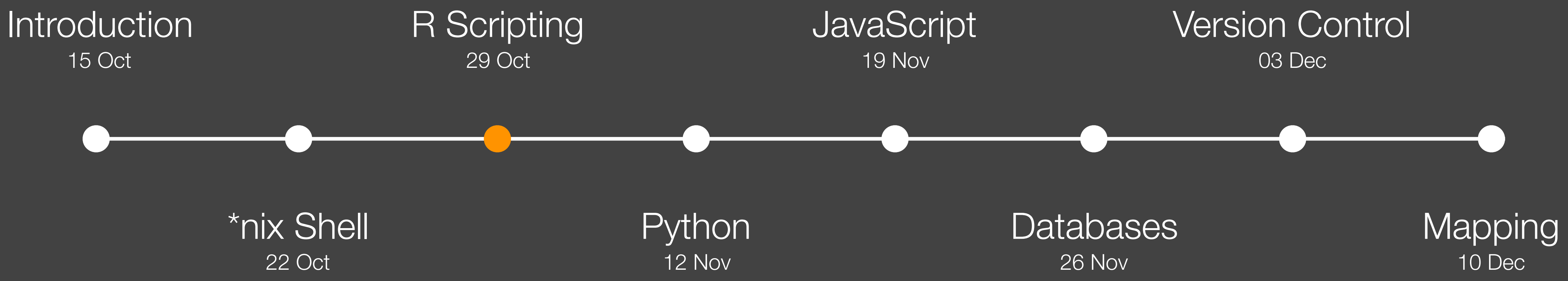
Technical Tuesdays

R Scripting

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

Technical Tuesdays



Introduction
15 Oct

*nix Shell
22 Oct

R Scripting
29 Oct

Python
12 Nov

JavaScript
19 Nov

Databases
26 Nov

Version Control
03 Dec

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

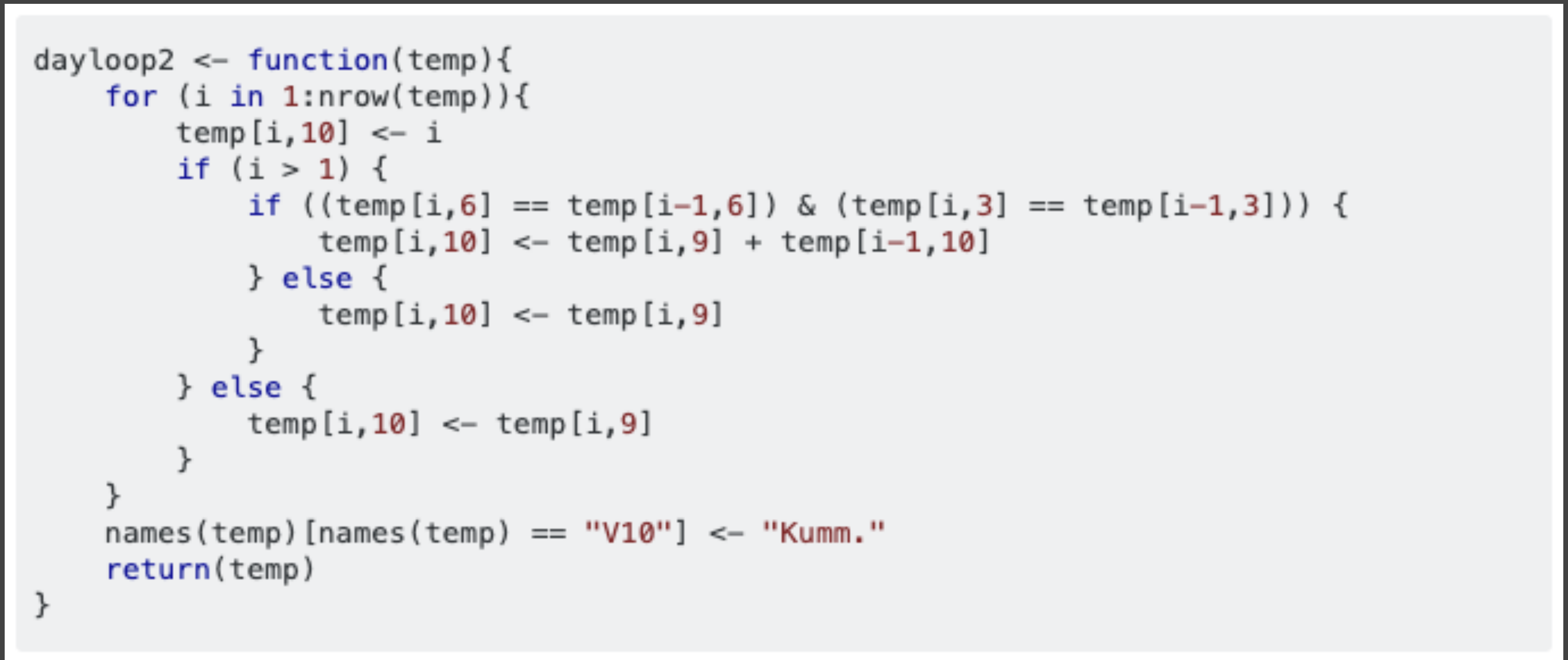
Packages (CRAN)

R ranks 15th in the TIOBE index

Everything is a vector

	Homogeneous	**Heterogeneous**
1d	Atomic vector	List
2d	Matrix	Dataframe
nd	Array	

Using R

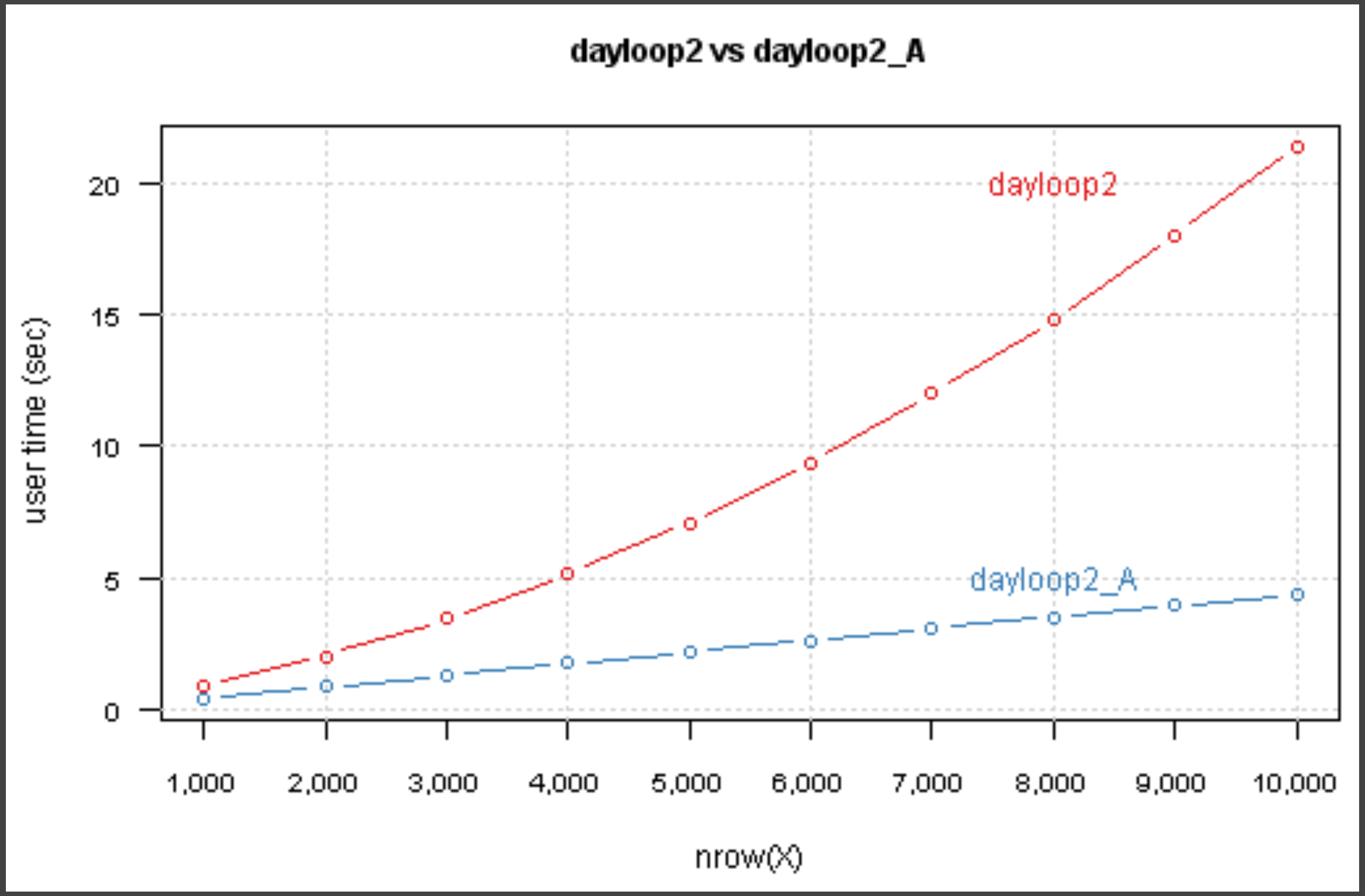


```r
dayloop2 <- function(temp){
    for (i in 1:nrow(temp)){
        temp[i,10] <- i
        if (i > 1) {
            if ((temp[i,6] == temp[i-1,6]) & (temp[i,3] == temp[i-1,3])) {
                temp[i,10] <- temp[i,9] + temp[i-1,10]
            } else {
                temp[i,10] <- temp[i,9]
  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

~ 850k rows

https://stackoverflow.com/questions/2908822/speed-up-the-loop-operation-in-r

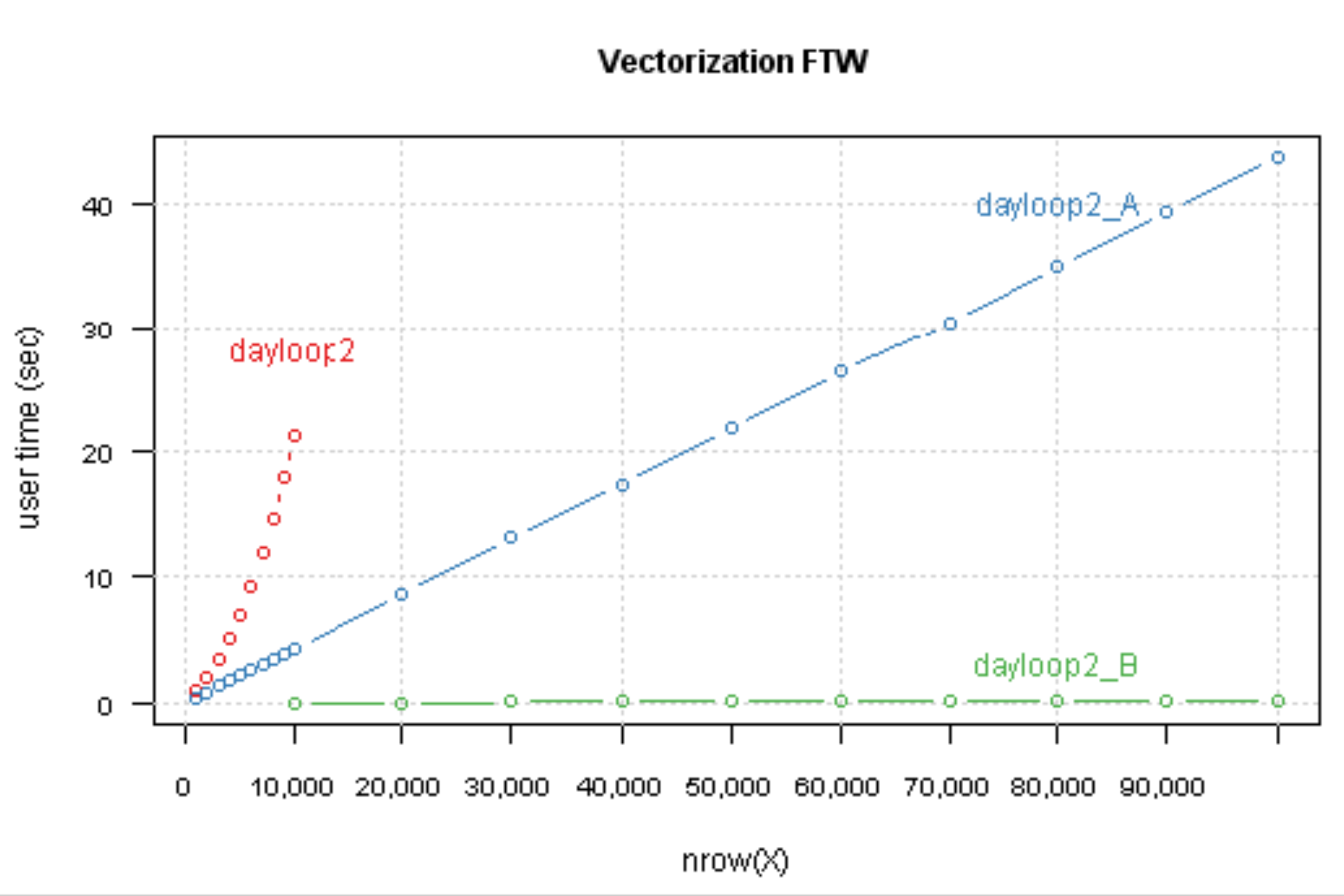
Using R



Reduce indexing

https://stackoverflow.com/questions/2908822/speed-up-the-loop-operation-in-r

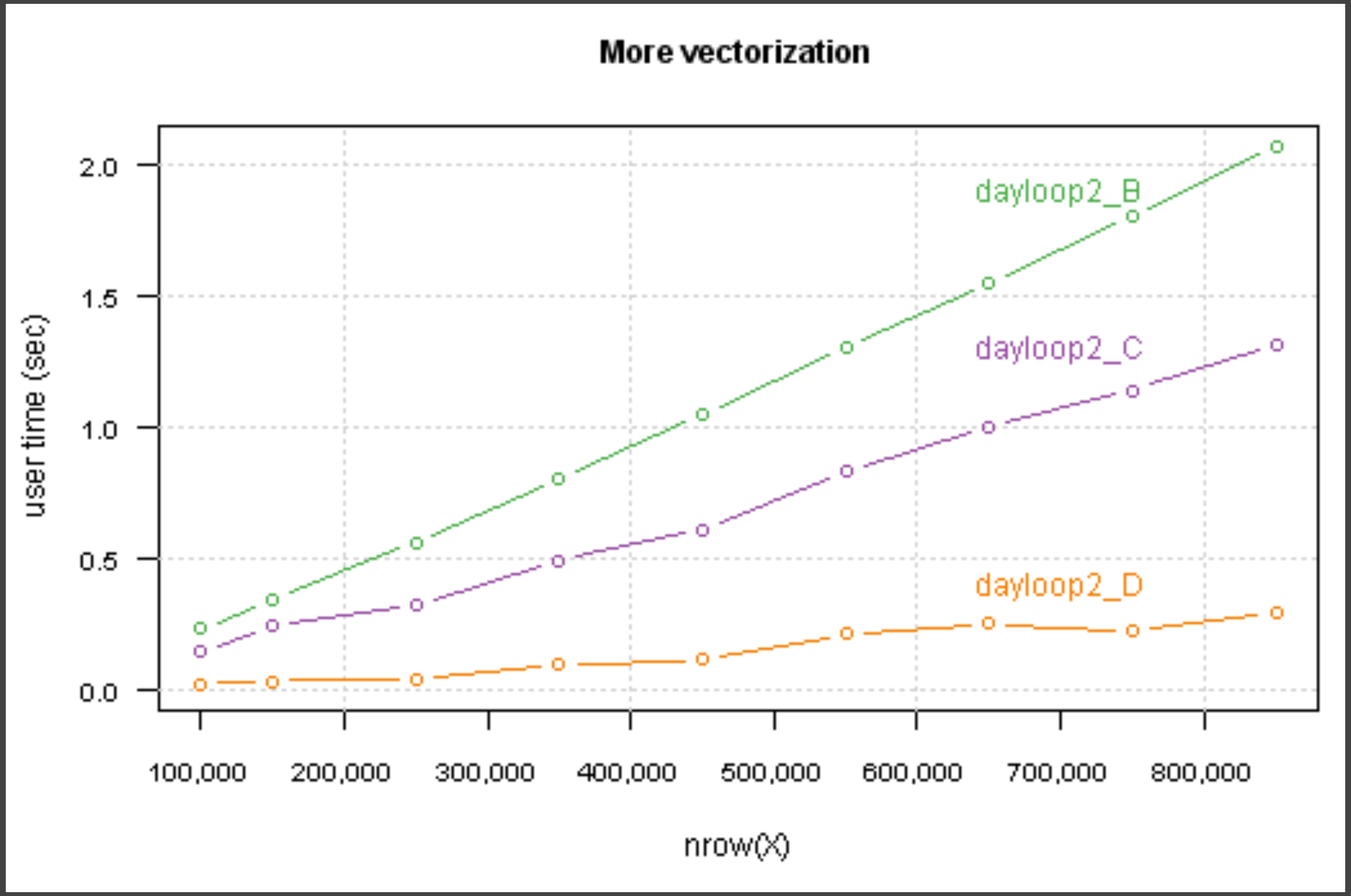
Using R



Vectorisation

https://stackoverflow.com/questions/2908822/speed-up-the-loop-operation-in-r

Using R



More vectorisation

https://stackoverflow.com/questions/2908822/speed-up-the-loop-operation-in-r

Advantages of R

IDE Rstudio

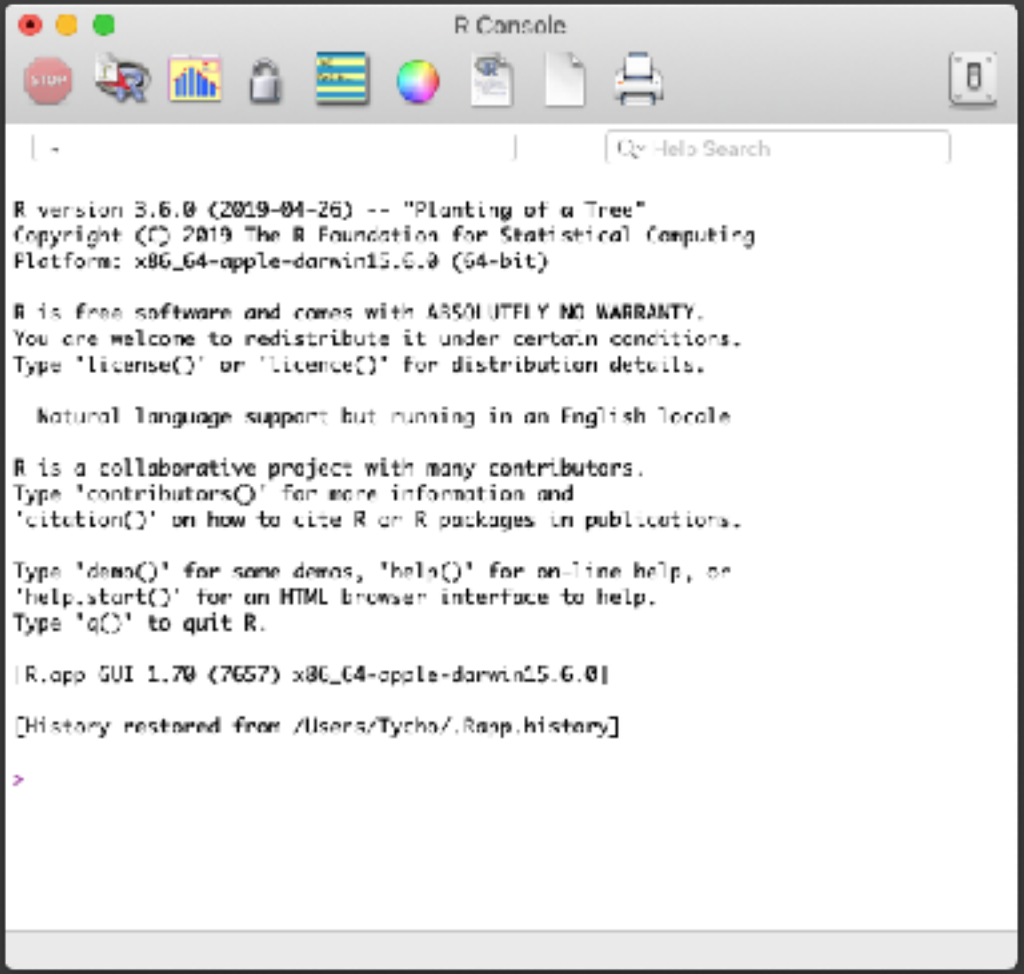
Tidyverse and ggplot2

Large ecosystem consisting of many actively developed packages (~15,000)

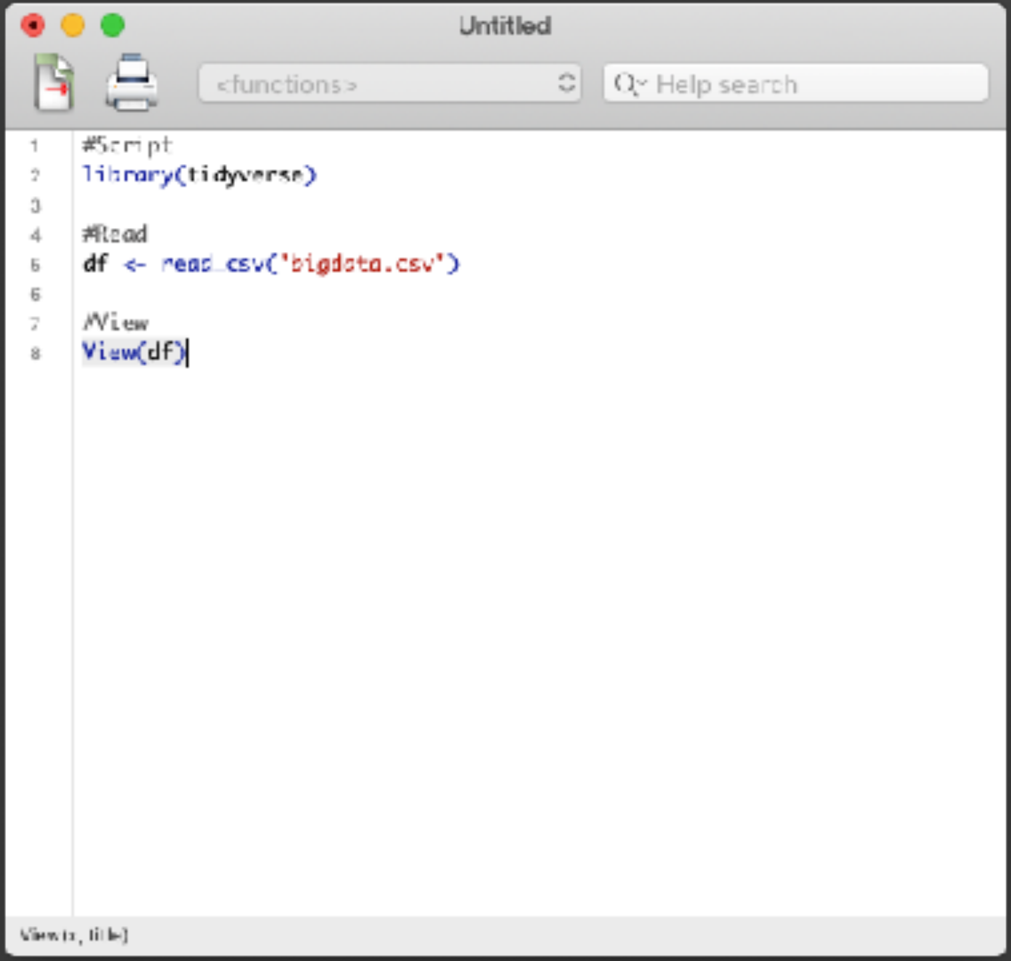
Can be linked to C, C++

R community and support

IDE



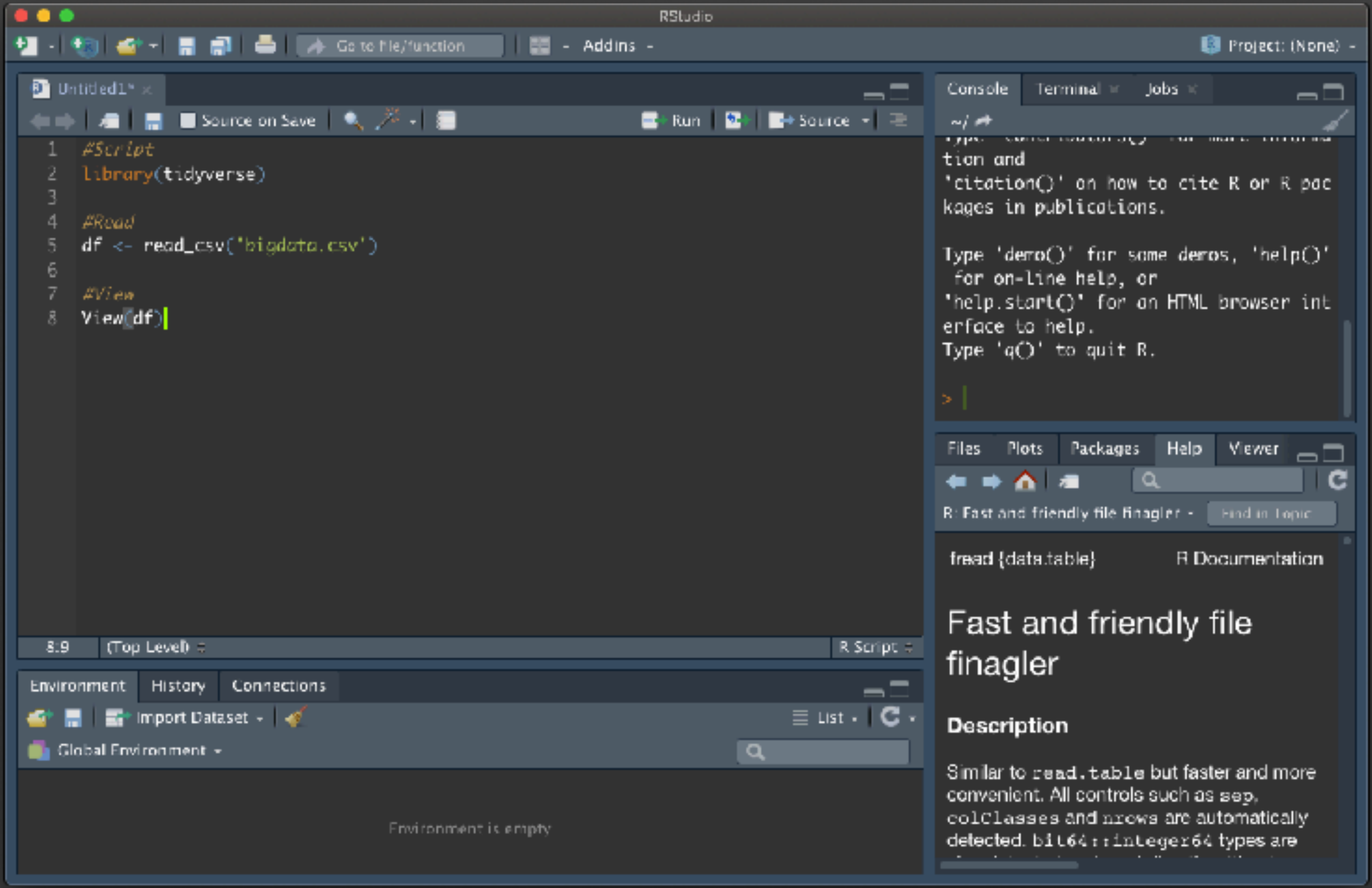
```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

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]

>
```



```
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

View(x, title)

IDE



```
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

```
tion and
'citation()' on how to cite R or R pac
kages in publications.

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

>
```

fread {data.table}   R Documentation

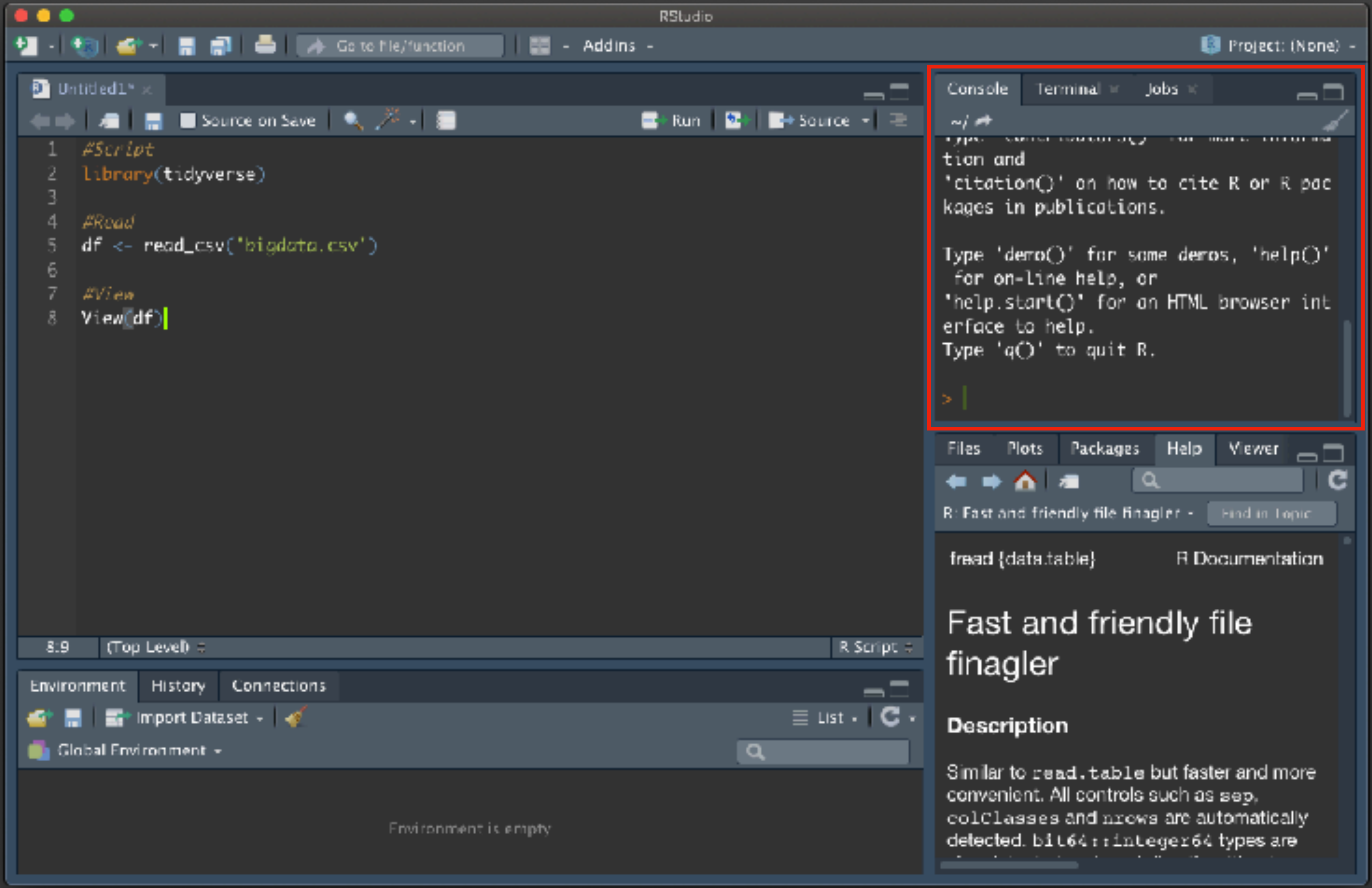
Fast and friendly file finagler

Description

Similar to `read.table` but faster and more convenient. All controls such as `sep`, `colClasses` and `nrows` are automatically detected. `bit64::integer64` types are

IDE

Assistant



```
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

```
tion and
'citation()' on how to cite R or R pac
kages in publications.

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

>
```

Environment is empty

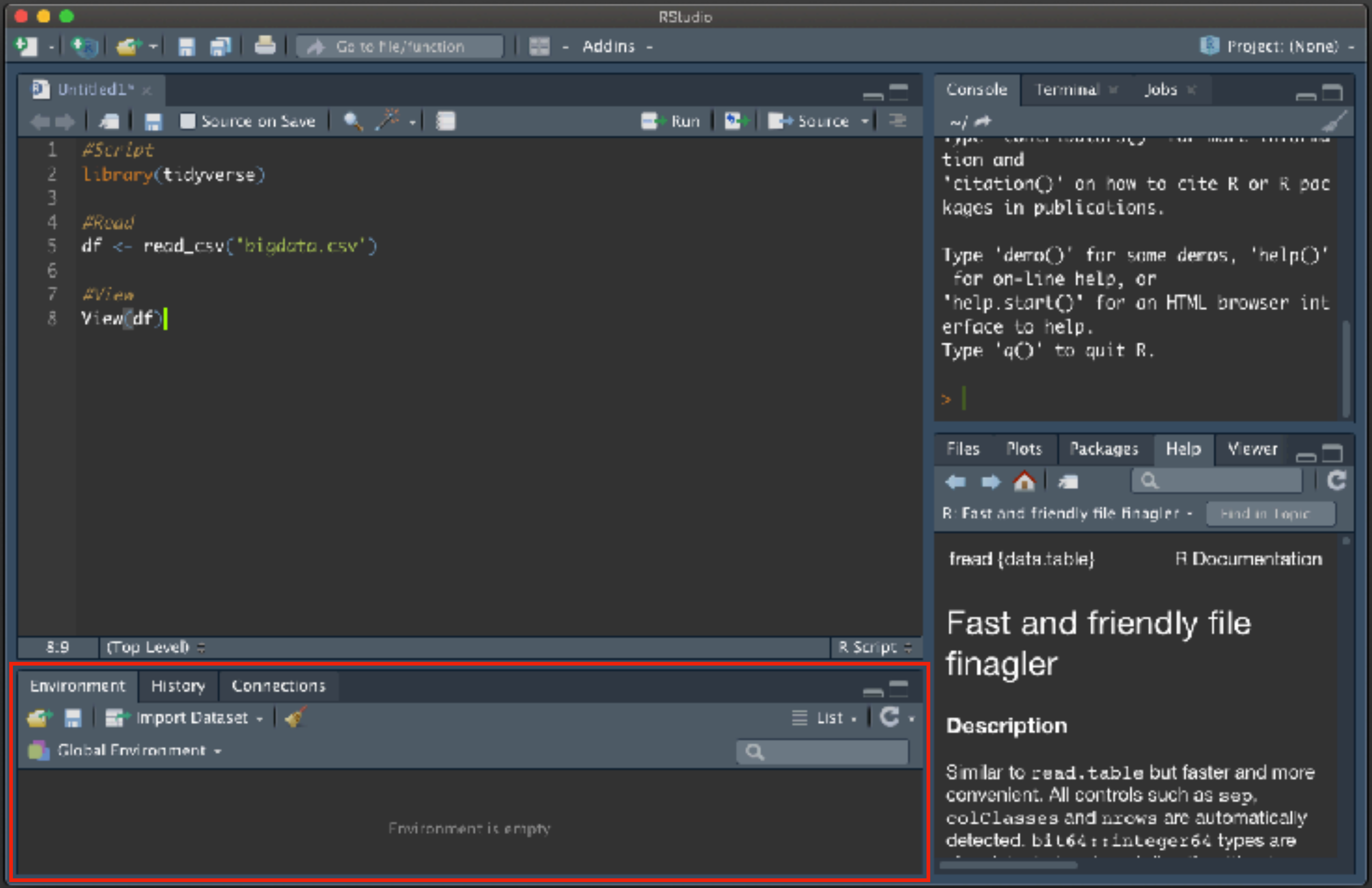
fread {data.table}　　R Documentation

Fast and friendly file finagler

Description

Similar to `read.table` but faster and more convenient. All controls such as `sep`, `colClasses` and `nrows` are automatically detected. `bit64::integer64` types are

IDE



```
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

```
tion and
'citation()' on how to cite R or R pac
kages in publications.

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

>
```

fread {data.table} R Documentation

Fast and friendly file finagler

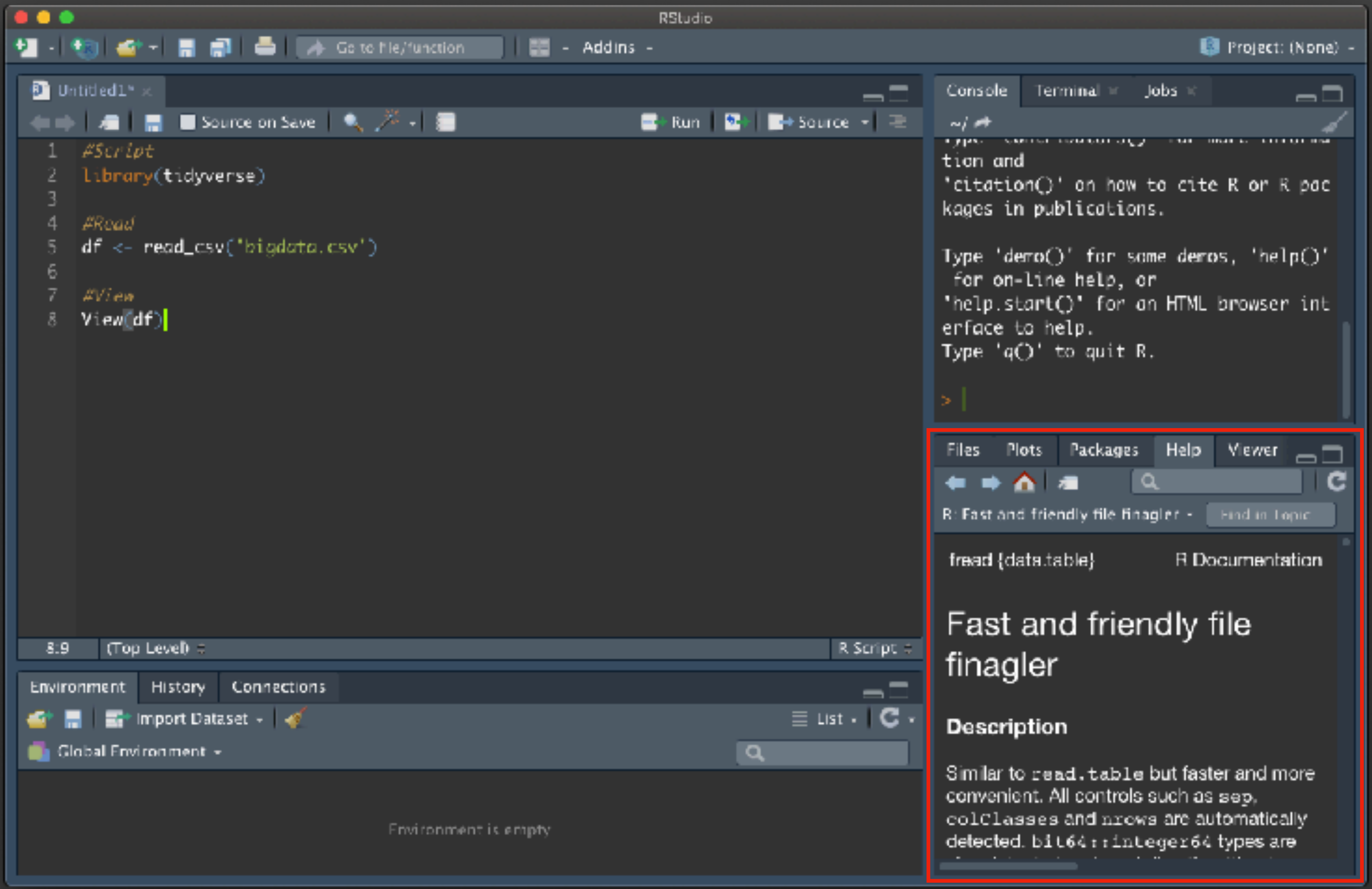
Description

Similar to `read.table` but faster and more convenient. All controls such as `sep`, `colClasses` and `nrows` are automatically detected. `bit64::integer64` types are

Environment is empty

Assistant's memory

IDE



```
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

```
tion and
'citation()' on how to cite R or R pac
kages in publications.

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

>
```

fread {data.table} R Documentation

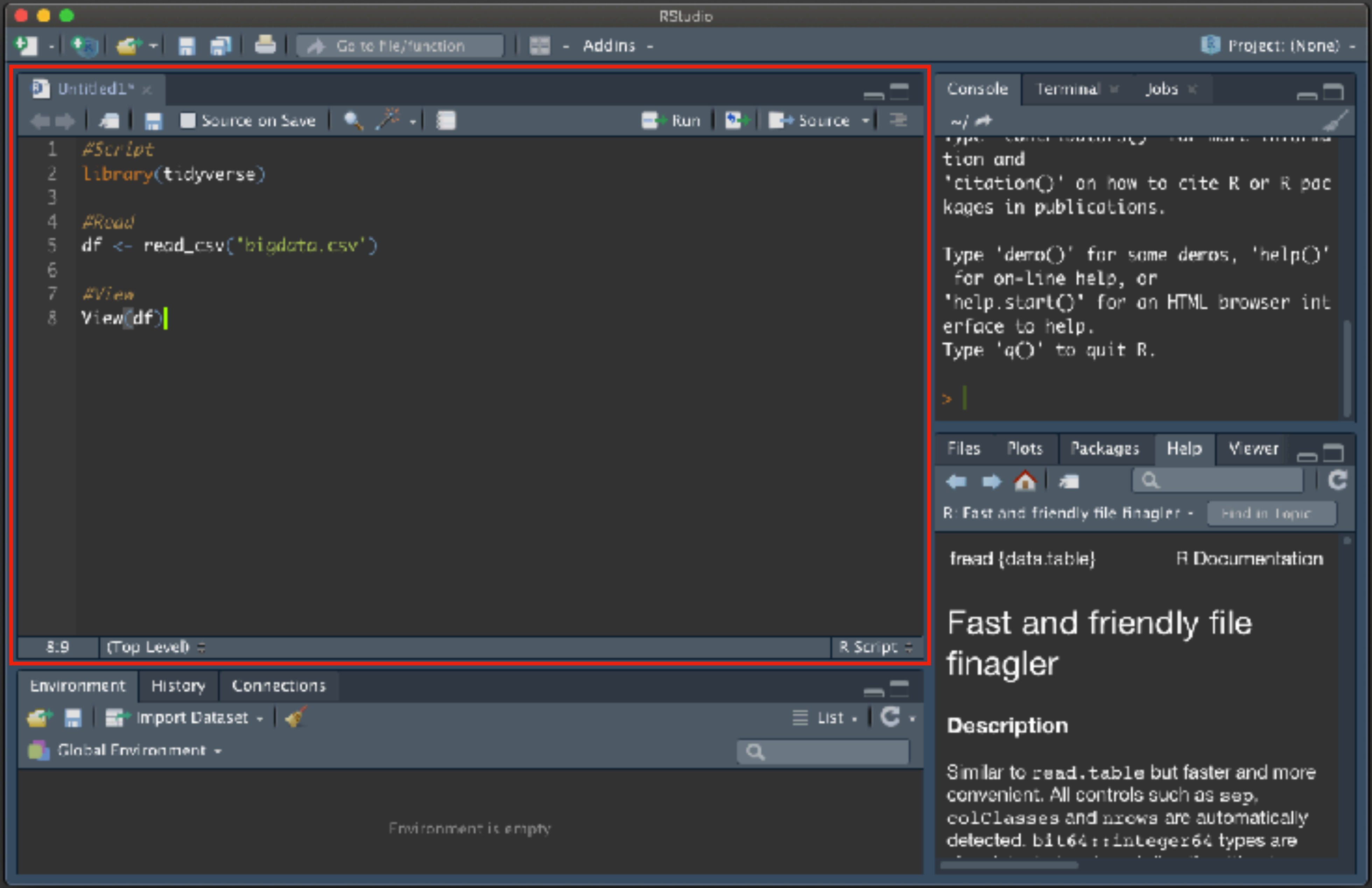
Fast and friendly file finagler

Description

Similar to `read.table` but faster and more convenient. All controls such as `sep`, `colClasses` and `nrows` are automatically detected. `bit64::integer64` types are

Archive

IDE



```r
#Script
library(tidyverse)

#Read
df <- read_csv('bigdata.csv')

#View
View(df)
```

```
tion and
'citation()' on how to cite R or R pac
kages in publications.

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

>
```

fread {data.table} R Documentation

Fast and friendly file finagler

Description

Similar to `read.table` but faster and more convenient. All controls such as `sep`, `colClasses` and `nrows` are automatically detected. `bit64::integer64` types are

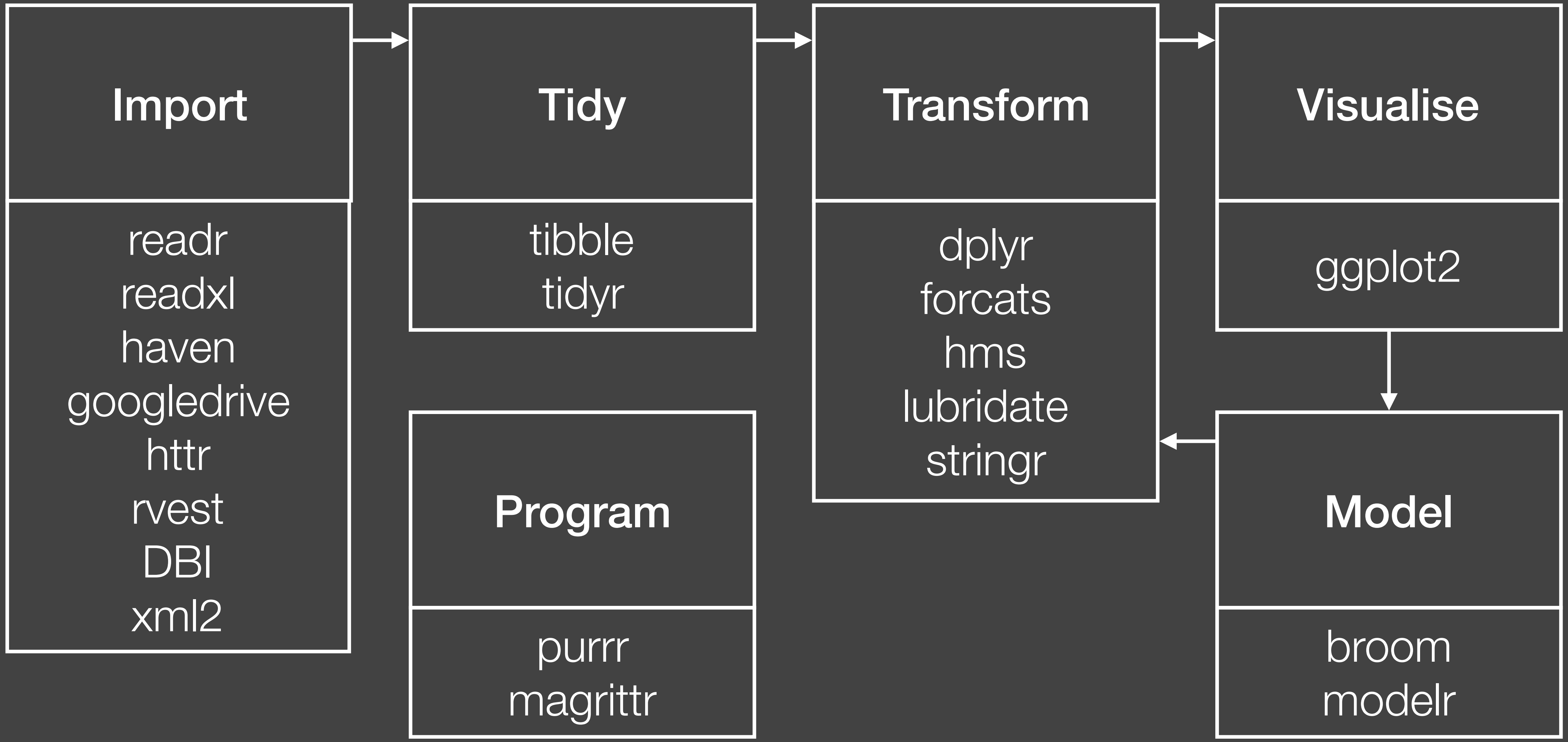
Instructions

Tidyverse

Collection of R packages

Tidy data

Tidyverse



Import

readr
readxl
haven
googledrive
httr
rvest
DBI
xml2

Tidy

tibble
tidyr

Transform

dplyr
forcats
hms
lubridate
stringr

Visualise

ggplot2

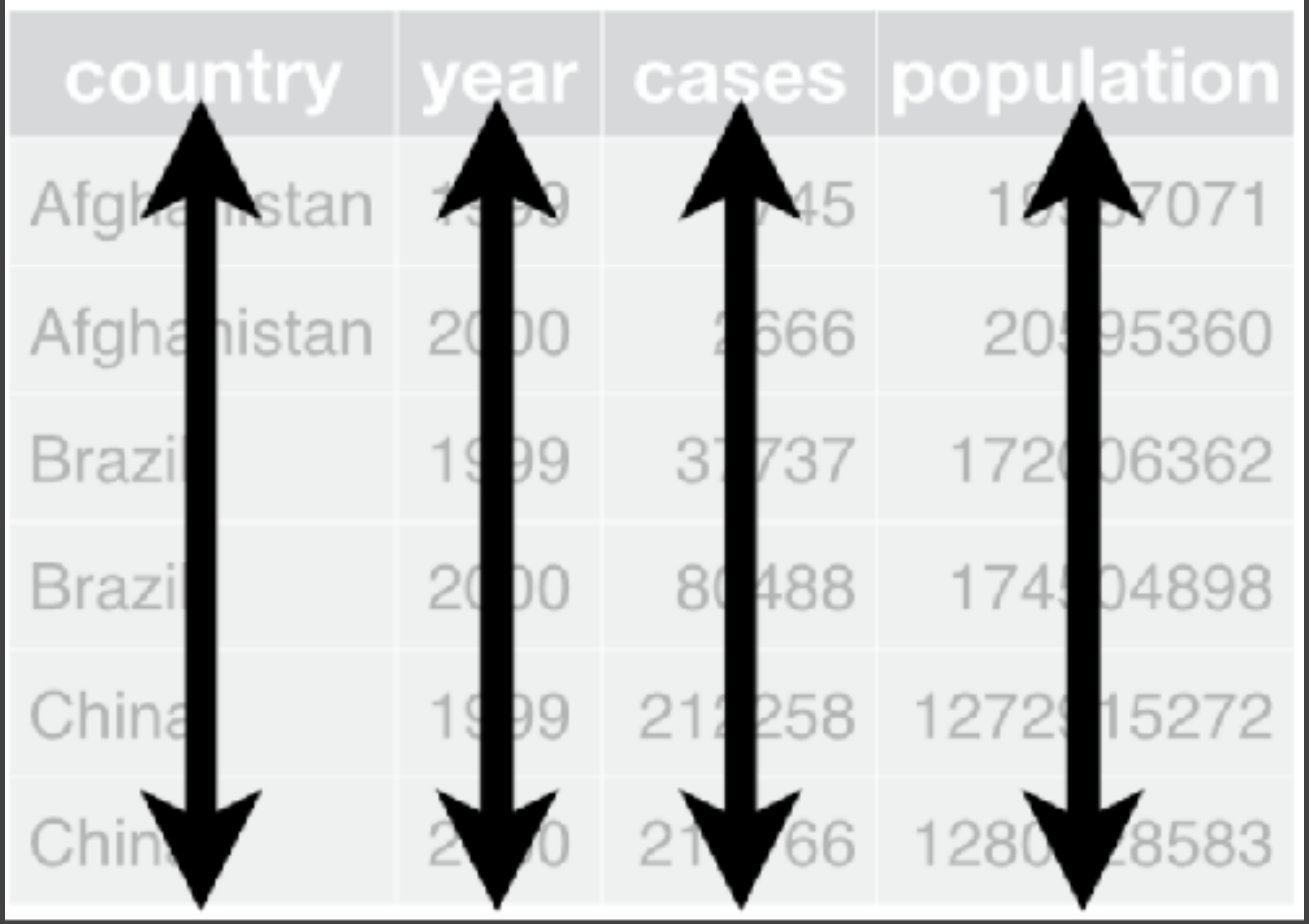
Model

broom
modelr

Program

purrr
magrittr

Tidy data

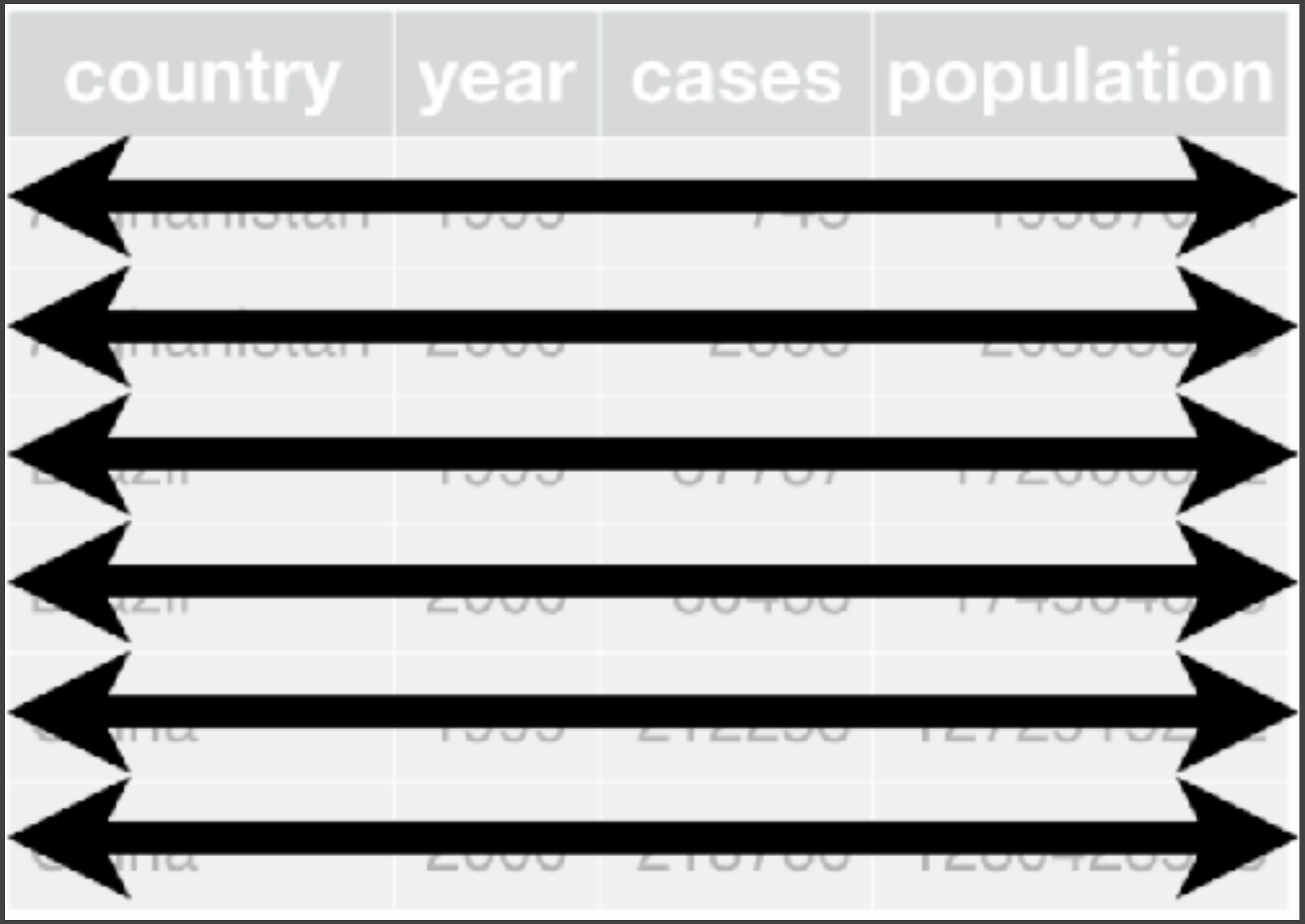


country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Variables

Hadley Wickham (2014), Tidy Data, *Journal of Statistical Software* 59(10)

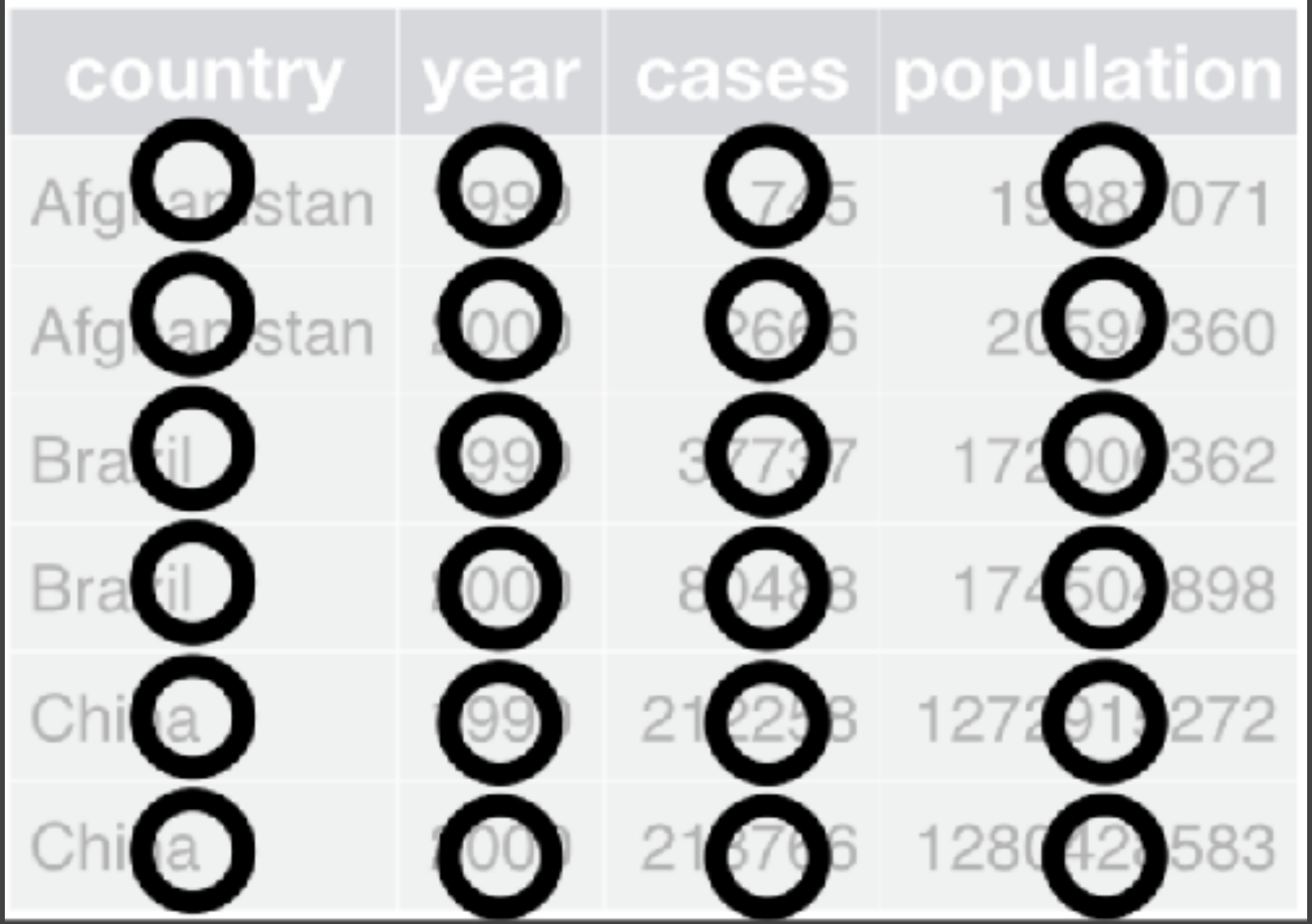
Tidy data



Observations

Hadley Wickham (2014), Tidy Data, *Journal of Statistical Software* 59(10)

Tidy data



country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Values

Hadley Wickham (2014), Tidy Data, *Journal of Statistical Software* 59(10)

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.

Hadley Wickham (2014), Tidy Data, *Journal of Statistical Software* 59(10)

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

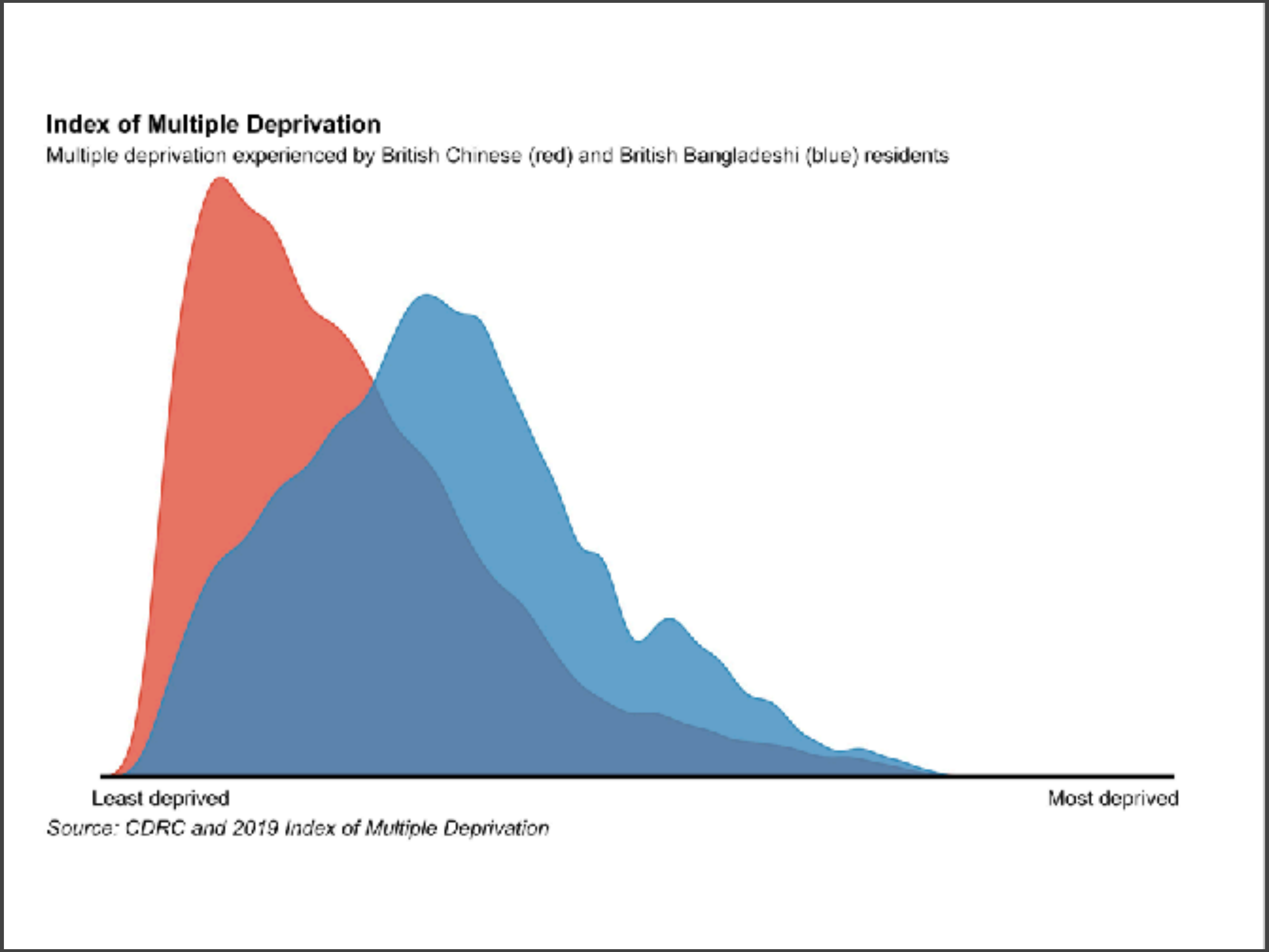
Tidy data

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

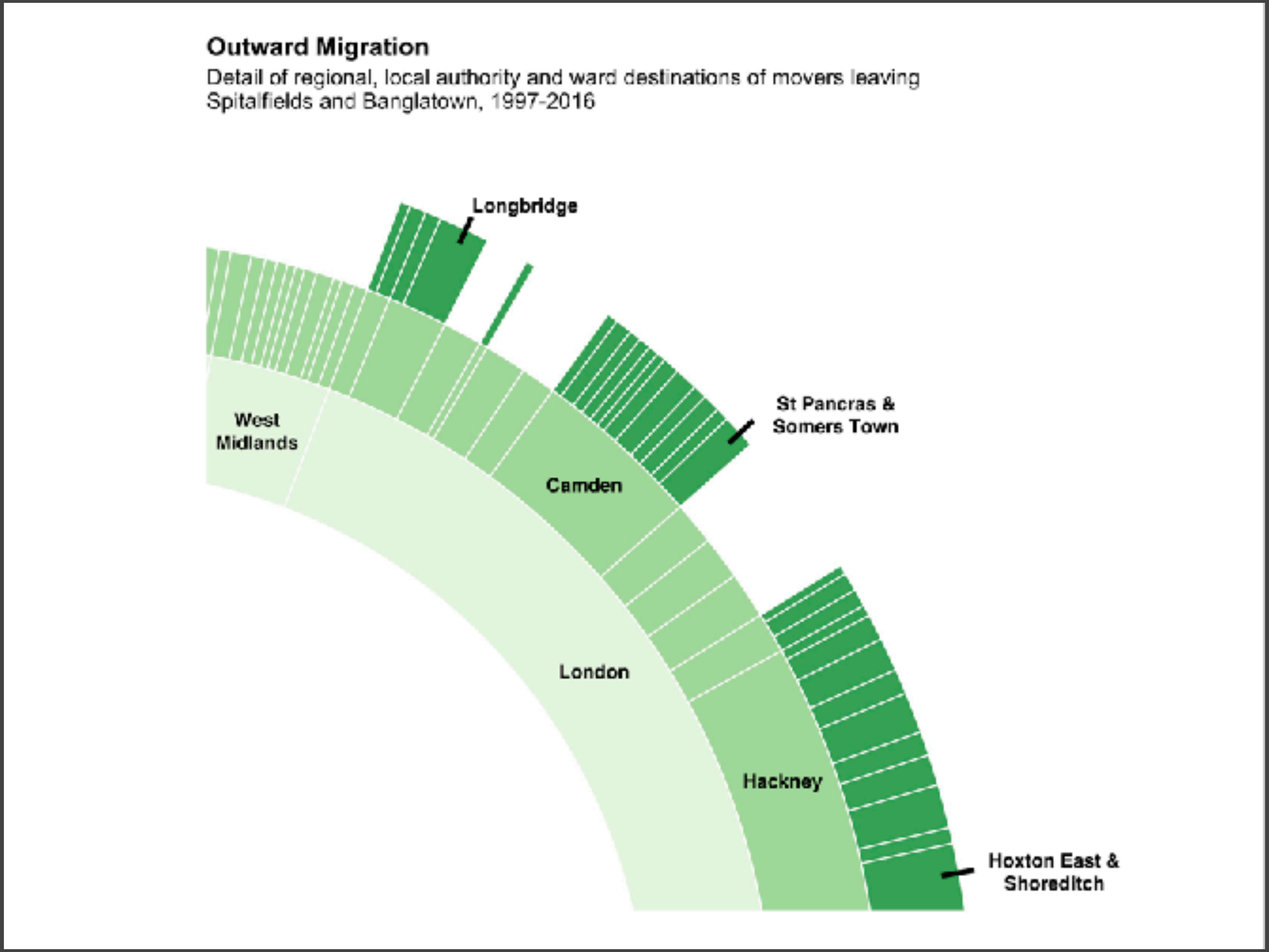


Index of Multiple Deprivation

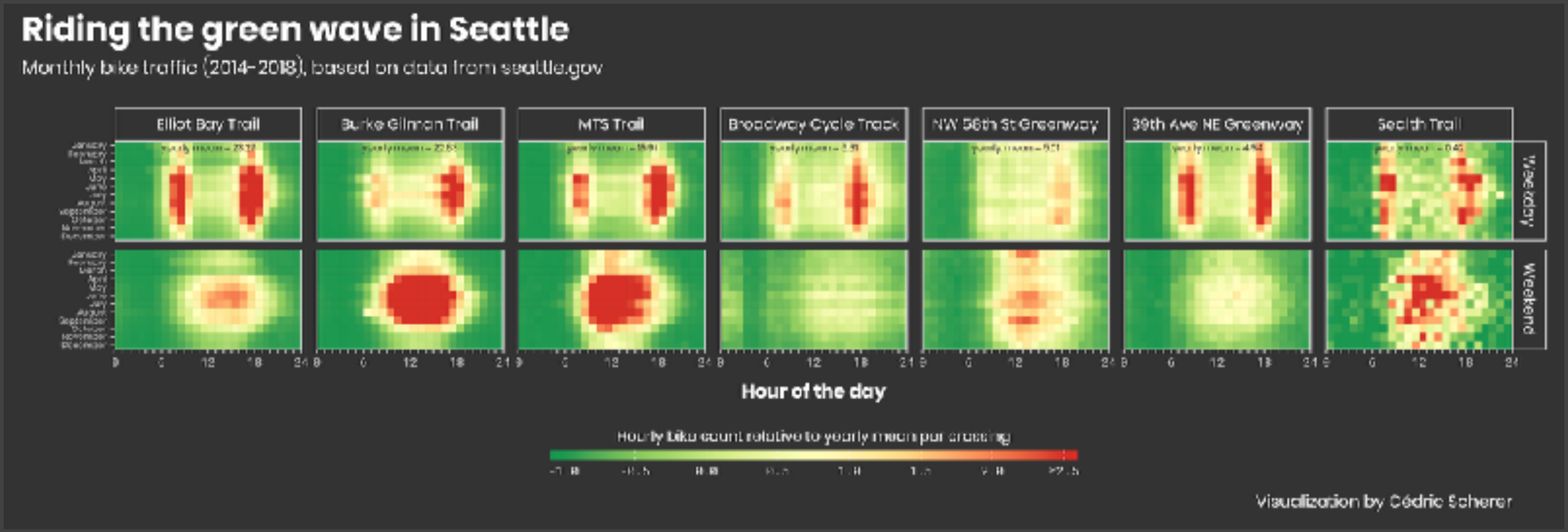
Multiple deprivation experienced by British Chinese (red) and British Bangladeshi (blue) residents

Source: CDRC and 2019 Index of Multiple Deprivation

ggplot2

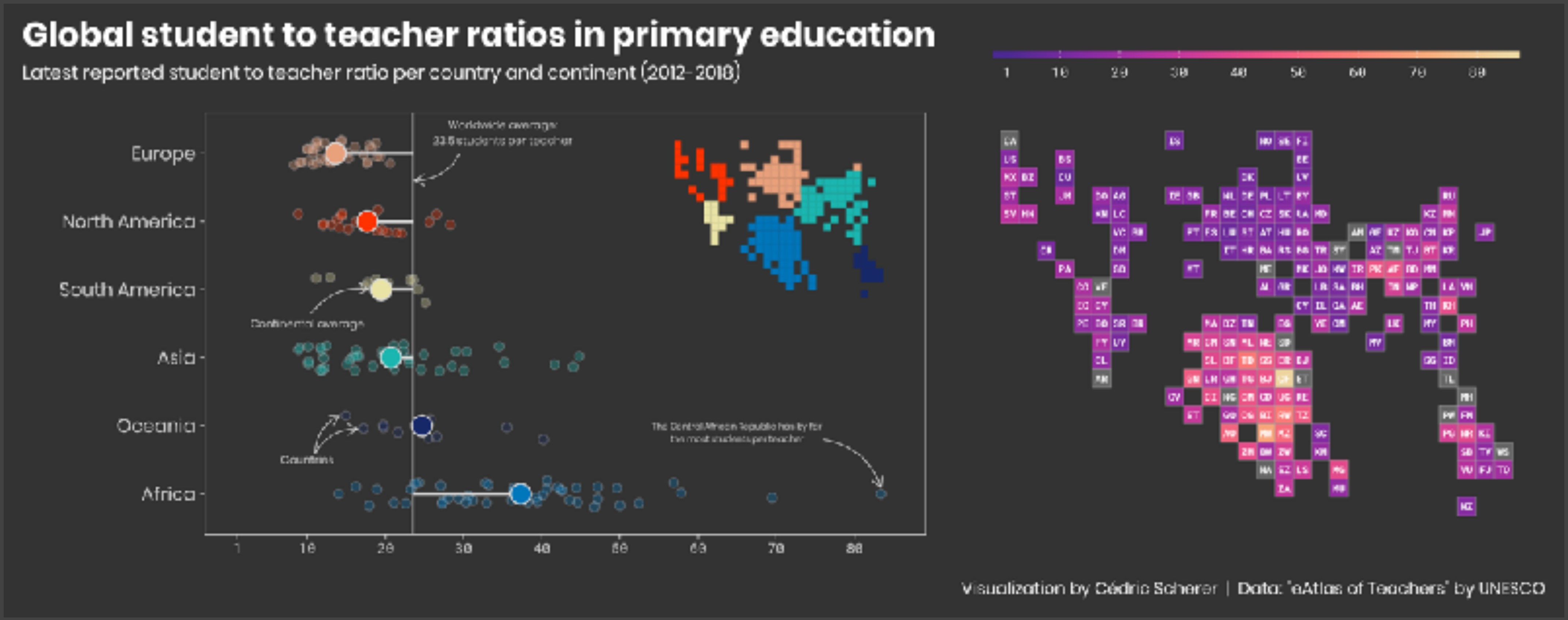

Outward Migration
Detail of regional, local authority and ward destinations of movers leaving Spitalfields and Banglatown, 1997-2016
Longbridge
St Pancras & Somers Town
West Midlands
Camden
London
Hackney
Hoxton East & Shoreditch

ggplot2



https://github.com/Z3tt/TidyTuesday

ggplot2



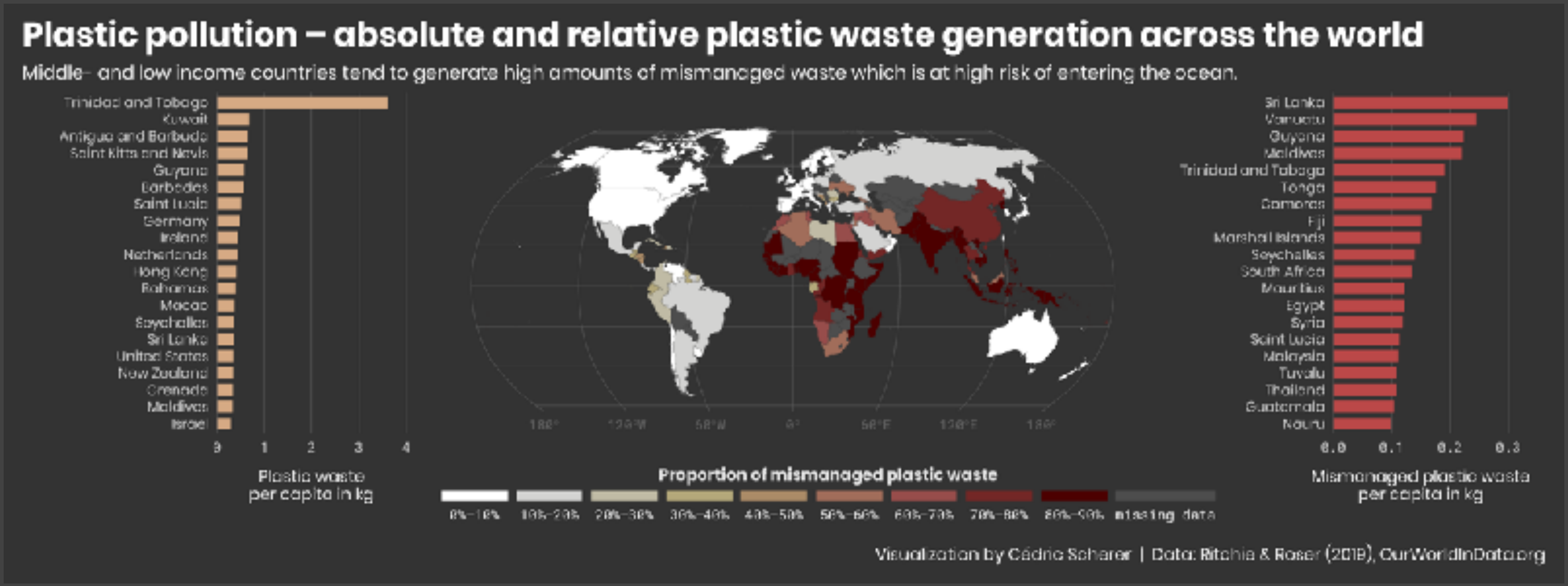
Global student to teacher ratios in primary education

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

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

https://github.com/Z3tt/TidyTuesday

ggplot2

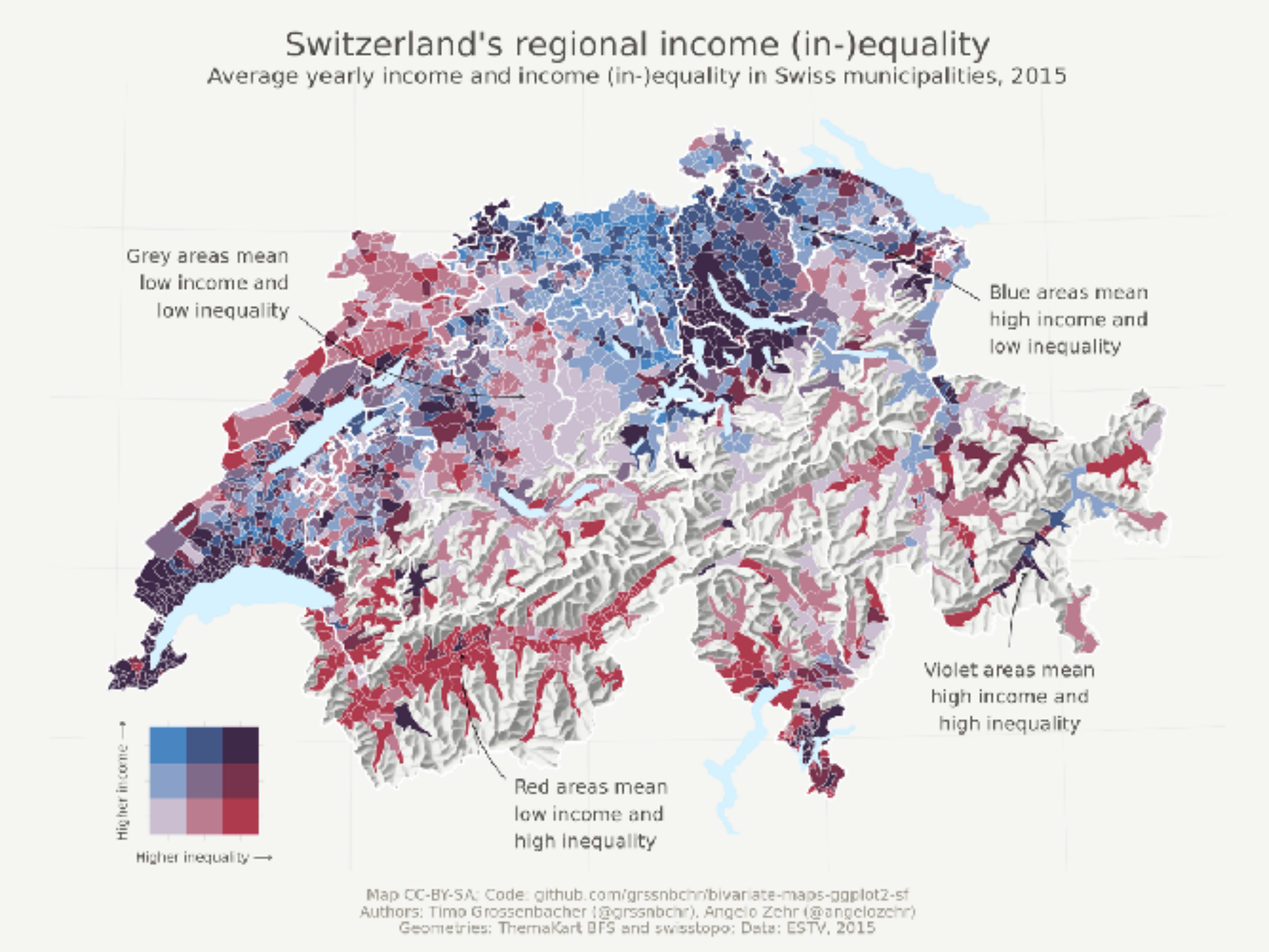

Plastic pollution – absolute and relative plastic waste generation across the world

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

https://github.com/Z3tt/TidyTuesday

ggplot2



Switzerland's regional income (in-)equality

Average yearly income and income (in-)equality in Swiss municipalities, 2015

Grey areas mean low income and low inequality

Blue areas mean high income and low inequality

Violet areas mean high income and high inequality

Red areas mean low income and high inequality

Map CC-BY-SA; Code: github.com/grssnbchr/bivariate-maps-ggplot2-sf
Authors: Timo Grossenbacher (@grssnbchr), Angelo Zehr (@angelozehr)
Geometries: ThemaKart BFS and swisstopo; Data: ESTV, 2015

https://timogrossenbacher.ch/2019/04/bivariate-maps-with-ggplot2-and-sf/

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/

useR!

https://www.r-project.org/conferences/

R-Ladies

https://rladies.org

Stackoverflow

~ 300,00 questions

#rstats

Twitter

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
Relational databases	RMySQL, RPostgresSQL, RSQLite
Various file formats	XLConnect, xlsx, foreign, haven

Manipulating data

What for	Packages
Data wrangling	dplyr, tidyr
Text	stringr, tidytext
Dates	lubridate

Modelling data

What for	Packages
Statistical models	caret, randomForest, glmnet, nnet, mice

Dealing with spatial data

What for	Packages
Spatial	sp, maptools, ggmap, tmap

Data visualisation

What for	Packages
Graphics	ggplot, ggraph, ggtext

Data output

What for	Packages
Results	R Markdown, shiny

General performance

What for	Packages
Large datasets	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(time, 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 m 14 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