Exploratory Data Analysis (with R)

Workshop @ University of Oxford on 4th December 2017

Martin John Hadley
@martinjhnhadley

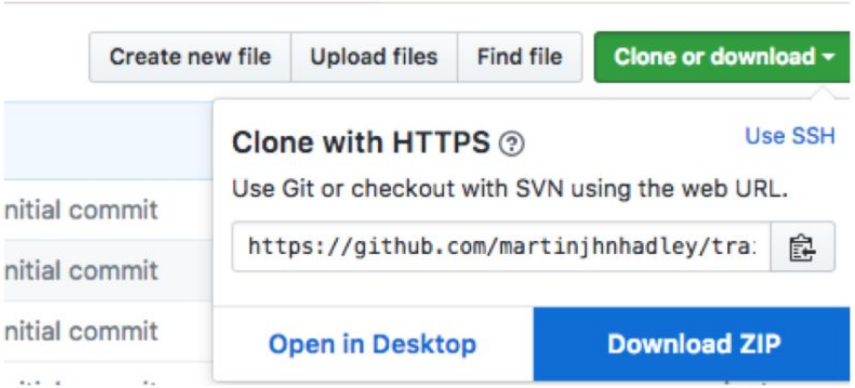
What do I need on my computer?

To participate in this workshop, please sure you have the following:

- An installation of R https://cran.r-project.org/
- An installation of RStudio https://www.rstudio.com/products/rstudio/download/

As close as possible to the course please download the materials in this repository (https://github.com/martinjhnhadley/2017_data-viz-missing-link). Here's a short URL https://goo.gl/9syL1E.

Click on "Clone or download" and select "Download as ZIP". Ensure to unzip the folders to a convenient location.



What is exploratory data analysis?

Exploratory data analysis (EDA) is the buzzword given to gaining a summary understanding of your datasets - particularly important when working with the following types of data:

- Data mining (website scrapings, meta analysis, “big data”)
- Survey data

EDA is about exploring data without a priori hypotheses.

Why use visualisations in EDA?

Visualisations allow large, complicated datasets to be reduced to digestible summaries.

Using lots of different visualisations allows us to get a real **feel** for our data.

Visualisations are therefore not only extremely powerful for our own EDA but also in communicating our research findings to others.

Why use interactivity in visualisations?

Providing interactivity in the visualisations we build for the public allows them to replicate (in a hand wavy way) the EDA we went through ourselves.

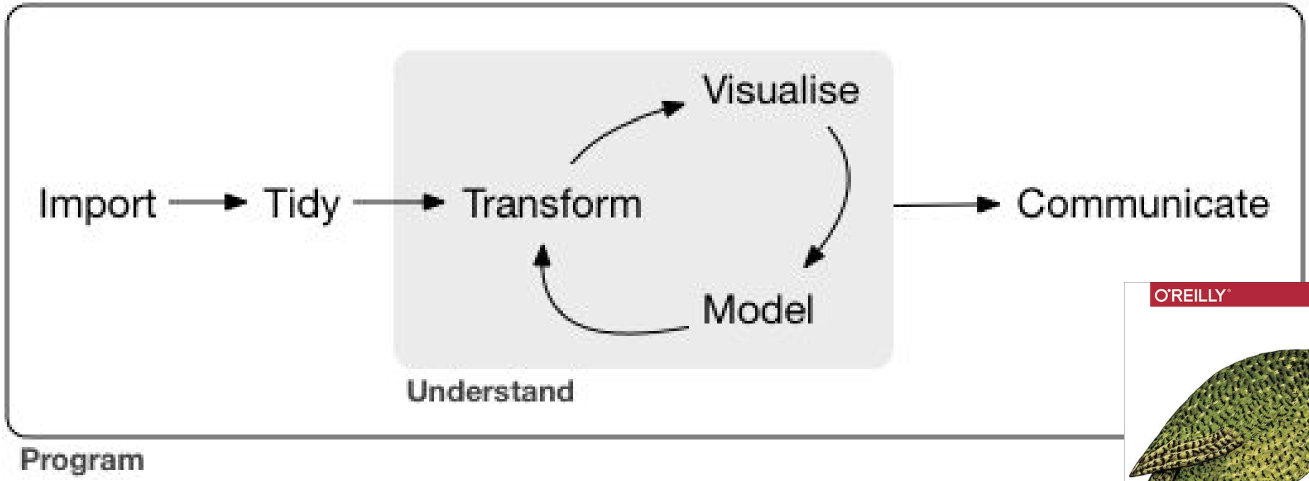
Interactivity provides an exploratory view on to the data, rather than a fixed narrative provided by static charts.

What’s this workshop on?

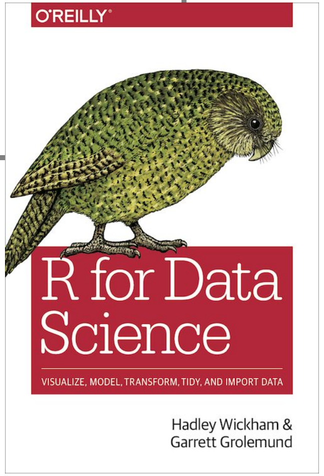
I’m going to try and persuade you that R is an excellent tool for EDA and for building interactive visualisations.

I’m also going to focus on the importance of a reproducible workflow in your analysis.

What's our workflow look like?



r4ds.had.co.nz/



O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund

What are our tools?

RStudio

tidyverse

%>%

Ceci n'est pas un pipe.

htmlwidgets

R is the programming (or scripting) language we’re using to make things happen

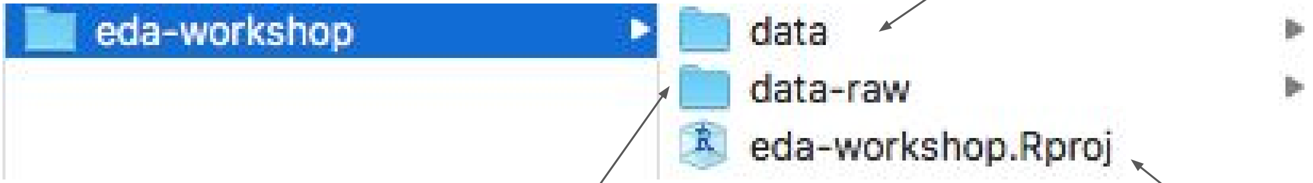
If you need a quick intro:
datacamp.com/courses/intro-to-statistics-with-r-introduction

RStudio is *the* IDE for data science (and package development) for R.

The company behind it - RStudio - build tools which work seamlessly together to make your lives easier (and to make R more powerful).

Organising your work

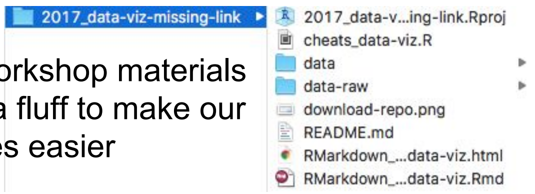
Cleaned, anonymised data for depositing somewhere that issues DOI

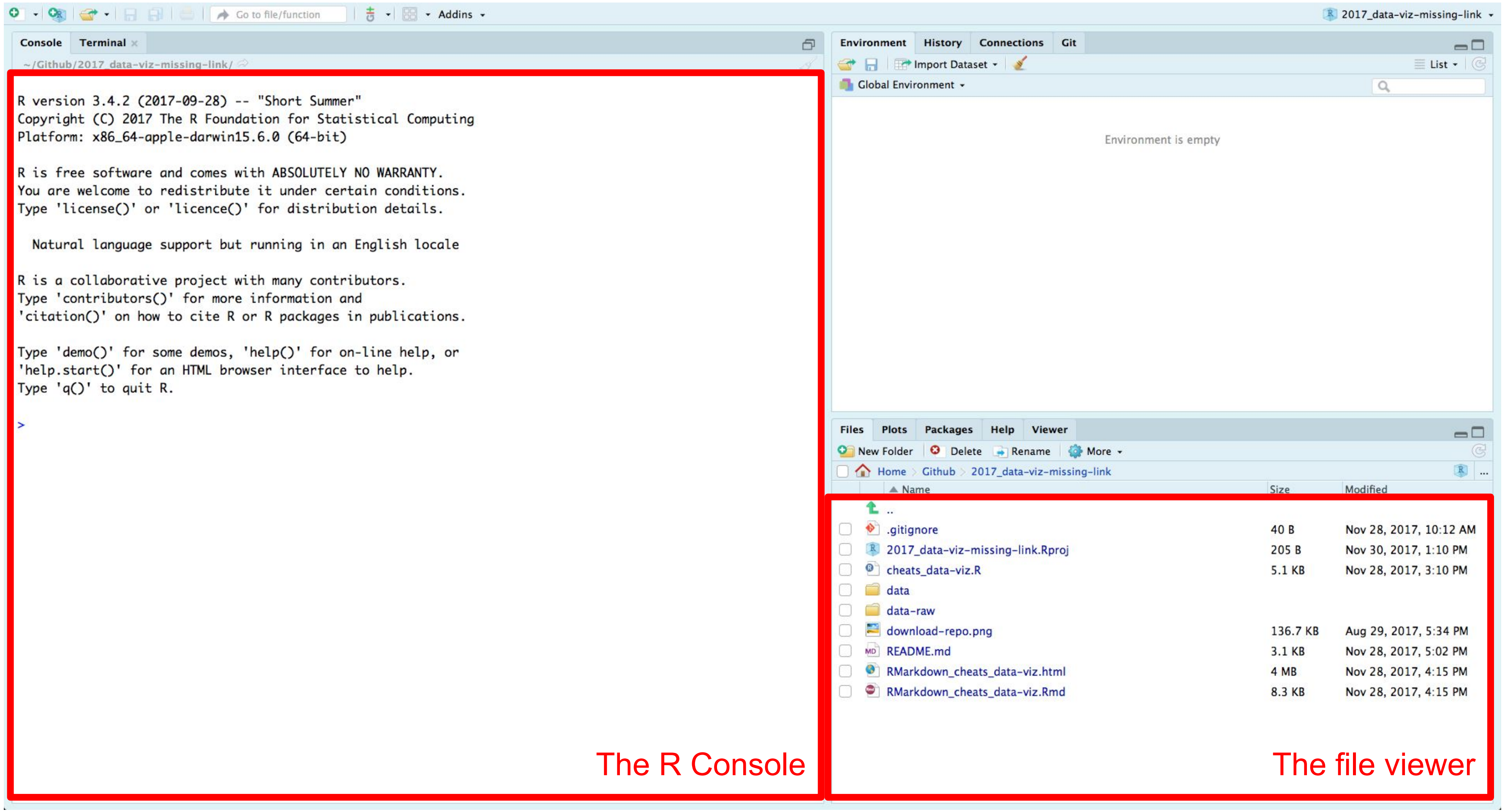


Raw, unedited data which is treated as “read-only”.
Add scripts here to clean the data and output new files to /data

.Rproj files help make your R code transportable by making all file paths relative to the /eda-workshop folder

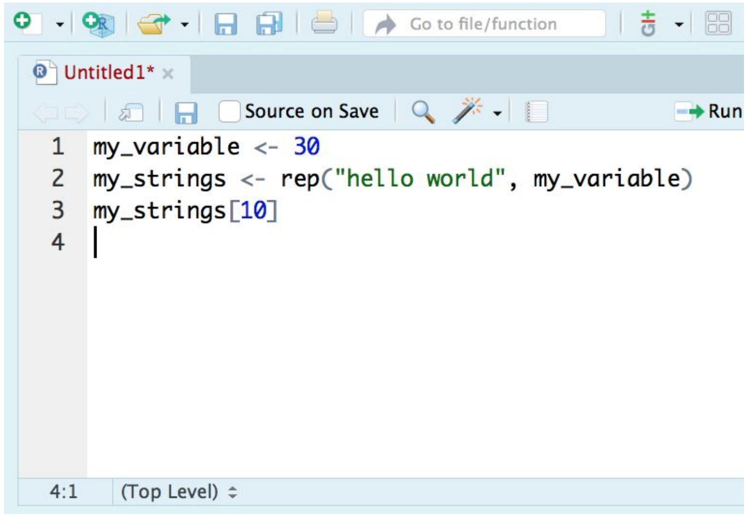
Note: Our workshop materials contain extra fluff to make our lives easier


2017_data-viz-missing-link
2017_data-v...ing-link.Rproj
cheats_data-viz.R
data
data-raw
download-repo.png
README.md
RMarkdown_...data-viz.html
RMarkdown_...data-viz.Rmd



```
R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 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.

>
```

Name	Size	Modified
..		
.gitignore	40 B	Nov 28, 2017, 10:12 AM
2017_data-viz-missing-link.Rproj	205 B	Nov 30, 2017, 1:10 PM
cheats_data-viz.R	5.1 KB	Nov 28, 2017, 3:10 PM
data		
data-raw		
download-repo.png	136.7 KB	Aug 29, 2017, 5:34 PM
README.md	3.1 KB	Nov 28, 2017, 5:02 PM
RMarkdown_cheats_data-viz.html	4 MB	Nov 28, 2017, 4:15 PM
RMarkdown_cheats_data-viz.Rmd	8.3 KB	Nov 28, 2017, 4:15 PM

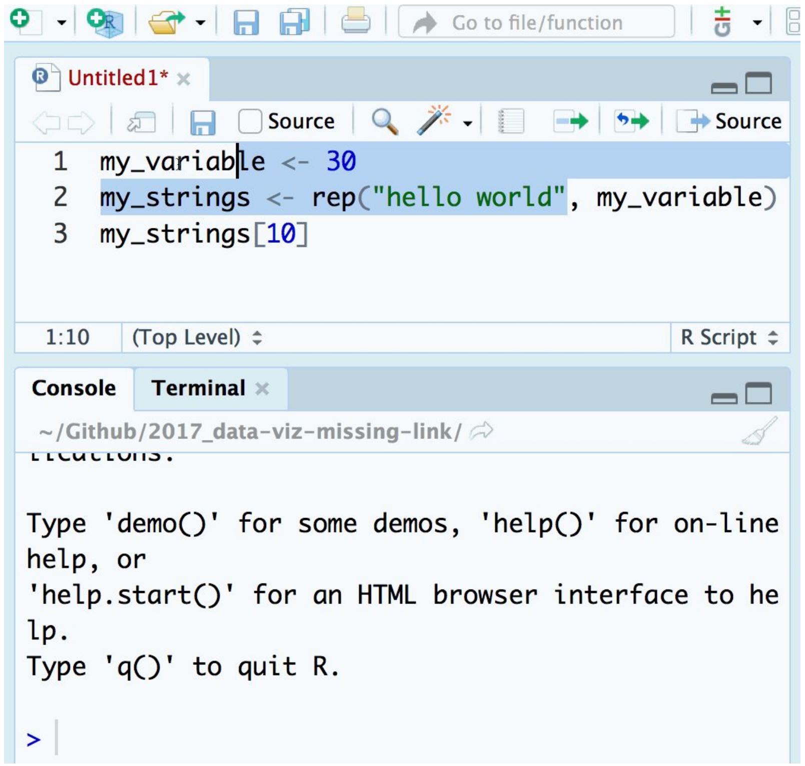


```
my_variable <- 30
my_strings <- rep("hello world", my_variable)
my_strings[10]
```

- Assignments made with <-
  Called the assignment operator or "the mushroom"

- Round brackets ( ) contain arguments to functions
  Just like in Excel

- Square brackets [ ] are used for extracting parts from something
  Often called "indexing"



```
my_variable <- 30
my_strings <- rep("hello world", my_variable)
my_strings[10]
```

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

>
```

- Select code you want to “run” and press:
  Cmd+Enter in macOS
  Ctrl+Enter in Windows
- You will often (unfortunately) misselect code and get unexpected stuff in the console
- The + in the console indicates its waiting for more input
  Put you cursor in here and hit Esc

Importing data-raw/journeys.csv

Whenever importing data from .csv, .tsv, .xlsx or other plain text file into RStudio it's highly recommend that you use the readr or readxl library from the tidyverse.

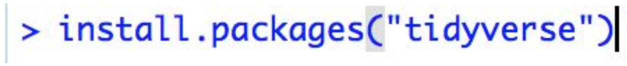
The tidyverse provides a consistent and elegant approach to doing data science with R.

The tidyverse is an ecosystem of packages maintained and developed by the folks at RStudio.

Development of the tidyverse packages started in late 2014 but only became formalised as the "tidyverse" in late 2016.

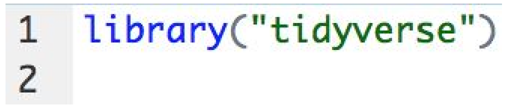
Using packages on your machine

1. Run install.packages(“packagename”) in the console.
   This only needs to be done once.



```
> install.packages("tidyverse")
```

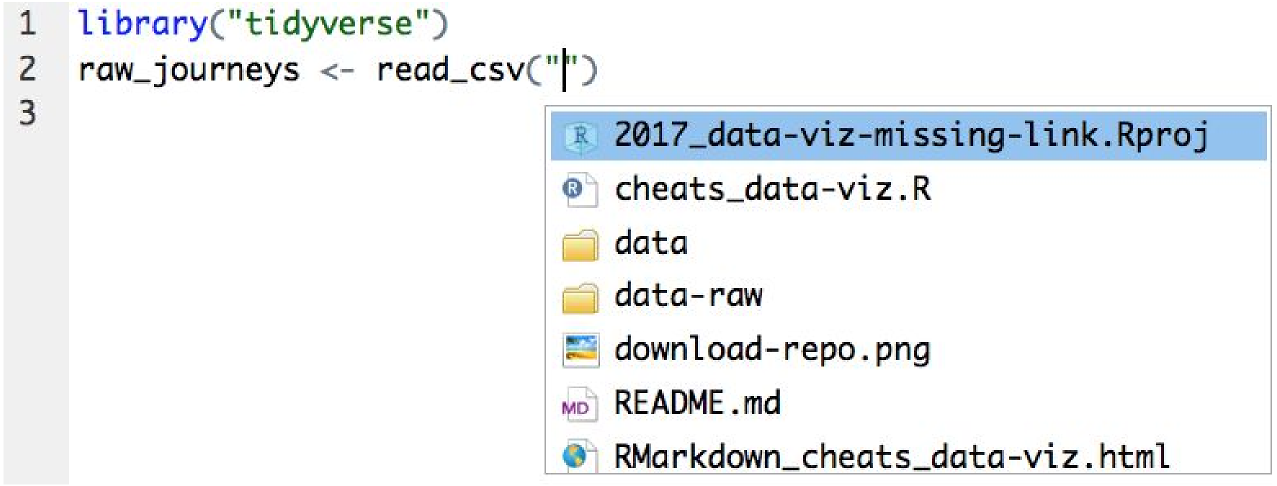
2. Run library(“packagename”) in your script to load the library.
   This needs to be done before you use the package.



```
library("tidyverse")
```

Importing data-raw/journeys.csv

1. Load the tidyverse library
2. Decide on an appropriate variable name
3. Use the file explorer to find your file by pressing TAB

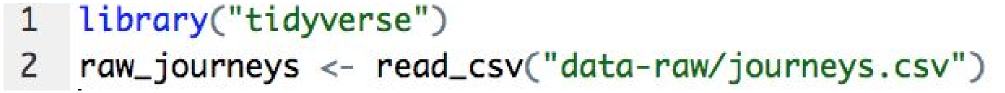


```r
library("tidyverse")
raw_journeys <- read_csv("")

```

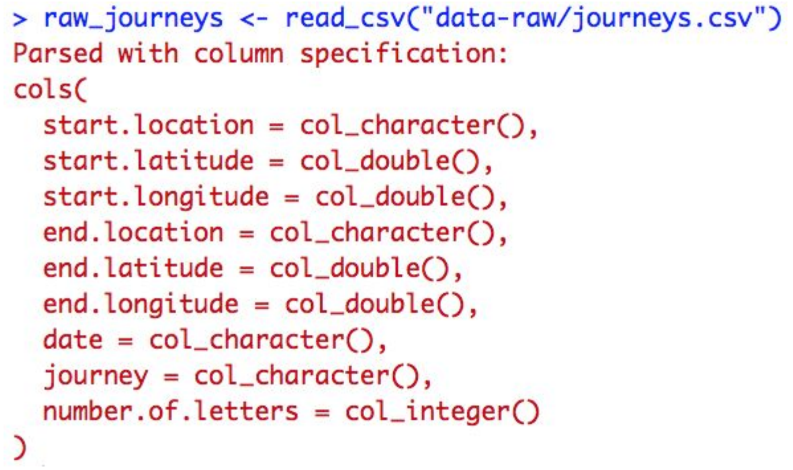
2017_data-viz-missing-link.Rproj
cheats_data-viz.R
data
data-raw
download-repo.png
README.md
RMarkdown_cheats_data-viz.html

Importing data-raw/journeys.csv

4. Run the code



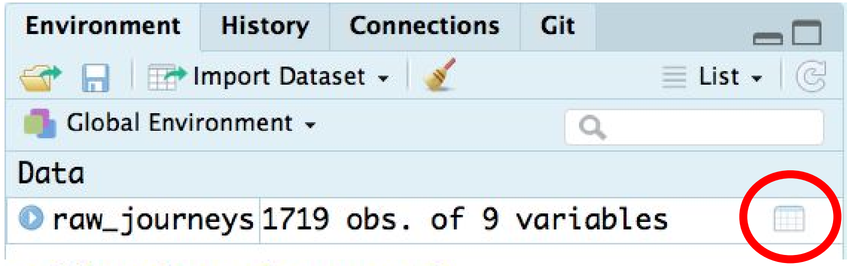
```r
library("tidyverse")
raw_journeys <- read_csv("data-raw/journeys.csv")
```



```
> raw_journeys <- read_csv("data-raw/journeys.csv")
Parsed with column specification:
cols(
  start.location = col_character(),
  start.latitude = col_double(),
  start.longitude = col_double(),
  end.location = col_character(),
  end.latitude = col_double(),
  end.longitude = col_double(),
  date = col_character(),
  journey = col_character(),
  number.of.letters = col_integer()
)
```

Importing data-raw/journeys.csv

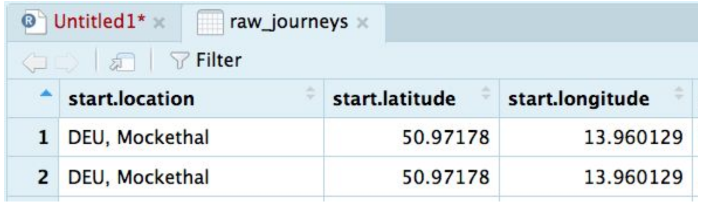
5. Check the data is imported as you expect



Data

raw_journeys 1719 obs. of 9 variables

```
> View(raw_journeys)
```



	start.location	start.latitude	start.longitude
1	DEU, Mockethal	50.97178	13.960129
2	DEU, Mockethal	50.97178	13.960129

```
> glimpse(raw_journeys)
Observations: 1,719
Variables: 9
$ start.location    <chr> "DEU, Mockethal", "DEU, Mockethal", "DEU, Mockethal",
$ start.latitude    <dbl> 50.97178, 50.97178, 50.97178, 50.97178, 50.97178, 50.9
$ start.longitude   <dbl> 13.96013, 13.96013, 13.96013, 13.96013, 13.96013, 13.9
$ end.location      <chr> "USA, New York (NY)", "USA, New York (NY)", "USA, New
```

Where possible, clean data with code

Using Excel to tidy data can be problematic as it’s difficult to proceduralise and reproduce easily.

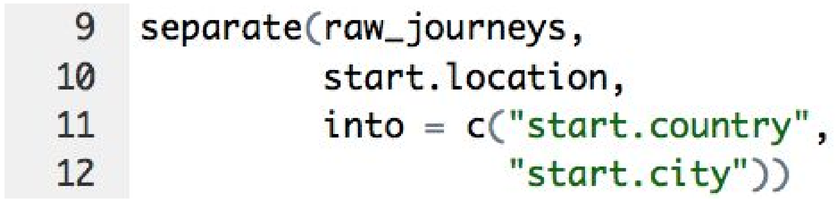
Cleaning your data with code means others can exactly replicate your data processing and this increases the reliability of your analysis.

The tidyverse contains two extremely powerful libraries for this data wrangling process:

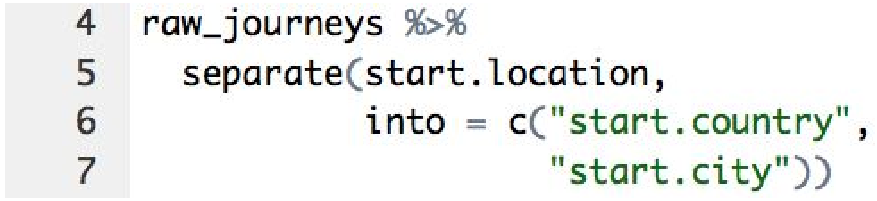
The separate verb

The tidyverse packages are composed of many functions that are best thought of as verbs, e.g. filter, select, gather, spread, mutate, summarise, separate...

The separate verb is used like the “text to columns” feature of Excel



```
separate(raw_journeys,
         start.location,
         into = c("start.country",
                  "start.city"))
```

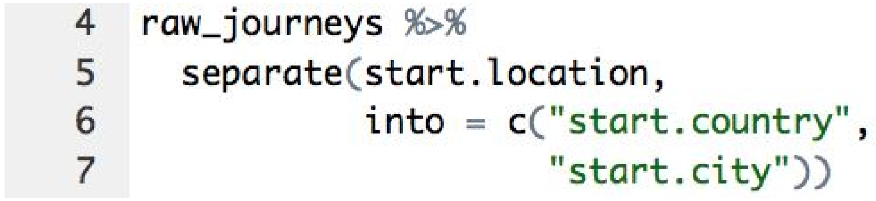


```
raw_journeys %>%
  separate(start.location,
           into = c("start.country",
                    "start.city"))
```

The pipe operator

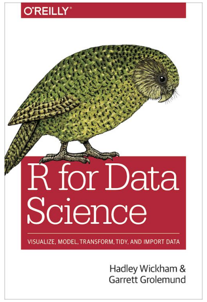
Understanding the pipe operator is a necessary part of learning the tidyverse, because the documentation pages, StackOverflow answers and the r4ds.had.co.nz book all use it.

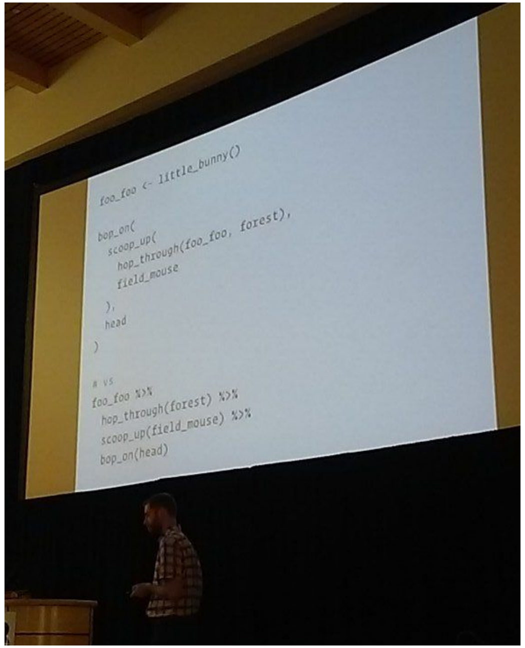
But it doesn't mean you have to use it.



```
raw_journeys %>%
  separate(start.location,
           into = c("start.country",
                    "start.city"))
```

I think the best motivating example for %>% comes from Hadley Wickham’s talk in useR2016.


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund

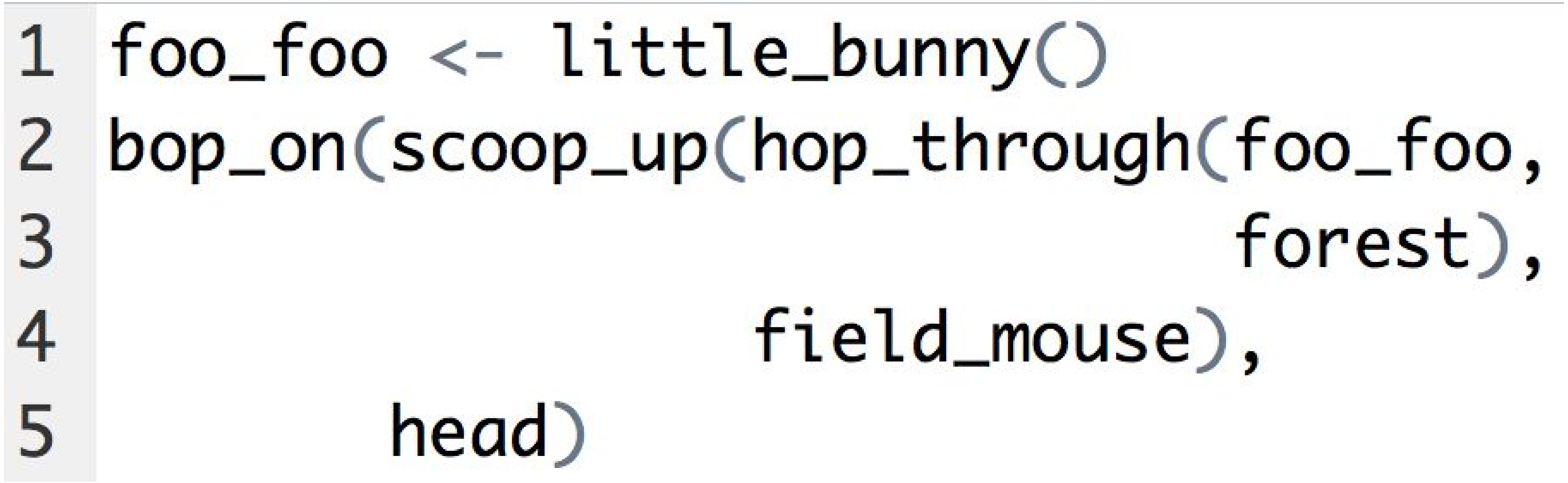


```
foo_foo <- little_bunny()

bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_mouse
  ),
  head
)

# vs
foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
```

Citation:
twitter.com/AmeliaMN/status/748193609401327616

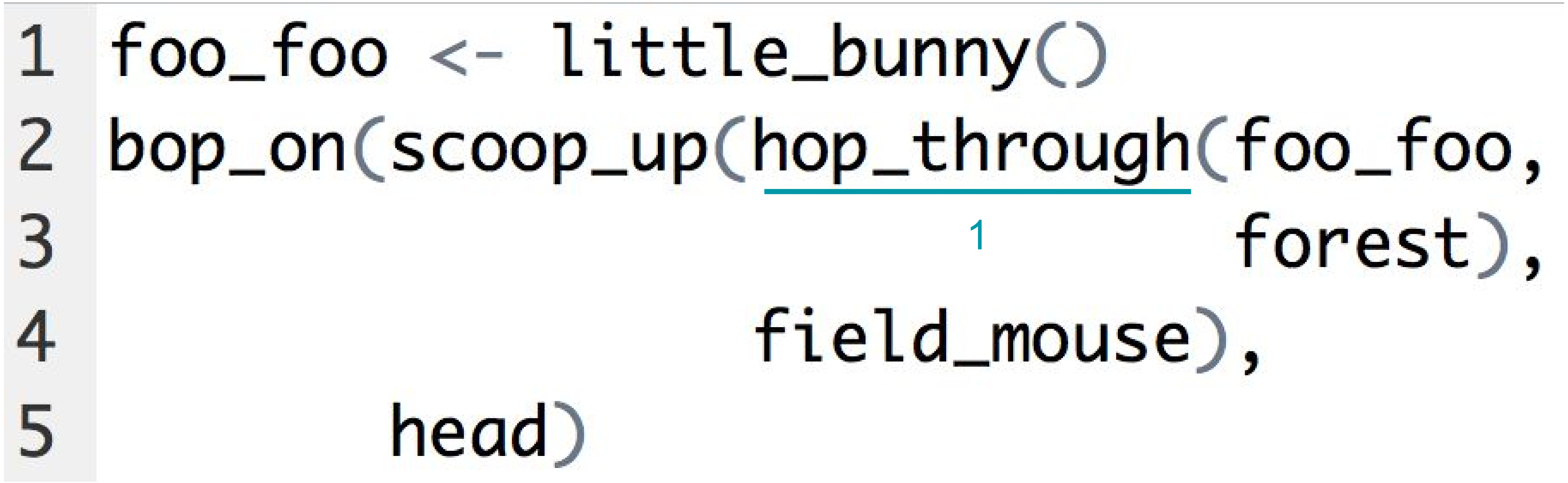
Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head

Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head



```r
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
                            forest),
                field_mouse),
       head)
```

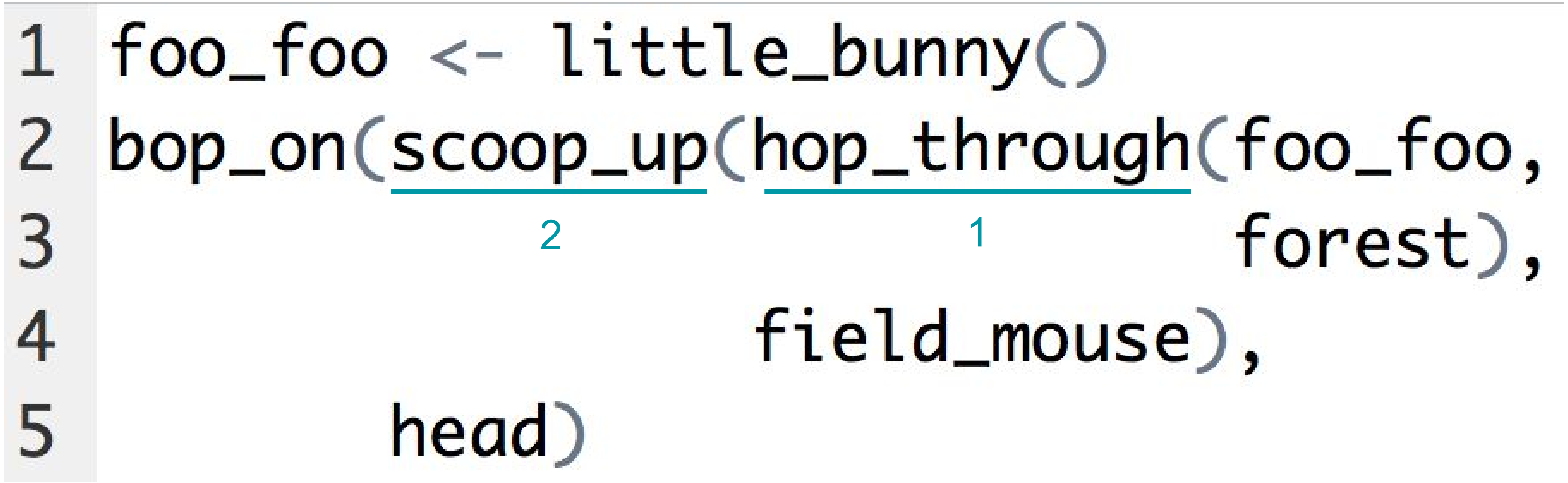
Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head



```
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
                            forest),
                field_mouse),
       head)
```

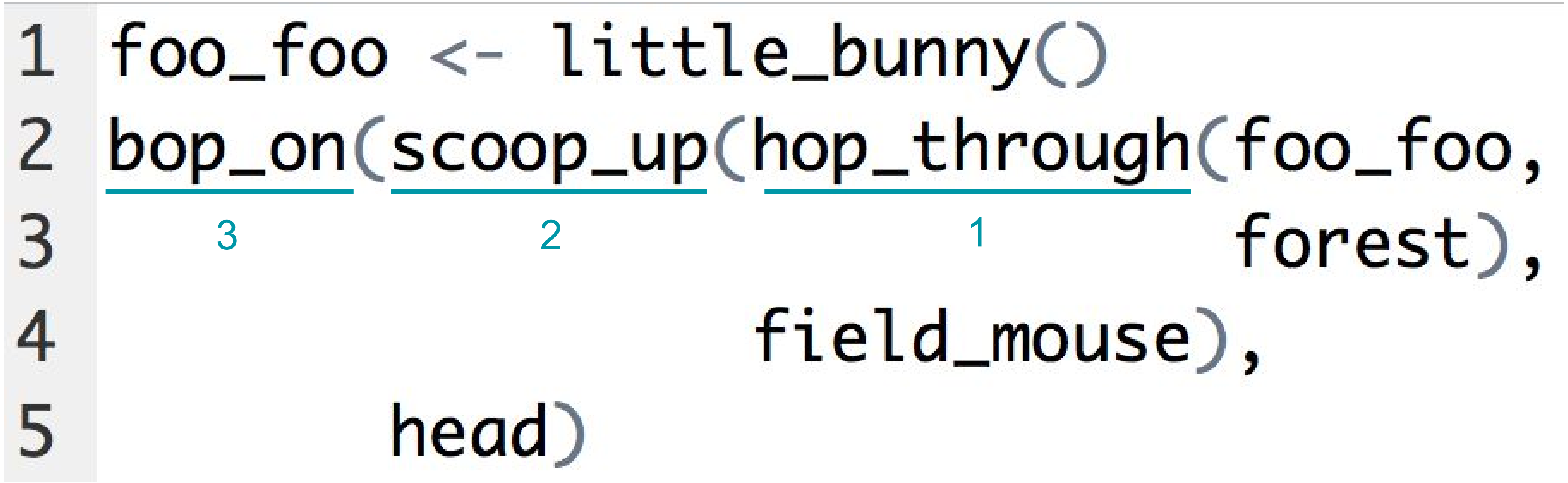
1

Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head

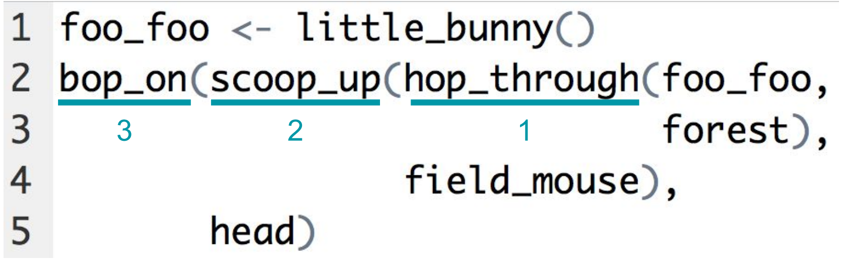


```
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
                            forest),
                field_mouse),
       head)
```

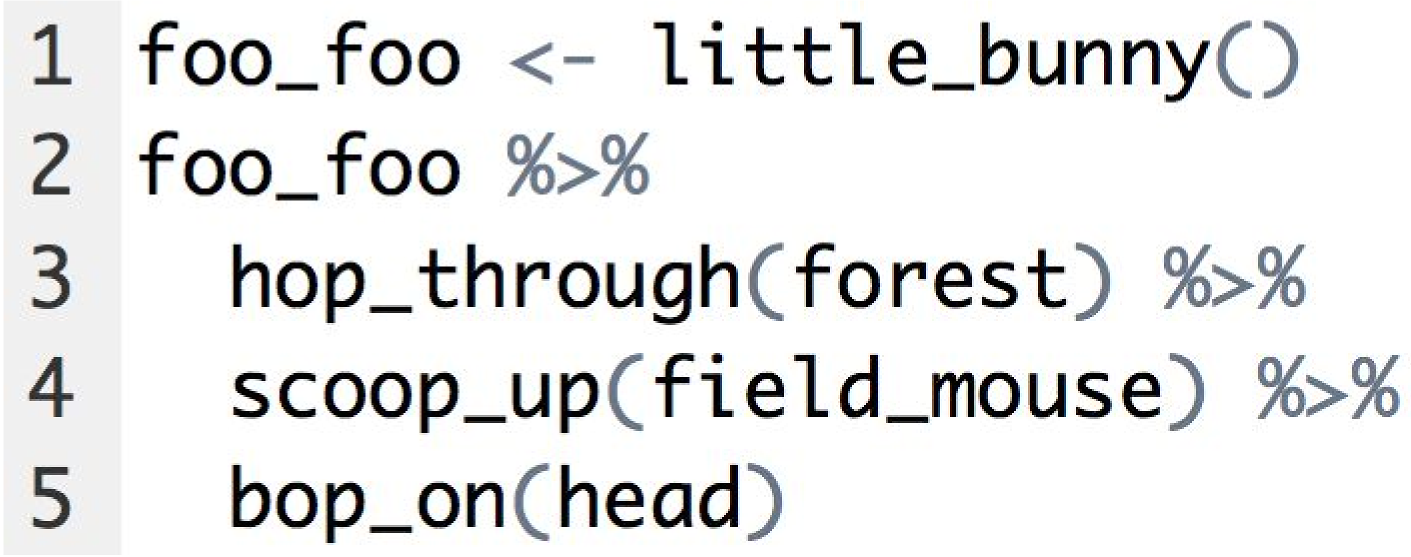
Little bunny Foo Foo
Went hopping through the forest
Scooping up the field mice
And bopping them on the head



```
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
                            forest),
                field_mouse),
       head)
```



```r
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
                            forest),
                field_mouse),
       head)
```



```r
foo_foo <- little_bunny()
foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
```

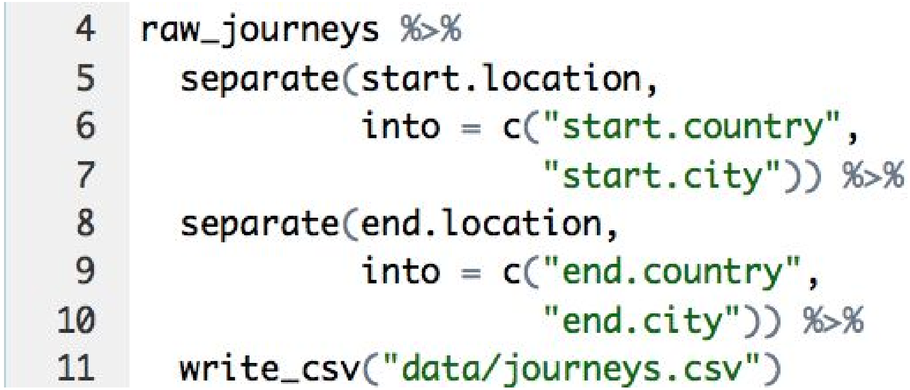
```r
foo_foo <- little_bunny()
bop_on(scoop_up(hop_through(foo_foo,
   3          2            1         forest),
                           field_mouse),
       head)
```

```r
foo_foo %>%
  hop_through(forest)
```

```r
hop_through(foo_foo, forest)
```

Chaining together multiple operations

Multiple operations can easily be piped together, as in the code below which exports a tidied version of our data to data/journeys.csv



```r
raw_journeys %>%
  separate(start.location,
           into = c("start.country",
                    "start.city")) %>%
  separate(end.location,
           into = c("end.country",
                    "end.city")) %>%
  write_csv("data/journeys.csv")
```

We're going to use the data/cheats_journeys.csv file instead as that's been wrangled more by the data-raw/cheats_data-clean.R script

Interactive maps with Leaflet

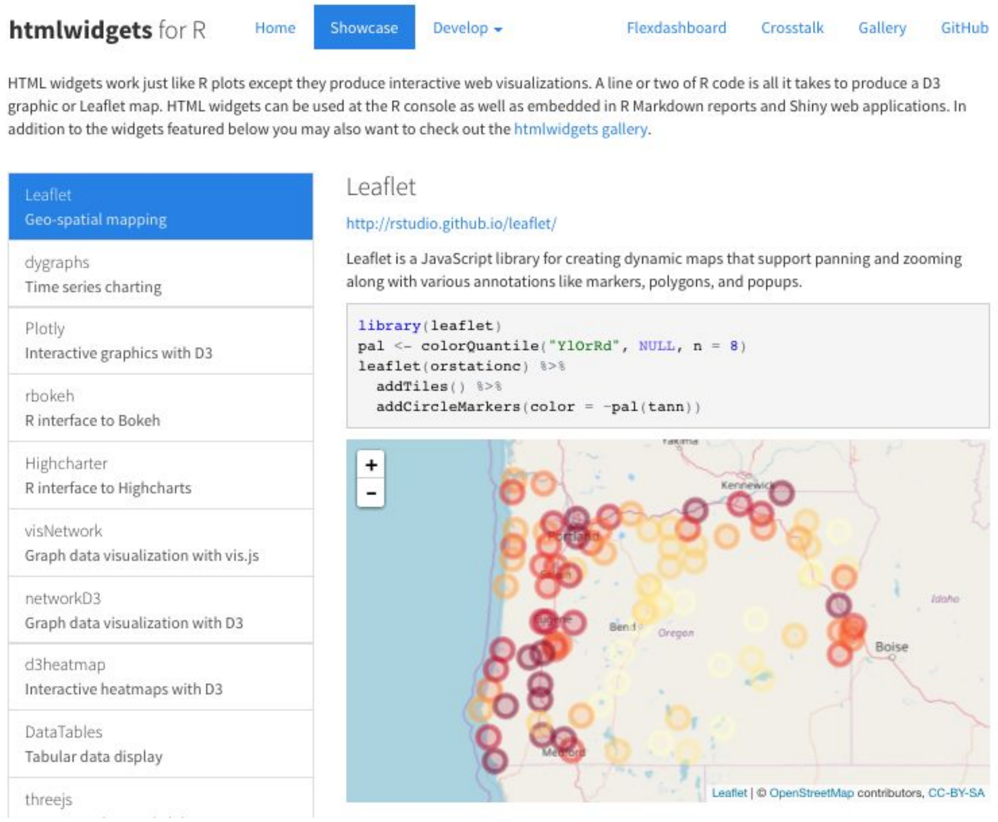
leaflet is a htmlwidget library that allows us to create interactive maps easily in R.

htmlwidgets is a framework for building bindings between JavaScript libraries and R.

htmlwidgets is a framework for building bindings between JavaScript libraries and R.

htmlwidgets allows develops to build their own R packages that provide end-users with access to these high-level JavaScript libraries with R code.

htmlwidgets.org



htmlwidgets for R Home Showcase Develop Flexdashboard Crosstalk Gallery GitHub

HTML widgets work just like R plots except they produce interactive web visualizations. A line or two of R code is all it takes to produce a D3 graphic or Leaflet map. HTML widgets can be used at the R console as well as embedded in R Markdown reports and Shiny web applications. In addition to the widgets featured below you may also want to check out the htmlwidgets gallery.

- Leaflet
  Geo-spatial mapping
- dygraphs
  Time series charting
- Plotly
  Interactive graphics with D3
- rbokeh
  R interface to Bokeh
- Highcharter
  R interface to Highcharts
- visNetwork
  Graph data visualization with vis.js
- networkD3
  Graph data visualization with D3
- d3heatmap
  Interactive heatmaps with D3
- DataTables
  Tabular data display
- threejs

Leaflet

http://rstudio.github.io/leaflet/

Leaflet is a JavaScript library for creating dynamic maps that support panning and zooming along with various annotations like markers, polygons, and popups.

```
library(leaflet)
pal <- colorQuantile("YlOrRd", NULL, n = 8)
leaflet(orstationc) %>%
  addTiles() %>%
  addCircleMarkers(color = ~pal(tann))
```

Interactive maps with Leaflet

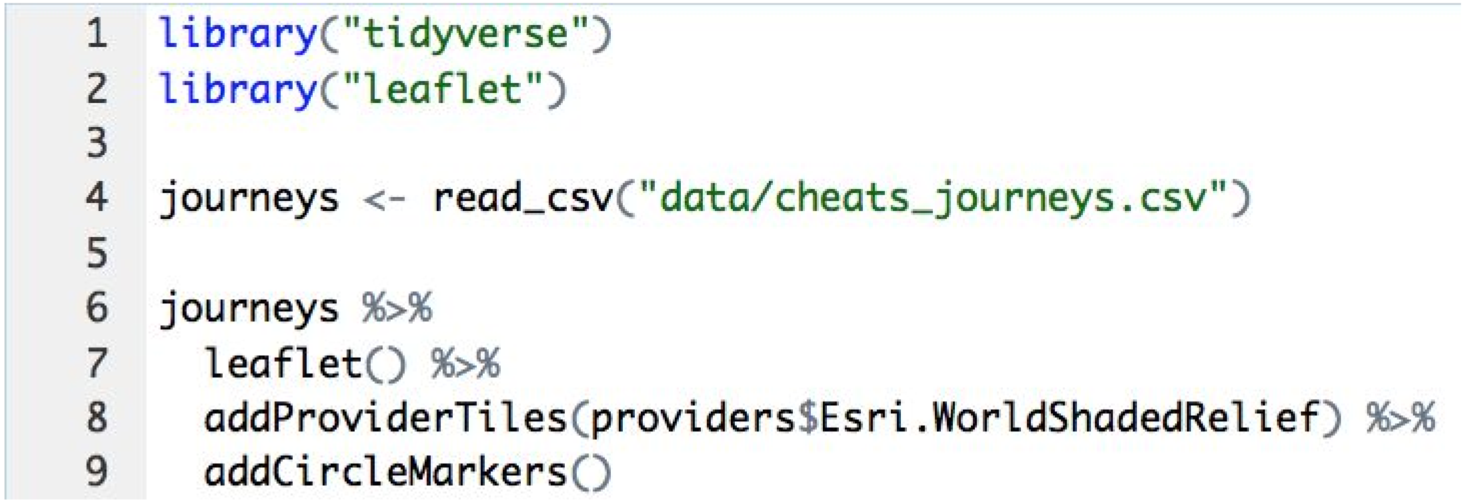
htmlwidgets is a framework for building bindings between JavaScript libraries and R.

htmlwidgets allows develops to build their own R packages that provide end-users with access to these high-level JavaScript libraries with R code.

The leaflet library is one of the flexible choices for making interactive maps.

Scatter geo plot

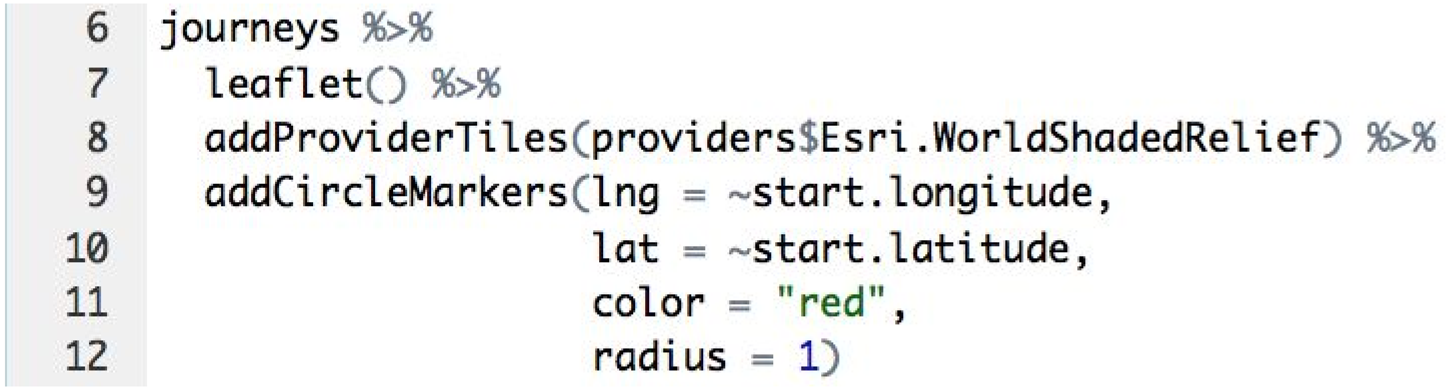
Scatter geo plots are useful for understanding the geographic distribution of locations, for instance the start and end locations of journeys in our dataset.



```r
library("tidyverse")
library("leaflet")

journeys <- read_csv("data/cheats_journeys.csv")

journeys %>%
  leaflet() %>%
  addProviderTiles(providers$Esri.WorldShadedRelief) %>%
  addCircleMarkers()
```

Scatter geo plot

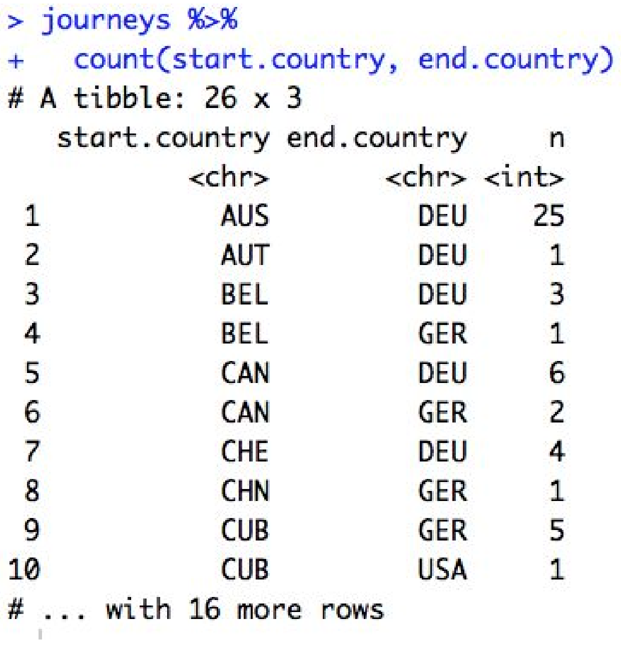
To access columns within the data argument of a leaflet map, use ~column.name



```r
journeys %>%
  leaflet() %>%
  addProviderTiles(providers$Esri.WorldShadedRelief) %>%
  addCircleMarkers(lng = ~start.longitude,
                   lat = ~start.latitude,
                   color = "red",
                   radius = 1)
```

Calculating frequencies of international journeys (I)

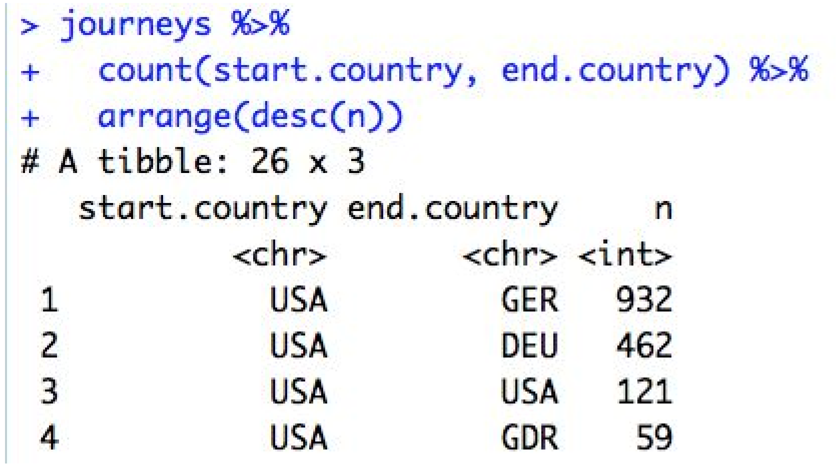
To rapidly calculate the frequency of different international journeys we can use the **count** verb



```
> journeys %>%
+   count(start.country, end.country)
# A tibble: 26 x 3
   start.country end.country     n
           <chr>       <chr> <int>
 1           AUS         DEU    25
 2           AUT         DEU     1
 3           BEL         DEU     3
 4           BEL         GER     1
 5           CAN         DEU     6
 6           CAN         GER     2
 7           CHE         DEU     4
 8           CHN         GER     1
 9           CUB         GER     5
10           CUB         USA     1
# ... with 16 more rows
```

Calculating frequencies of international journeys (II)

The **arrange** verb is used for re-ordering tibbles (data.frames)

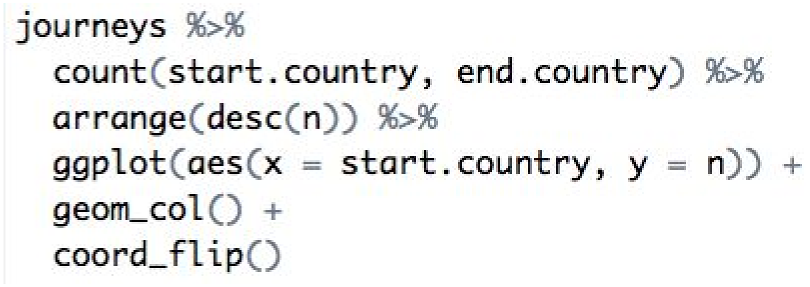


```
> journeys %>%
+   count(start.country, end.country) %>%
+   arrange(desc(n))
# A tibble: 26 x 3
  start.country end.country     n
          <chr>       <chr> <int>
1           USA         GER   932
2           USA         DEU   462
3           USA         USA   121
4           USA         GDR    59
```

Static charts with ggplot2 (I)

The ggplot2 library is part of the tidyverse and provides a consistent and very powerful approach to creating static charts with R.

It’s important to note that ggplot2 is one of the oldest parts of the tidyverse, and was built before the %>% operator was conceived.



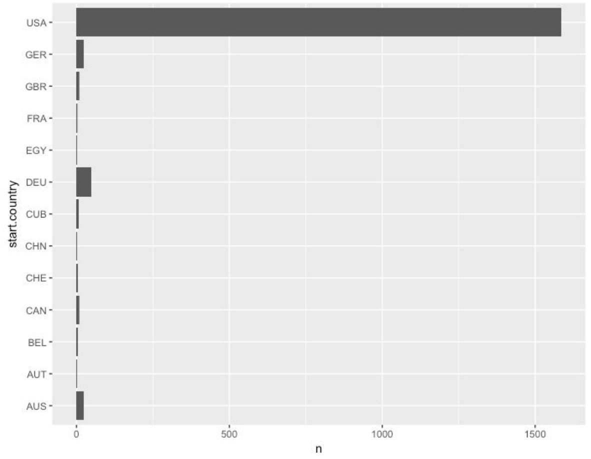
```r
journeys %>%
  count(start.country, end.country) %>%
  arrange(desc(n)) %>%
  ggplot(aes(x = start.country, y = n)) +
  geom_col() +
  coord_flip()
```

Static charts with ggplot2 (II)

We need to do a little work on our data before we can chart it:

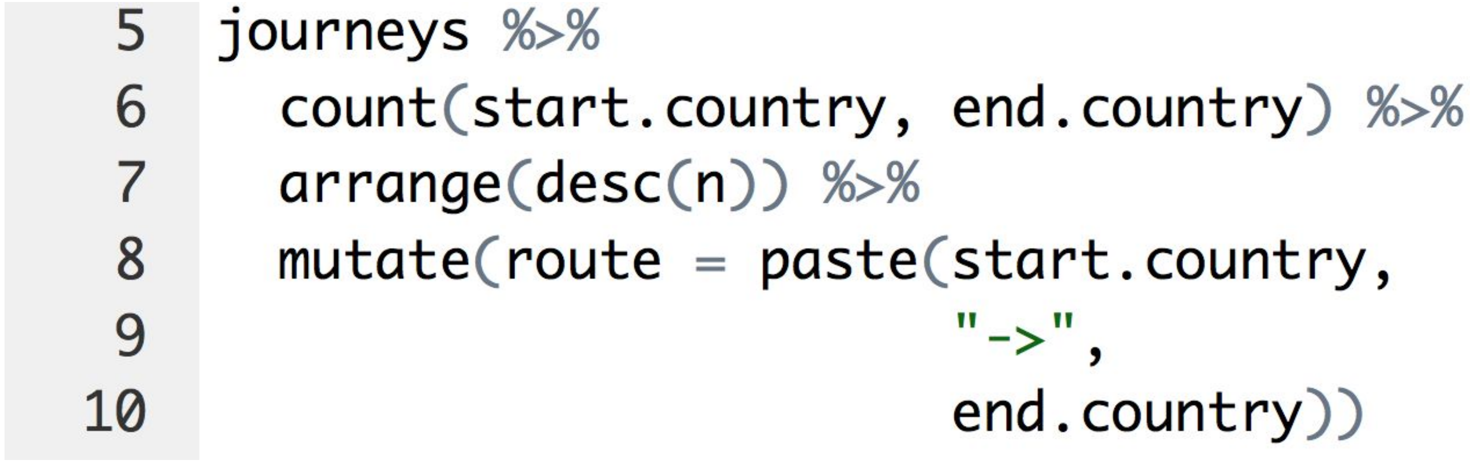
- We need to combine the start.country and end.country into a new column
- We need to order the columns from biggest to smallest

```
journeys %>%
  count(start.country, end.country) %>%
  arrange(desc(n)) %>%
  ggplot(aes(x = start.country, y = n)) +
  geom_col() +
  coord_flip()
```



Adding new columns with mutate

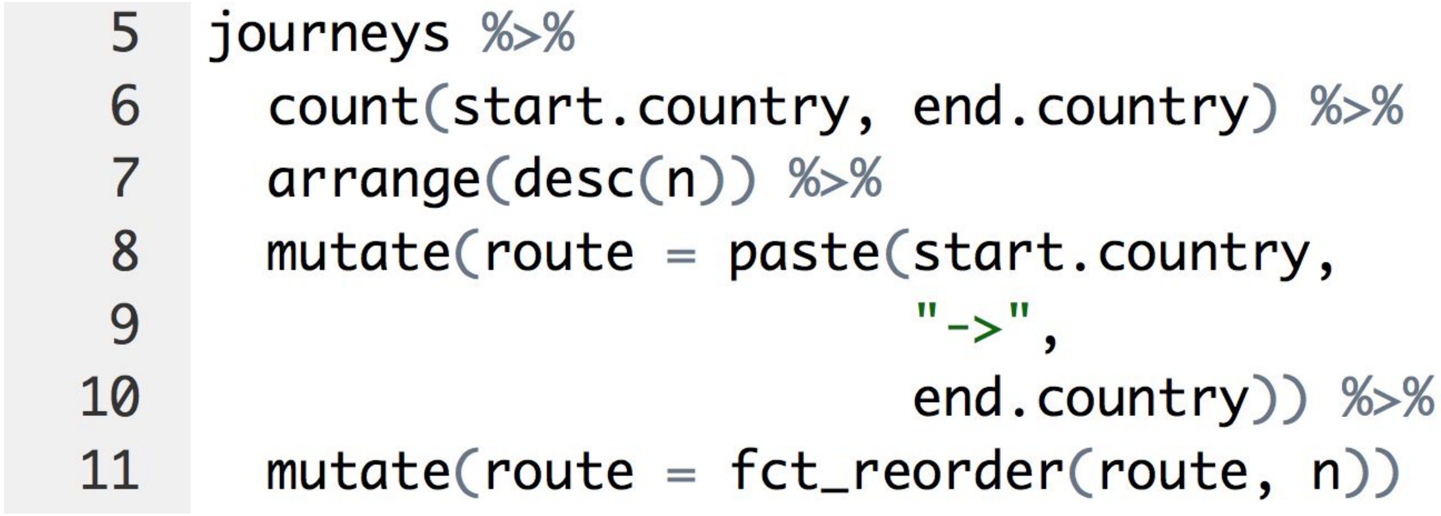
The **mutate** verb allows existing columns to be modified and for new columns to be added easily



```
journeys %>%
  count(start.country, end.country) %>%
  arrange(desc(n)) %>%
  mutate(route = paste(start.country,
                       "->",
                       end.country))
```

Ordering factors with forcats

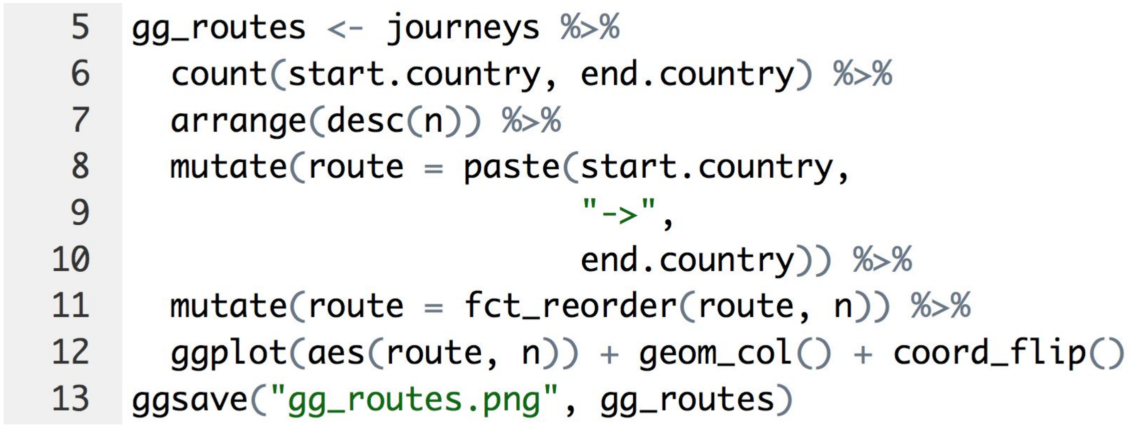
Factors are used by ggplot2 to decide the order in which categorical variables should be displayed. The forcats library provides useful tools for modifying factors



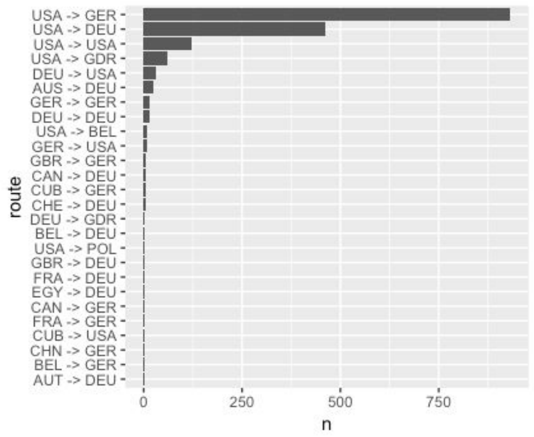
```r
journeys %>%
  count(start.country, end.country) %>%
  arrange(desc(n)) %>%
  mutate(route = paste(start.country,
                       "->",
                       end.country)) %>%
  mutate(route = fct_reorder(route, n))
```

Static charts with ggplot2 (III)

ggplot2 is incredibly useful for creating static plots, and they can be exported as images easily using **ggsave**



```r
gg_routes <- journeys %>%
  count(start.country, end.country) %>%
  arrange(desc(n)) %>%
  mutate(route = paste(start.country,
                       "->",
                       end.country)) %>%
  mutate(route = fct_reorder(route, n)) %>%
  ggplot(aes(route, n)) + geom_col() + coord_flip()
ggsave("gg_routes.png", gg_routes)
```

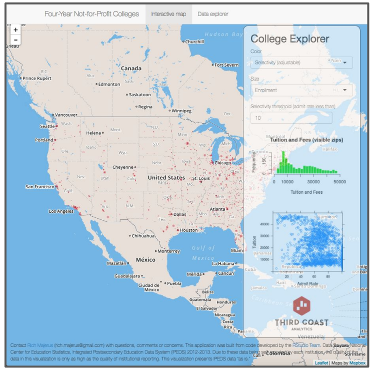


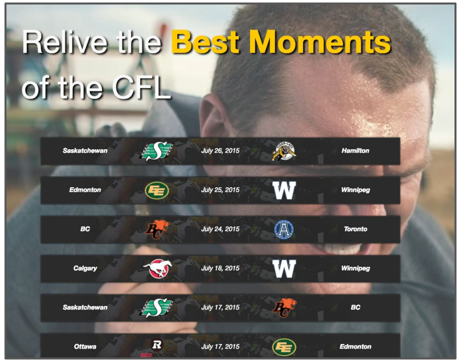
RMarkdown is a simple way to build reports and presentations that include R code.

RMarkdown can output html reports and presentations that include htmlwidgets.

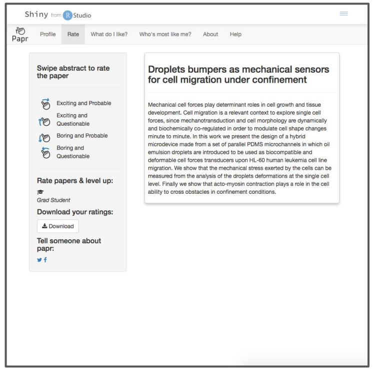
RMarkdown reports can easily and freely be published to RPubs.com

Shiny is a framework for creating interactive web applications using R.




Relive the Best Moments of the CFL

Saskatchewan	July 26, 2015	Hamilton
Edmonton	July 25, 2015	Winnipeg
BC	July 24, 2015	Toronto
Calgary	July 18, 2015	Winnipeg
Saskatchewan	July 17, 2015	BC
Ottawa	July 17, 2015	Edmonton



Swipe abstract to rate the paper

Exciting and Probable
Exciting and Questionable
Boring and Probable
Boring and Questionable

Rate papers & level up:

Grad Student

Download your ratings:

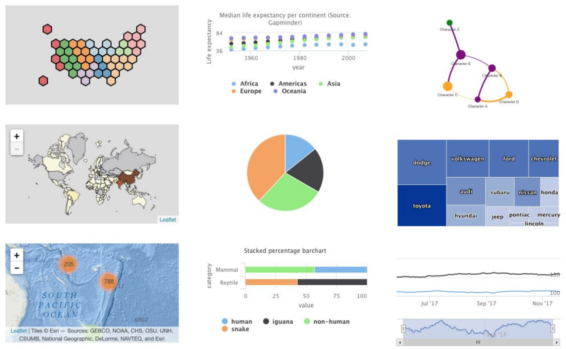
Download

Tell someone about papr:

Droplets bumpers as mechanical sensors for cell migration under confinement

Mechanical cell forces play determinant roles in cell growth and tissue development. Cell migration is a relevant context to explore single cell forces, since mechanotransduction and cell morphology are dynamically and biochemically co-regulated in order to modulate cell shape changes minute to minute. In this work we present the design of a hybrid microdevice made from a set of parallel PDMS microchannels in which oil emulsion droplets are introduced to be used as biocompatible and deformable cell forces transducers upon HL-60 human leukemia cell line migration. We show that the mechanical stress exerted by the cells can be measured from the analysis of the droplets deformations at the single cell level. Finally we show that acto-myosin contraction plays a role in the cell ability to cross obstacles in confinement conditions.

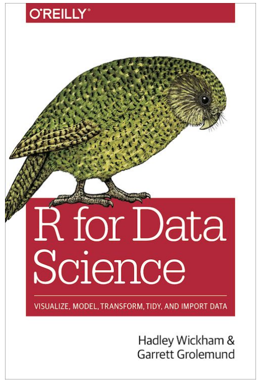
OxShef



- Which chart should I use?
  oxshef.github.io/oxshef_charts
- Where can I host visualisations?
  oxshef.github.io/oxshef_shiny
  oxshef.github.io/oxshef_dash
- Which publishers support embedding?
  oxshef.github.io/oxshef_publishers

DataCamp

http://r4ds.had.co.nz/

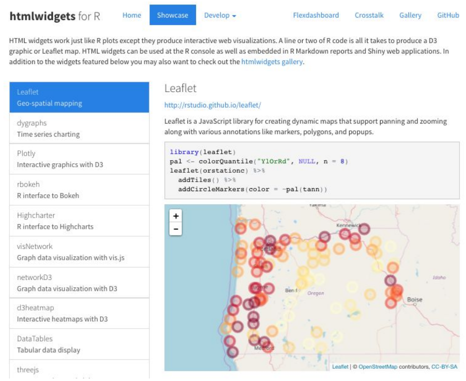

O'REILLY

R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

Hadley Wickham & Garrett Grolemund

htmlwidgets.org



HTML widgets work just like R plots except they produce interactive web visualizations. A line or two of R code is all it takes to produce a D3 graphic or Leaflet map. HTML widgets can be used at the R console as well as embedded in R Markdown reports and Shiny web applications. In addition to the widgets featured below you may also want to check out the htmlwidgets gallery.

Leaflet
Geo-spatial mapping

dygraphs
Time series charting

Plotly
Interactive graphics with D3

rbokeh
R interface to Bokeh

Highcharter
R interface to Highcharts

visNetwork
Graph data visualization with vis.js

networkD3
Graph data visualization with D3

d3heatmap
Interactive heatmaps with D3

DataTables
Tabular data display

threejs

Leaflet

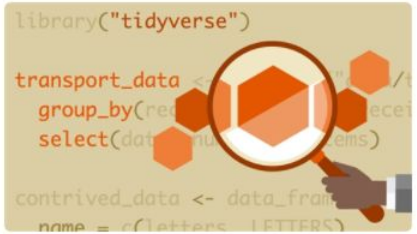
http://rstudio.github.io/leaflet/

Leaflet is a JavaScript library for creating dynamic maps that support panning and zooming along with various annotations like markers, polygons, and popups.

```
library(leaflet)
pal <- colorQuantile("YlOrRd", NULL, n = 8)
leaflet(orstationc) %>%
  addTiles() %>%
  addCircleMarkers(color = ~pal(tann))
```

lynda.com

LEARNING



Learning the R Tidyverse

Learn to integrate the tidyverse into your R workflow and get new tools for importing, filtering, visualizing, and modeling research and statistical data.

3h 44m Intermediate Views: 2,826 1 week ago

R: Interactive Visualizations with htmlwidgets

Learn how to rapidly create rich, interactive data visualizations with R and htmlwidgets—packages that connect R to popular JavaScript libraries like Plotly, Leaflet, and DT.

5h 25m Intermediate Views: 3,490 Oct 4, 2017

Creating Interactive Presentations with Shiny and R

Make the results of big data analysis more compelling and clear. Learn how to create interactive presentations and dashboards with RStudio and Shiny.

1h 53m Intermediate Views: 58,734 Apr 27, 2016