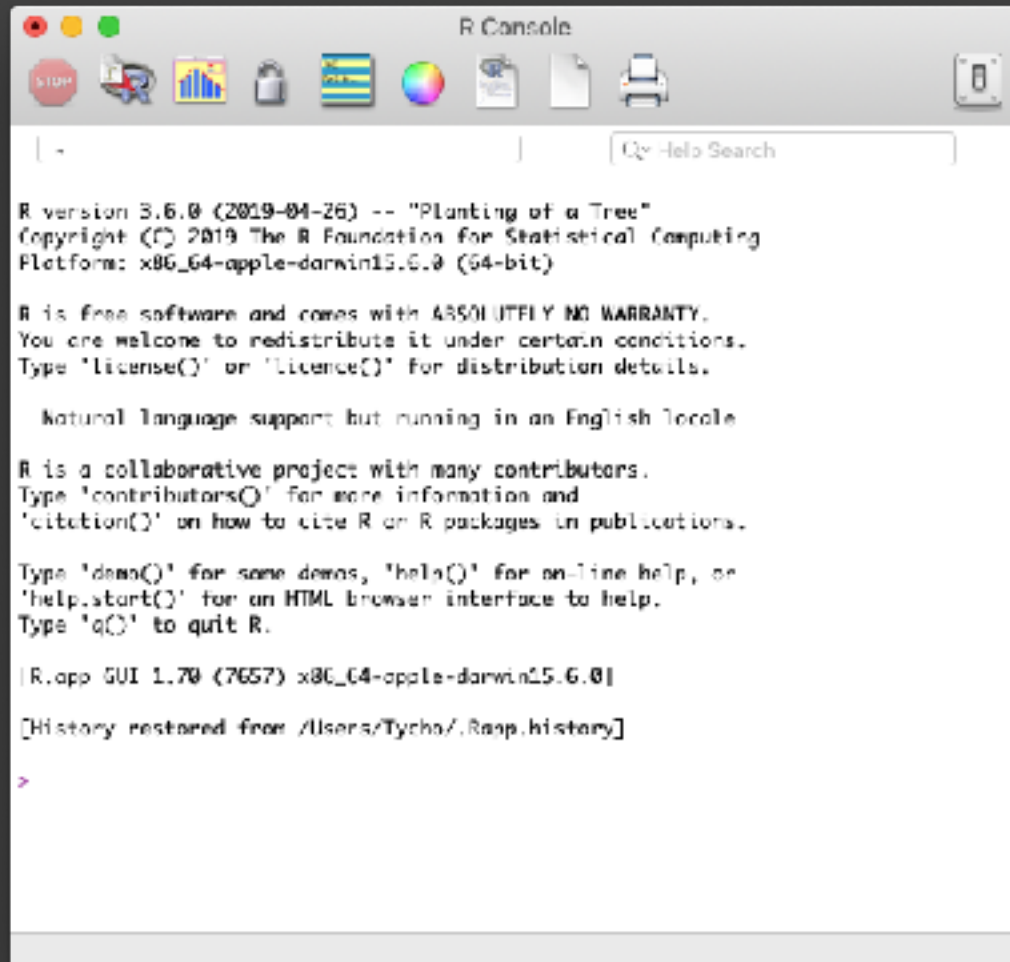
IDE Rstudio

Tidyverse and ggplot2

Large ecosystem consisting of many actively developed packages

R community and support

IDE

A screenshot of the R Console window. The title bar says "R Console". The window contains the standard R startup text, including the version (3.6.0), copyright (© 2019 The R Foundation for Statistical Computing), platform (x86_64-apple-darwin15.6.0), and a welcome message. It also shows the R.app GUI version (1.70) and the path to the history file. The prompt is a red greater-than sign (>).

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

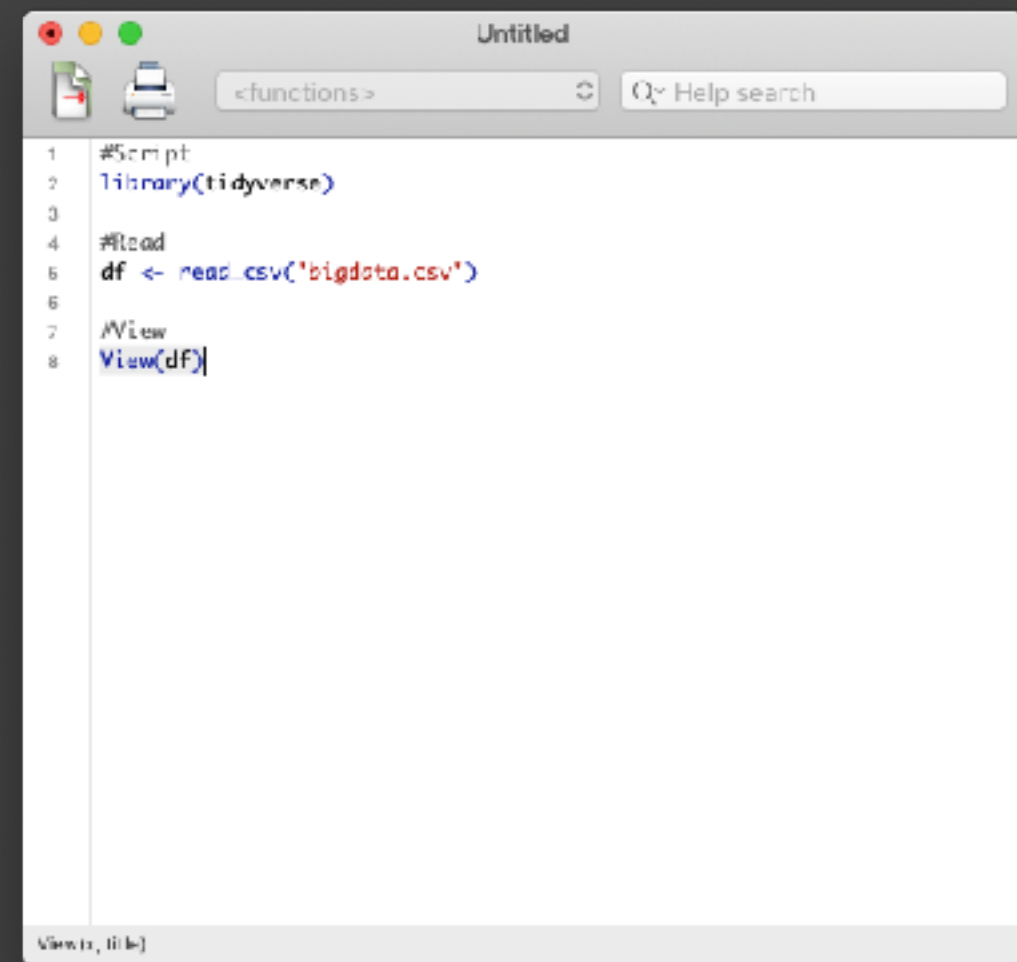
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

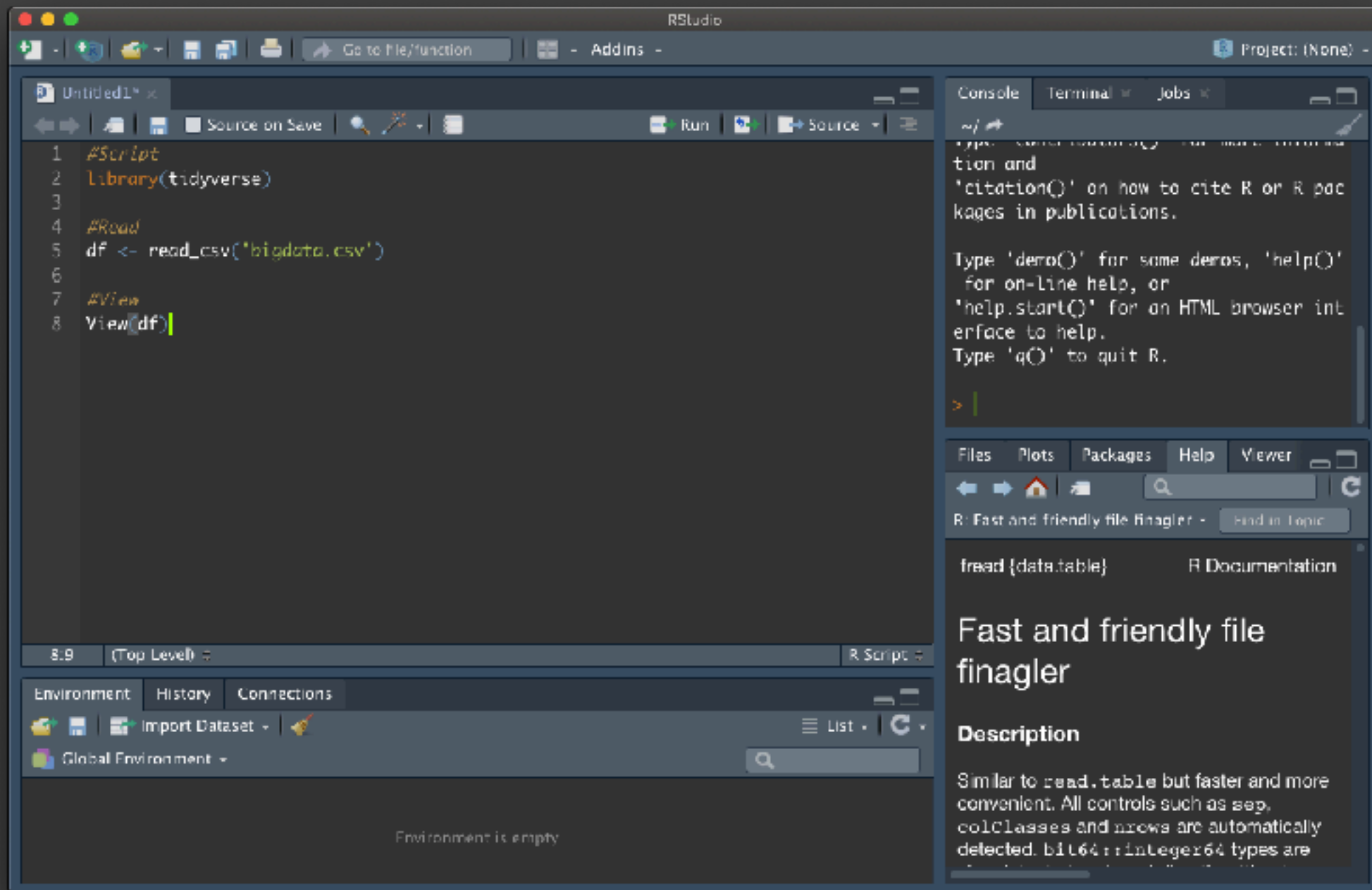
|R.app GUI 1.70 (7657) x86_64-apple-darwin15.6.0|
[History restored from /Users/Tycho/.Rapp.history]

>
```

A screenshot of an R script editor window titled "Untitled". The window has a toolbar with icons for saving, printing, and a search bar. The search bar contains the text "<functions>" and "Q Help search". The script content is as follows:

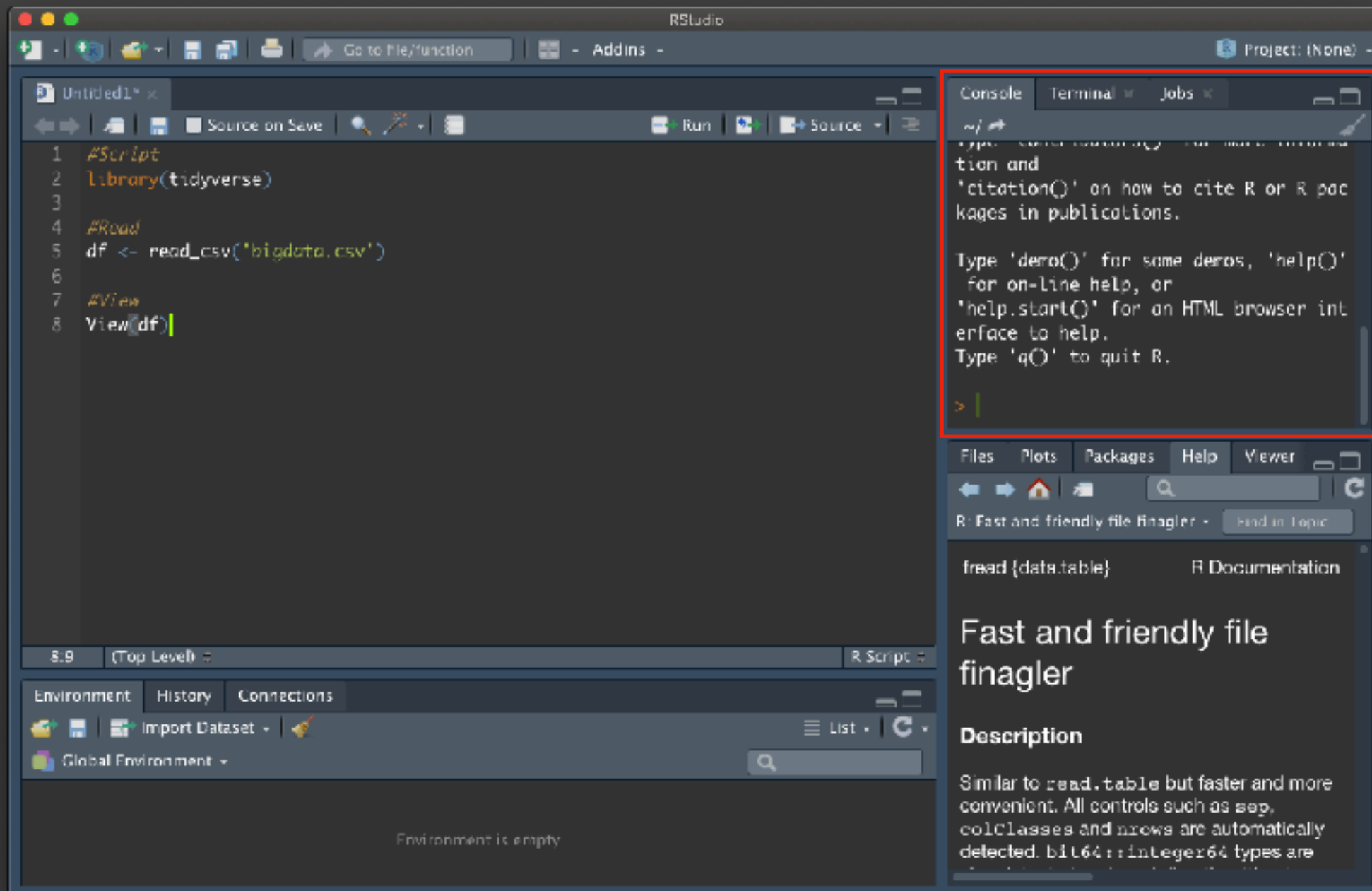
```
1 #Script
2 library(tidyverse)
3
4 #load
5 df <- read_csv('bigdata.csv')
6
7 #View
8 View(df)
```

IDE

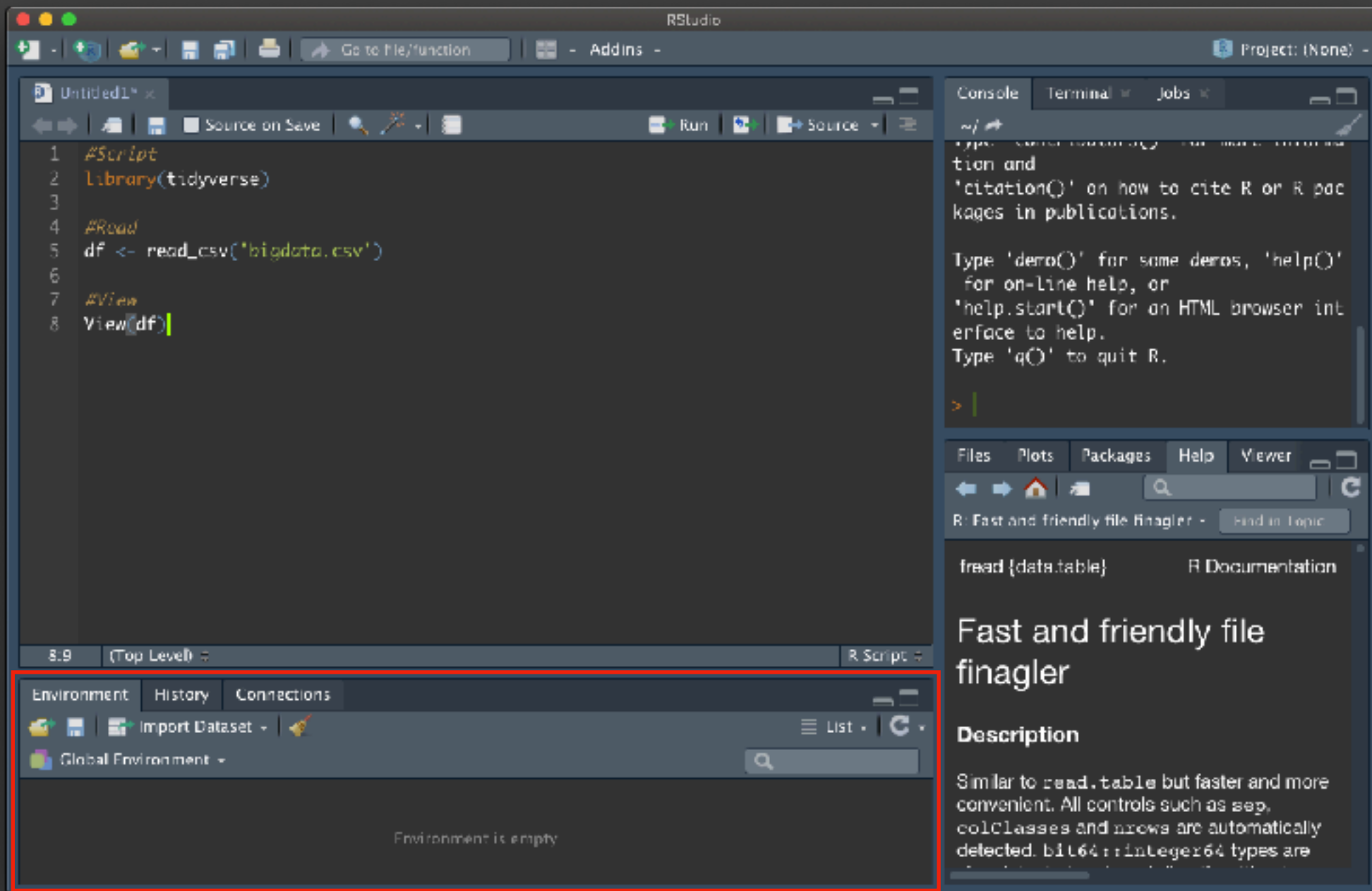


IDE

Assistant

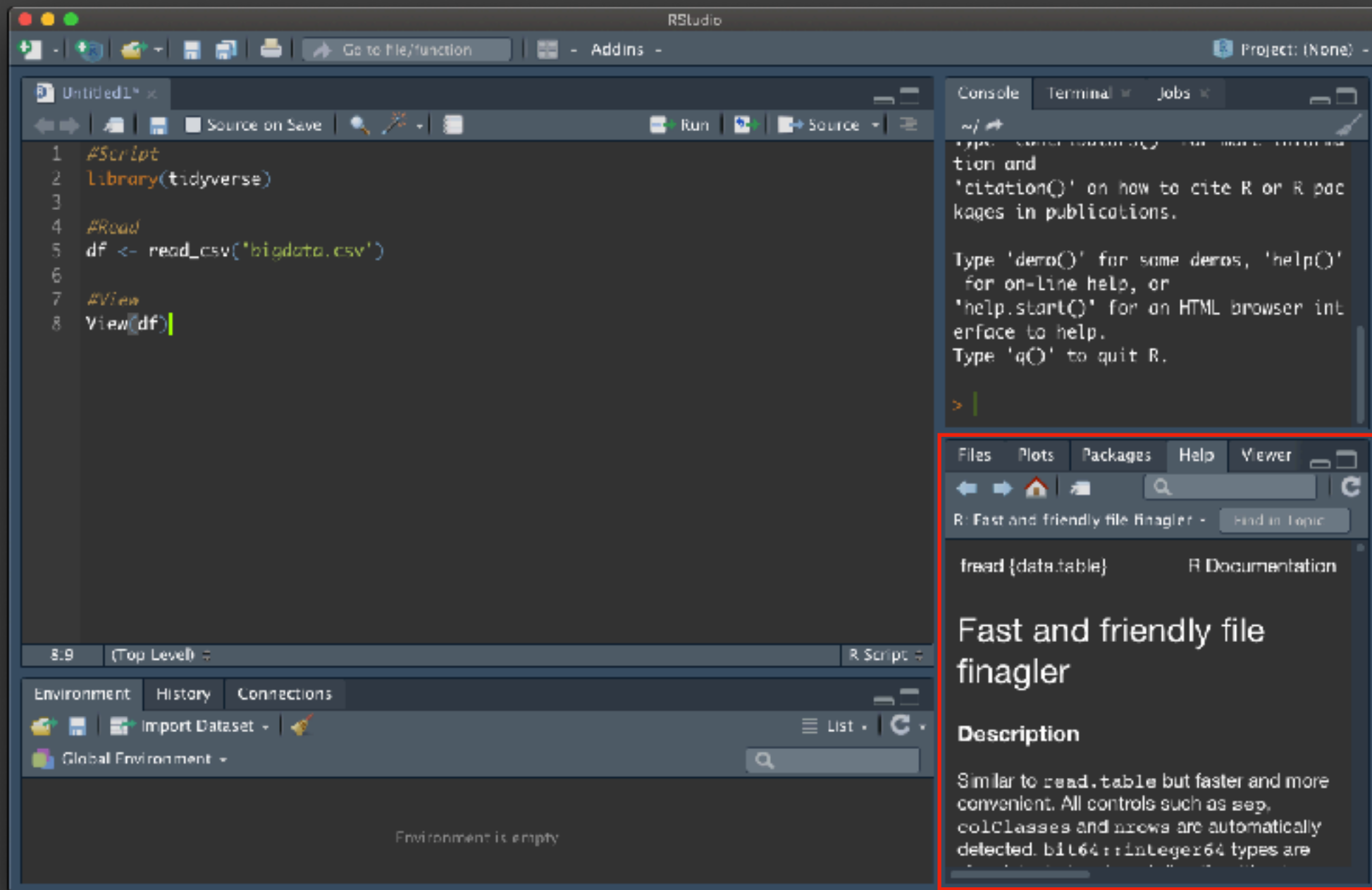


IDE



Assistant's
memory

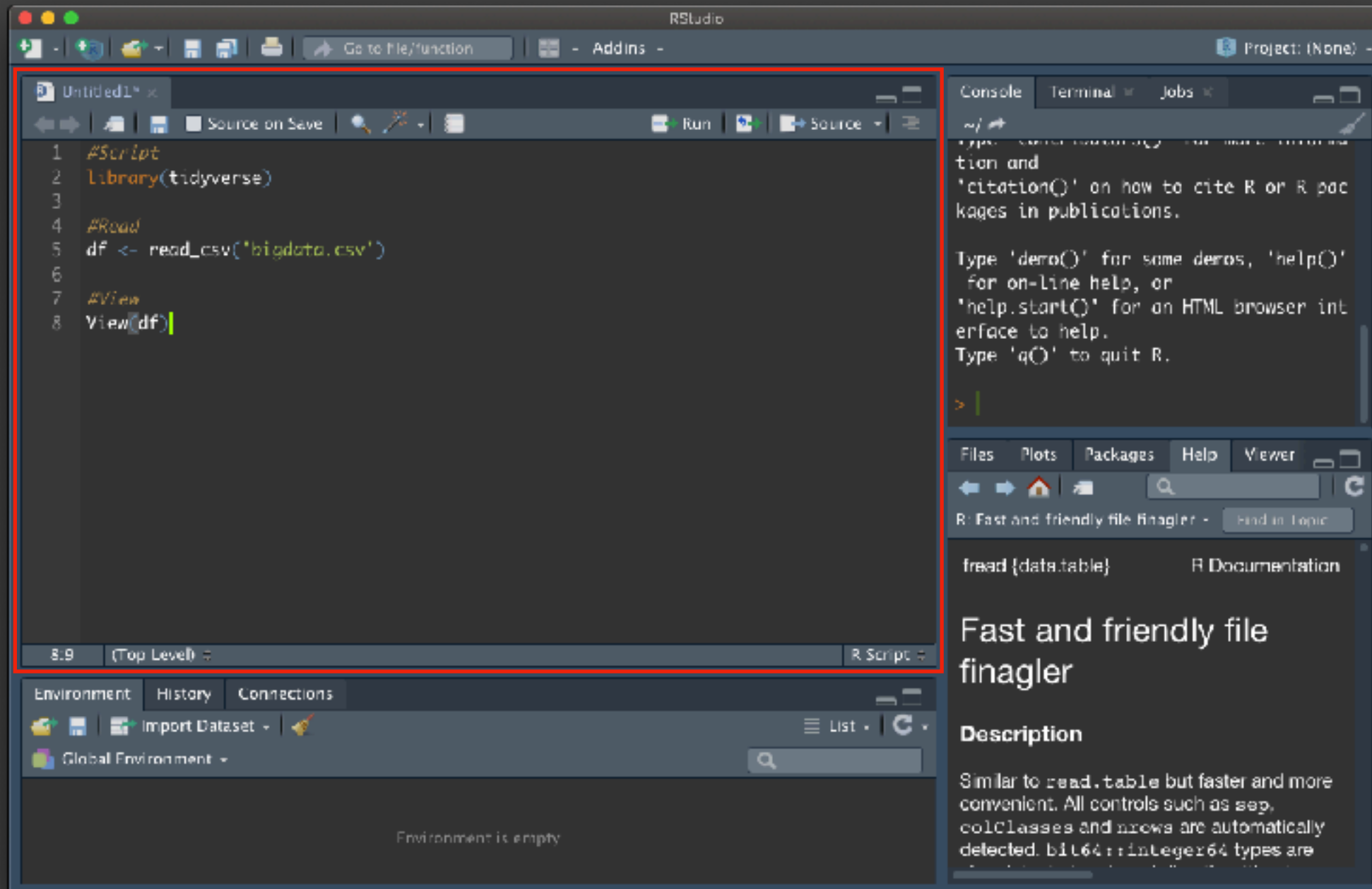
IDE



Archive

IDE

Instructions

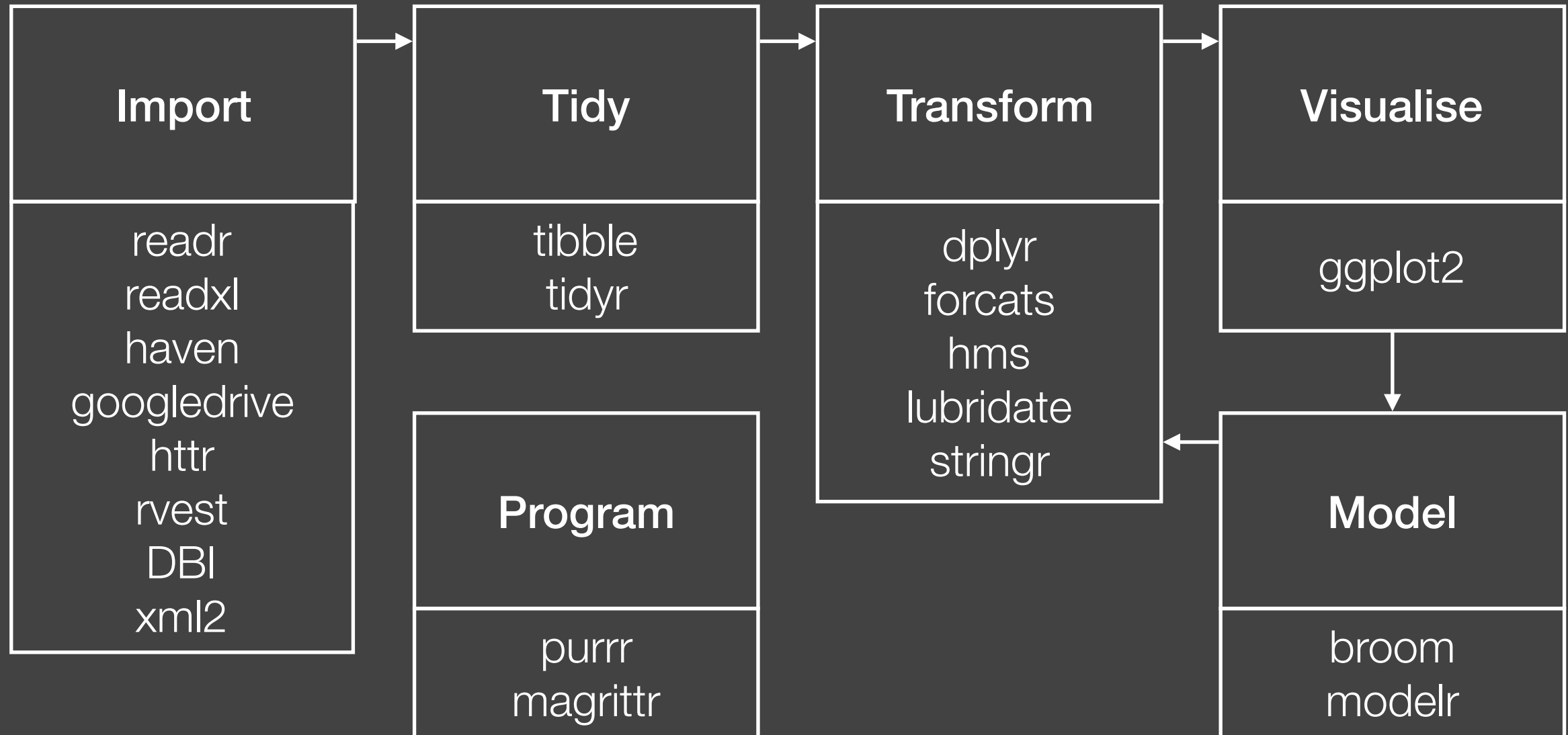


Tidyverse

Collection of R packages

Tidy data

Tidyverse



Tidy data

| country | year | cases | population |
|-------------|------|-------|------------|
| Afghanistan | 1999 | 745 | 15557071 |
| Afghanistan | 2000 | 2666 | 20595360 |
| Brazil | 1999 | 3737 | 172006362 |
| Brazil | 2000 | 8488 | 174504898 |
| China | 1999 | 21258 | 1272915272 |
| China | 2000 | 2166 | 128028583 |

Variables

Tidy data

| country | year | cases | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745 | 199870 |
| Afghanistan | 2000 | 2000 | 200000 |
| Egypt | 1999 | 57707 | 720000 |
| Egypt | 2000 | 60400 | 740040 |
| China | 1999 | 212200 | 12720102 |
| China | 2000 | 210700 | 12004200 |

Observations

Tidy data

| country | year | cases | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745 | 19987071 |
| Afghanistan | 2000 | 2666 | 20595360 |
| Brazil | 1999 | 37737 | 172000362 |
| Brazil | 2000 | 80483 | 174504898 |
| China | 1999 | 212293 | 1272913272 |
| China | 2000 | 213766 | 1280425583 |

Values

Tidy mistakes

Column headers are values, not variable names

Multiple variables are stored in one column

Variables are stored in both rows and columns

Multiple types of observational units are stored in the same table

A single observational unit is stored in multiple tables.

Tidy data

| country | year | type | count |
|-------------|------|------------|---------------|
| Afghanistan | 1999 | cases | 745 |
| Afghanistan | 1999 | population | 19,987,071 |
| Afghanistan | 2000 | cases | 2,666 |
| Afghanistan | 2000 | population | 20,595,360 |
| Brazil | 1999 | cases | 3,7737 |
| Brazil | 1999 | population | 172,006,362 |
| Brazil | 2000 | cases | 80,488 |
| Brazil | 2000 | population | 174,504,898 |
| China | 1999 | cases | 212,258 |
| China | 1999 | population | 1,272,915,272 |
| China | 2000 | cases | 213,766 |
| China | 2000 | population | 1,280,428,583 |

Tidy data

| country | year | rate |
|-------------|------|-------------------------|
| Afghanistan | 1999 | 745 / 19,987,071 |
| Afghanistan | 2000 | 2,666 / 20,595,360 |
| Brazil | 1999 | 3,7737 / 172,006,362 |
| Brazil | 2000 | 80,488 / 174,504,898 |
| China | 1999 | 212,258 / 1,272,915,272 |
| China | 2000 | 213,766 / 1,280,428,583 |

Tidy data

| country | 1999 | 2000 |
|-------------|---------|---------|
| Afghanistan | 745 | 2,666 |
| Brazil | 3,7737 | 80,488 |
| China | 212,258 | 213,766 |

Cases

| country | 1999 | 2000 |
|-------------|---------------|---------------|
| Afghanistan | 19,987,071 | 20,595,360 |
| Brazil | 172,006,362 | 174,504,898 |
| China | 1,272,915,272 | 1,280,428,583 |

Population

Tidy data

| country | year | cases | population |
|-------------|------|---------|---------------|
| Afghanistan | 1999 | 745 | 19,987,071 |
| Afghanistan | 2000 | 2,666 | 20,595,360 |
| Brazil | 1999 | 3,7737 | 172,006,362 |
| Brazil | 2000 | 80,488 | 174,504,898 |
| China | 1999 | 212,258 | 1,272,915,272 |
| China | 2000 | 213,766 | 1,280,428,583 |

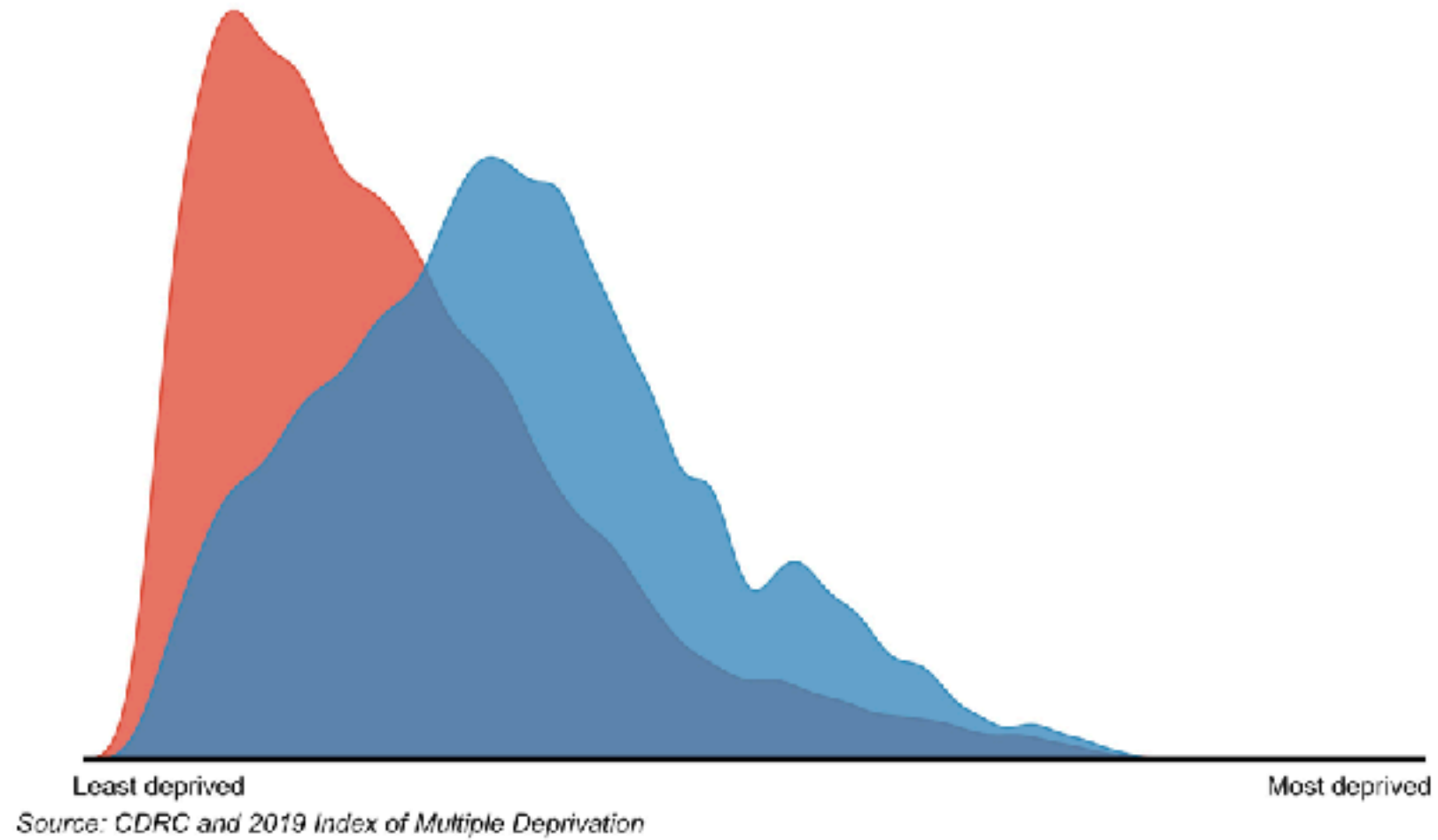
ggplot2

ggplot2 is a system for declaratively creating graphics,
based on "The Grammar of Graphics"

ggplot2

Index of Multiple Deprivation

Multiple deprivation experienced by British Chinese (red) and British Bangladeshi (blue) residents



ggplot2

Outward Migration

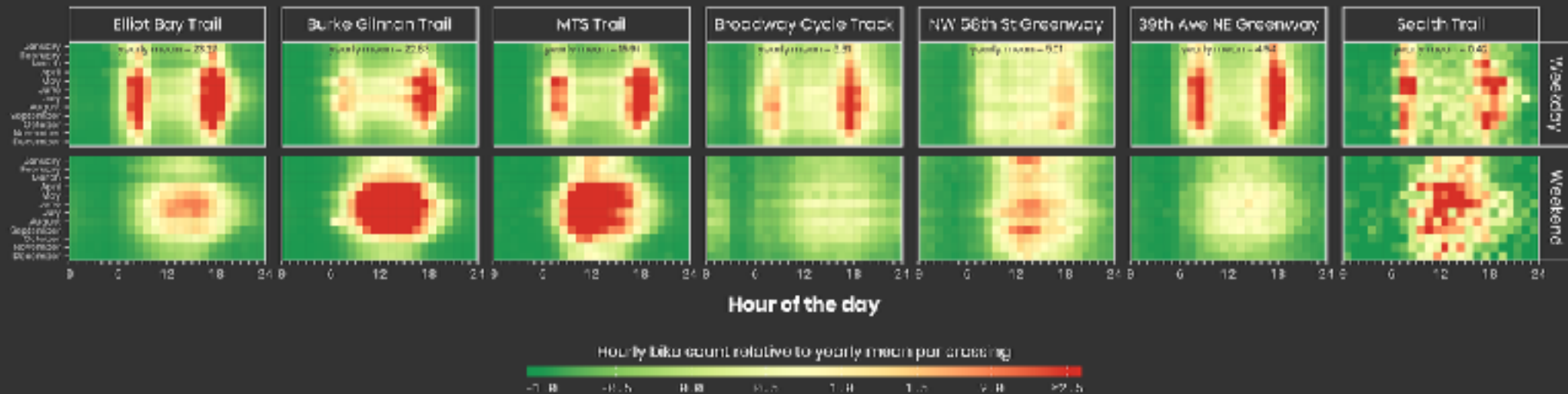
Detail of regional, local authority and ward destinations of movers leaving Spitalfields and Banglatown, 1997-2016



ggplot2

Riding the green wave in Seattle

Monthly bike traffic (2014-2018), based on data from seattle.gov

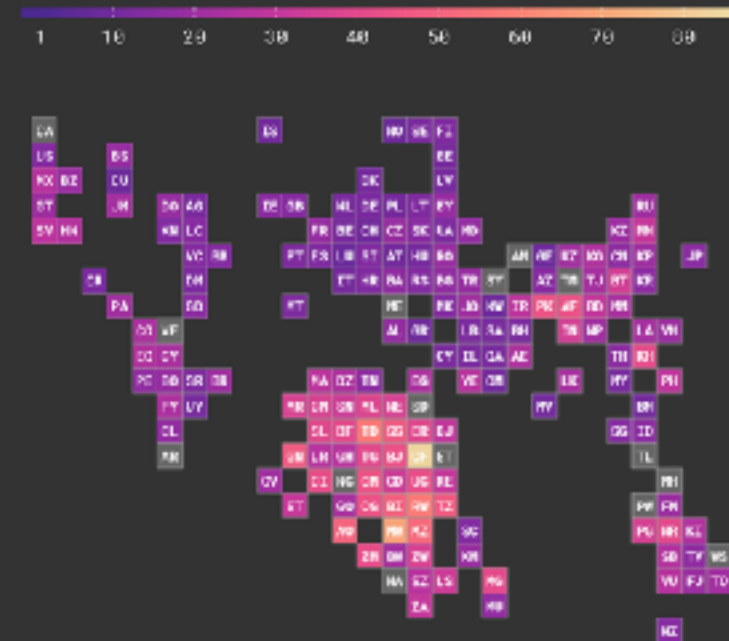
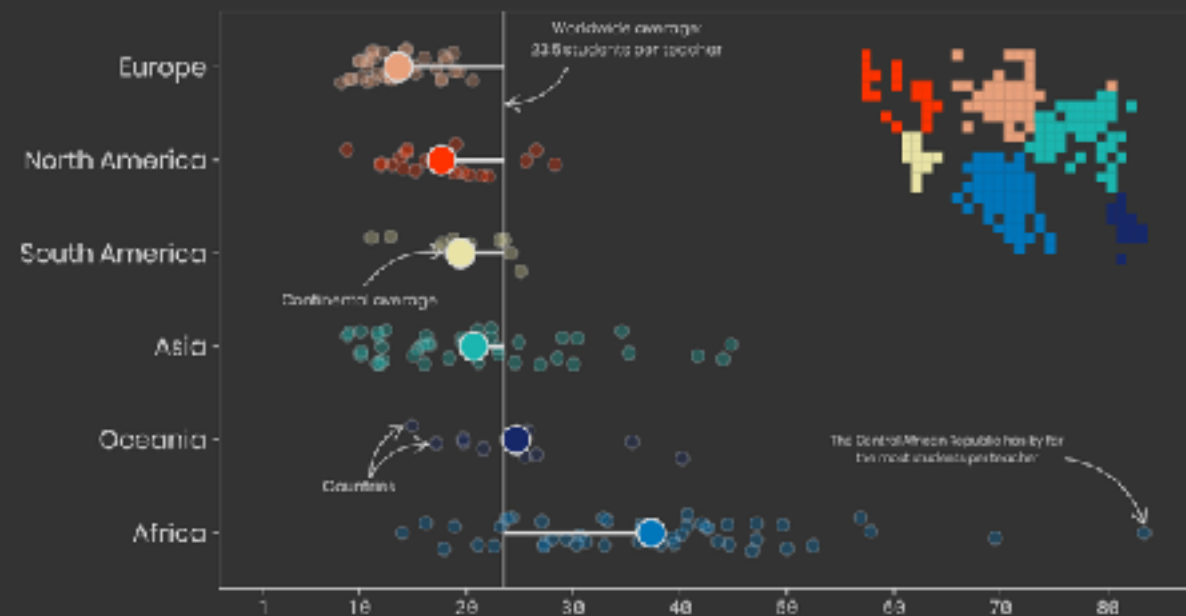


Visualization by Cédric Scherer

ggplot2

Global student to teacher ratios in primary education

Latest reported student to teacher ratio per country and continent (2012-2018)



Visualization by Cédric Scherer | Data: "eAtlas of Teachers" by UNESCO

ggplot2

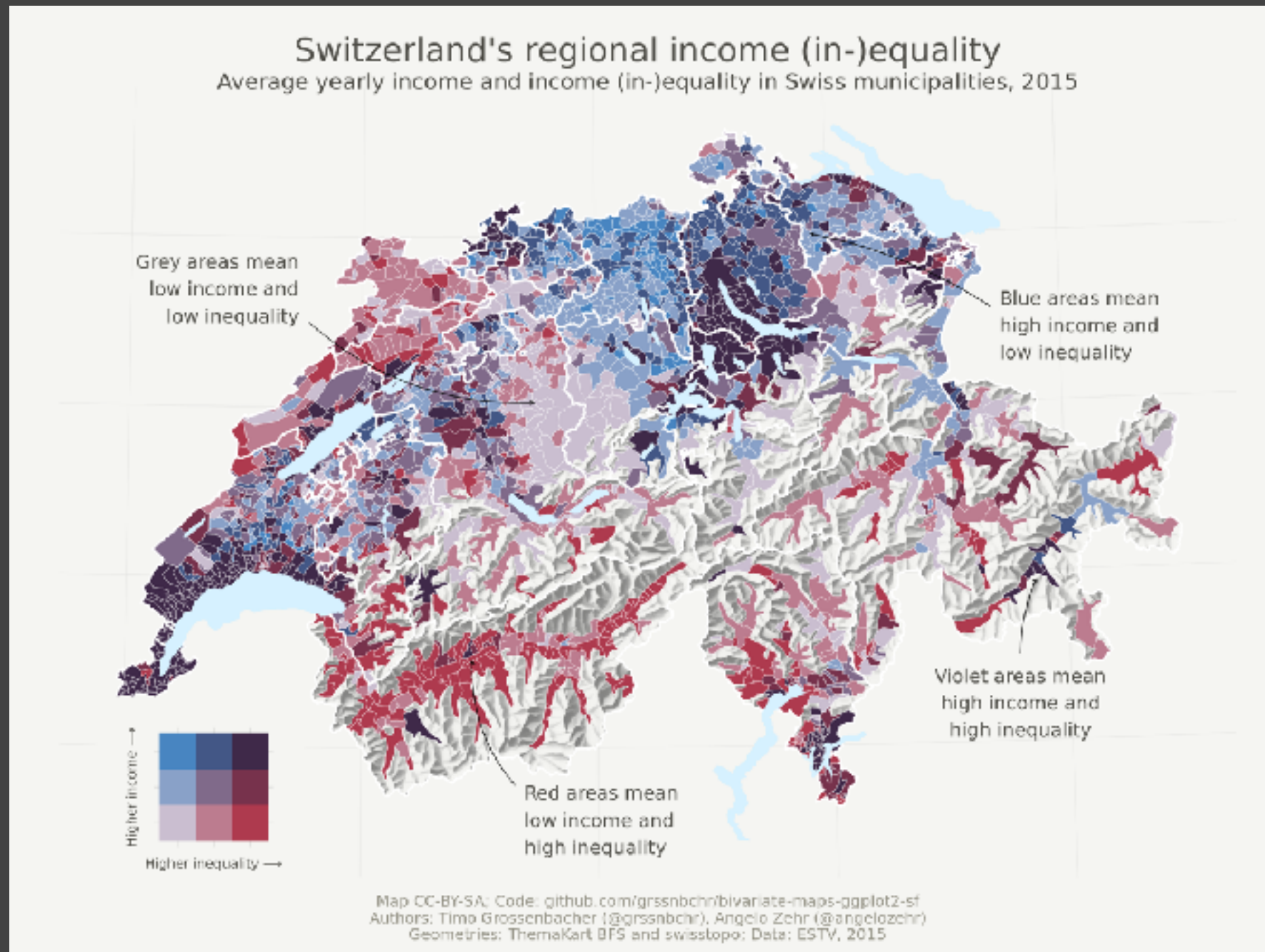
Plastic pollution – absolute and relative plastic waste generation across the world

Middle- and low income countries tend to generate high amounts of mismanaged waste which is at high risk of entering the ocean.



Visualization by Cédric Scherer | Data: Ritchie & Roser (2018), OurWorldInData.org

ggplot2



ggplot2

data {raw data}

+

layers {shapes and summarised data}

+

aesthetics {making of the objects}

+

scales

+

coordinate system

+

facets

+

visual theme

Community

LondonR

<https://www.meetup.com/LondonR/>

R-Ladies

<https://rladies.org>

Resources

R for Data Science

<https://r4ds.had.co.nz>

Advanced R

<https://adv-r.hadley.nz>

ggplot2: Elegant Graphics for Data Analysis

<https://ggplot2-book.org/>

Hands-On Programming with R

<https://rstudio-education.github.io/hopr/>

Useful packages

Loading data

| What for | Packages |
|-----------------------------|---------------------------------|
| <i>Relational databases</i> | RMySQL, RPostgresSQL, RSQLite |
| <i>Various file formats</i> | XLConnect, xlsx, foreign, haven |

Manipulating data

| What for | Packages |
|-----------------------|-------------------|
| <i>Data wrangling</i> | dplyr, tidyr |
| <i>Text</i> | stringr, tidytext |
| <i>Dates</i> | lubridate |

Modelling data

| What for | Packages |
|---------------------------|---|
| <i>Statistical models</i> | caret, randomForest, glmnet, nnet, mice |

Dealing with spatial data

| What for | Packages |
|----------------|---------------------------|
| <i>Spatial</i> | sp, maptools, ggmap, tmap |

Data visualisation

| What for | Packages |
|-----------------|------------------------|
| <i>Graphics</i> | ggplot, ggraph, ggtext |

Data output

| What for | Packages |
|----------------|-------------------|
| <i>Results</i> | R Markdown, shiny |

General performance

| What for | Packages |
|-----------------------|----------------------|
| <i>Large datasets</i> | parallel, data.table |

Demo

Parallel

Pipeline: R

```
# -----  
# Loading the libraries  
# -----  
library(tidyverse)  
library(RJSONIO)  
day_folder <- "location_of_the_data"  
sensors <- paste(day_folder, dir(day_folder), sep = "/")[1:25]  
  
# -----  
# Read and Parse JSON files  
# -----  
for(sensor in sensors) {  
  files <- paste(sensor, dir(sensor), sep = "/")  
  for( file in files ) {  
    records <- fromJSON(file);  
    location <- vector();  
    timestamp <- vector();  
    macaddress <- vector();  
    for(record in records) {  
      location <- append(location, get_location(file))  
      timestamp <- append(timestamp, get_time(file))  
      fullmac <- paste0(record$MacAddress, record$VendorMacPart)  
      macaddress <- append(macaddress, fullmac); }  
    df <- data.frame(location, timestamp, mac)  
    probes <- rbind(probes, df) } }  
  
# -----  
# Aggregate the counts for each interval  
# -----  
probes %>%  
  group_by(location, time) %>%  
  summarise(count = length(unique(paste0(vendor, mac)))) %>%  
  write.csv("output.csv", row.names=FALSE)
```

20_m14_s

Pipeline: Unix tools

```
# -----  
# Get a list of locations  
# -----  
awkc="awk -vFPAT='^[^,]*|\\\"[^\"]*\\\"' -v OFS=','"  
folder="location_of_the_data"  
sensors=`ls $FOLDER | head -n 25`  
  
# -----  
# Loop through locations and parse and output results  
# -----  
for sensor in $sensors;  
do  
  jq_string=".[] | \  
    [\"$sensor\",\  
      .timestamp_from_filename,\  
      .VendorMacPart+.MacAddress] \  
    | @csv";  
  cmd="jq -r '$jq_string' $folder$sensor/*.pd \  
    | sort | uniq \  
    | $awkc '{print \$1, \$2}' \  
    | sort | uniq -c";  
  echo "$(eval $cmd)" > output.csv;  
done
```

18_s

Pipeline: GNU Parallel

```
# -----  
# Get a list of locations  
# -----  
awkc="awk -vFPAT='^[^,]*|\\\"[^\"]*\\\"' -v OFS=','"  
folder="location_of_the_data"  
sensors=`ls $folder | head -n 25`  
  
# -----  
# Set up the processing pipeline  
# -----  
jq_string=".[] | \  
  [\"{}\",\  
    .timestamp_from_filename,\  
    .VendorMacPart+.MacAddress] \  
  | @csv";  
cmd="jq -r '$jq_string' $folder{}/*.pd \  
  | sort | uniq \  
  | $awkc '{print \$1, \$2}' \  
  | sort | uniq -c";  
  
# -----  
# Apply the pipeline in parallel over each location  
# -----  
echo "$sensors" \  
  | parallel "$cmd" \  
  > output.csv
```

3_s

Questions