RStudio

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212

Email: info@rstudio.com
Web: http://www.rstudio.com

Expert Data Wrangling with R

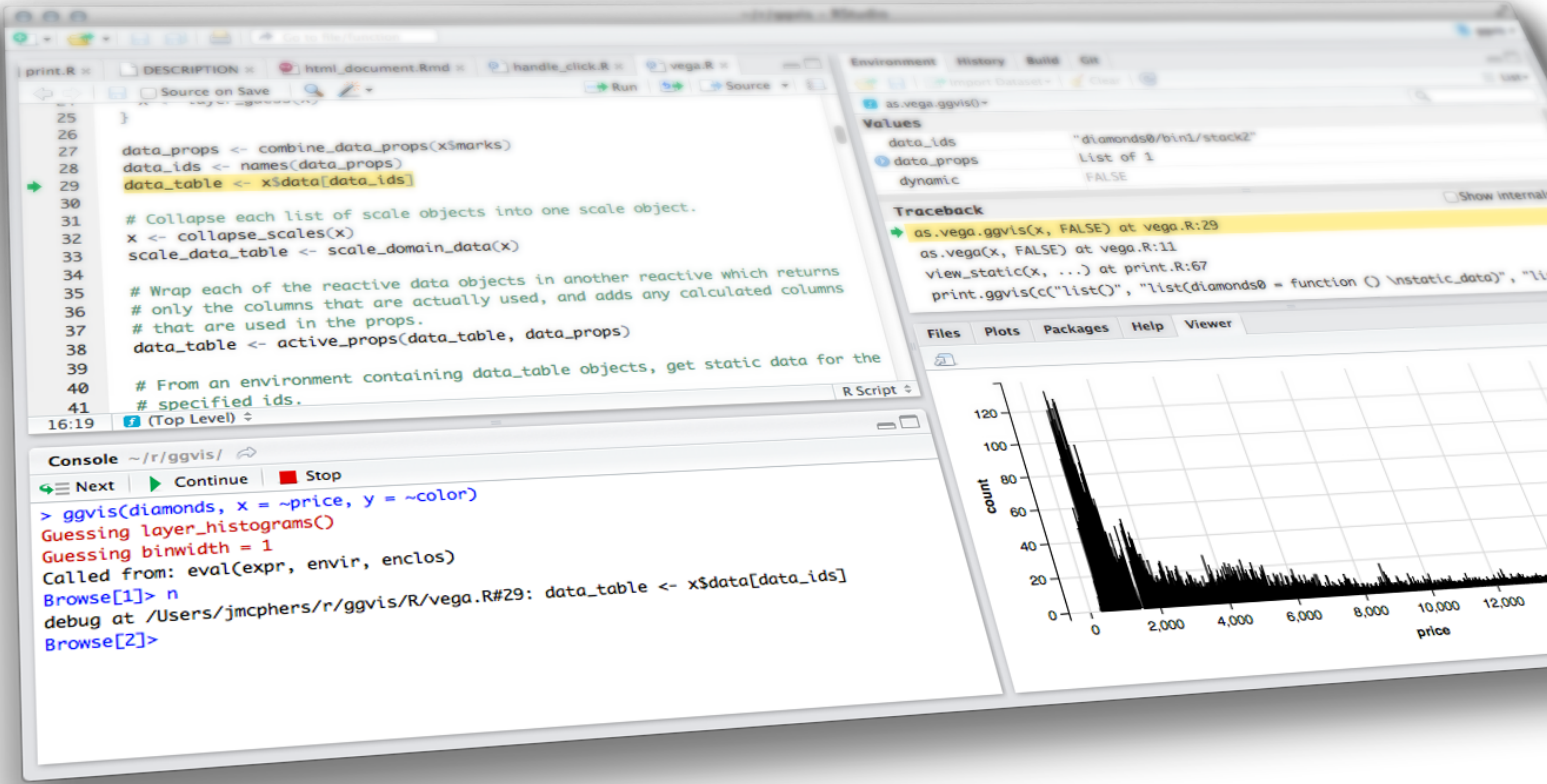
How to make the most of your data

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

Garrett Grolemund

Data Scientist and Master Instructor
November 2014
Email: garrett@rstudio.com

Introduction


print.R
DESCRIPTION
html_document.Rmd
handle_click.R
vega.R
Source on Save
Run
Source
}
data_props <- combine_data_props(x$marks)
data_ids <- names(data_props)
data_table <- x$data[data_ids]
# Collapse each list of scale objects into one scale object.
x <- collapse_scales(x)
scale_data_table <- scale_domain_data(x)
# Wrap each of the reactive data objects in another reactive which returns
# only the columns that are actually used, and adds any calculated columns
# that are used in the props.
data_table <- active_props(data_table, data_props)
# From an environment containing data_table objects, get static data for the
# specified ids.
R Script
16:19
(Top Level)
Console ~/r/ggvis/
Next
Continue
Stop
> ggvis(diamonds, x = ~price, y = ~color)
Guessing layer_histograms()
Guessing binwidth = 1
Called from: eval(expr, envir, enclos)
Browse[1]> n
debug at /Users/jmcphers/r/ggvis/R/vega.R#29: data_table <- x$data[data_ids]
Browse[2]>
Environment
History
Build
Git
as.vega.ggvis()
Values
data_ids
"diamonds0/bin1/stack2"
data_props
List of 1
dynamic
FALSE
Traceback
Show internals
as.vega.ggvis(x, FALSE) at vega.R:29
as.vega(x, FALSE) at vega.R:11
view_static(x, ...) at print.R:67
Files
Plots
Packages
Help
Viewer
120
100
80
60
40
20
0
count
0
2,000
4,000
6,000
8,000
10,000
12,000
price

RStudio

Wrangling
Munging
Cleaning
Carpentry
Manipulation
Transformation

50-80%

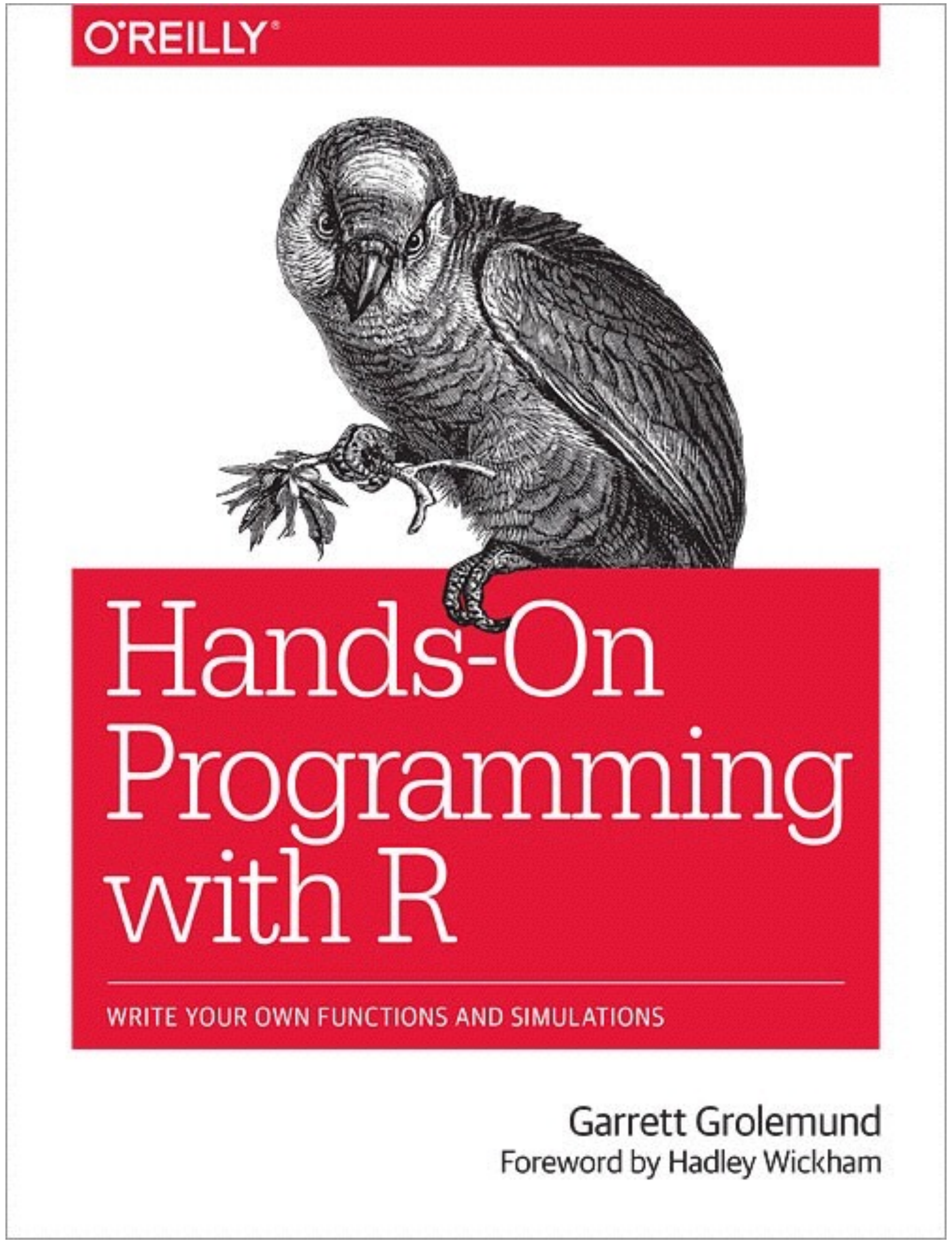
of your time?

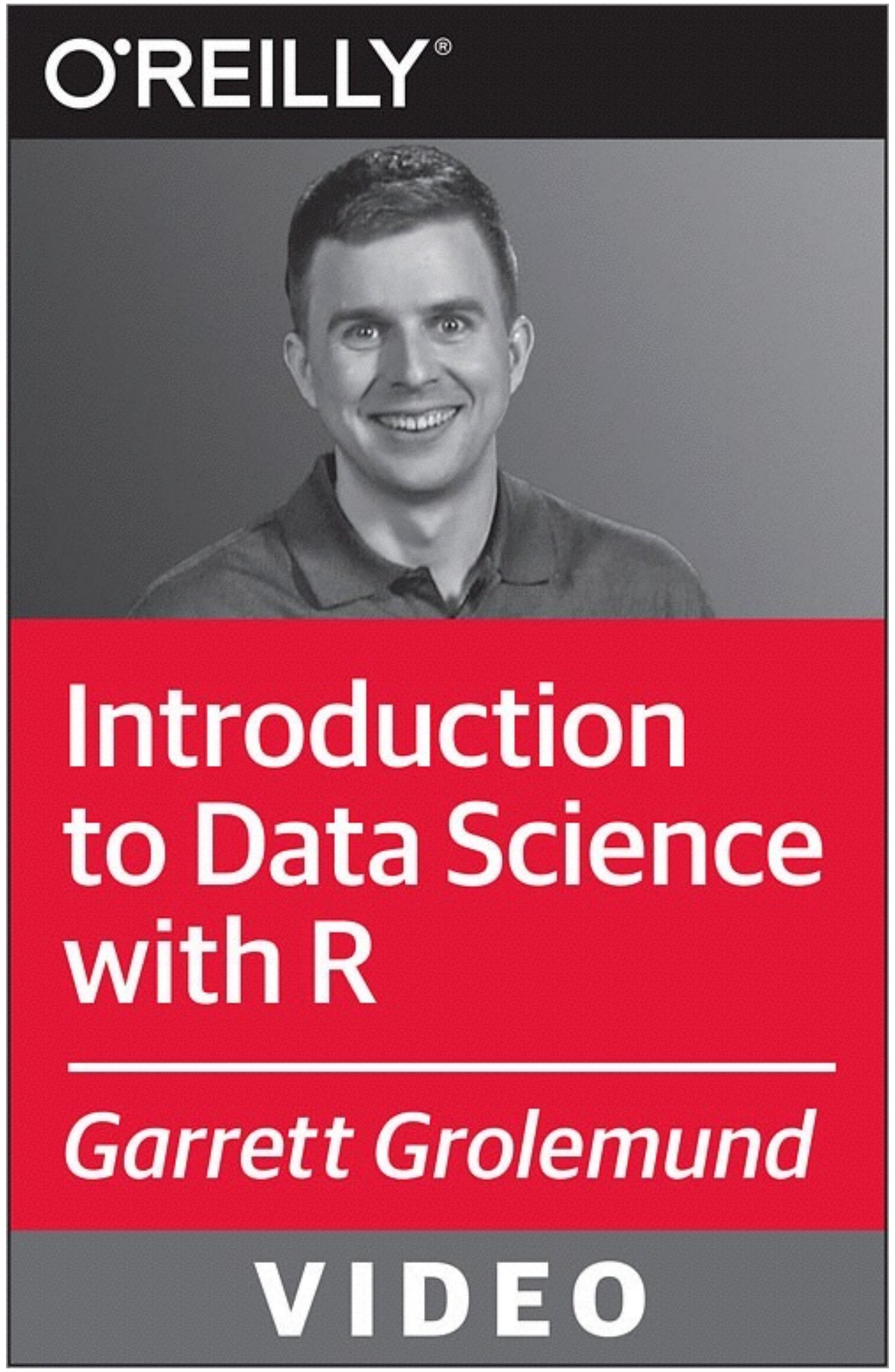
HELLO
my name is

Garrett

@StatGarrett

RStudio®


O'REILLY
Hands-On Programming with R
WRITE YOUR OWN FUNCTIONS AND SIMULATIONS
Garrett Grolemund
Foreword by Hadley Wickham


O'REILLY
Introduction to Data Science with R
Garrett Grolemund
VIDEO

Data Wrangling

Data Wrangling

The practical task of transforming the *layout*, *substance*, and *display* of your data.

Data Manipulation

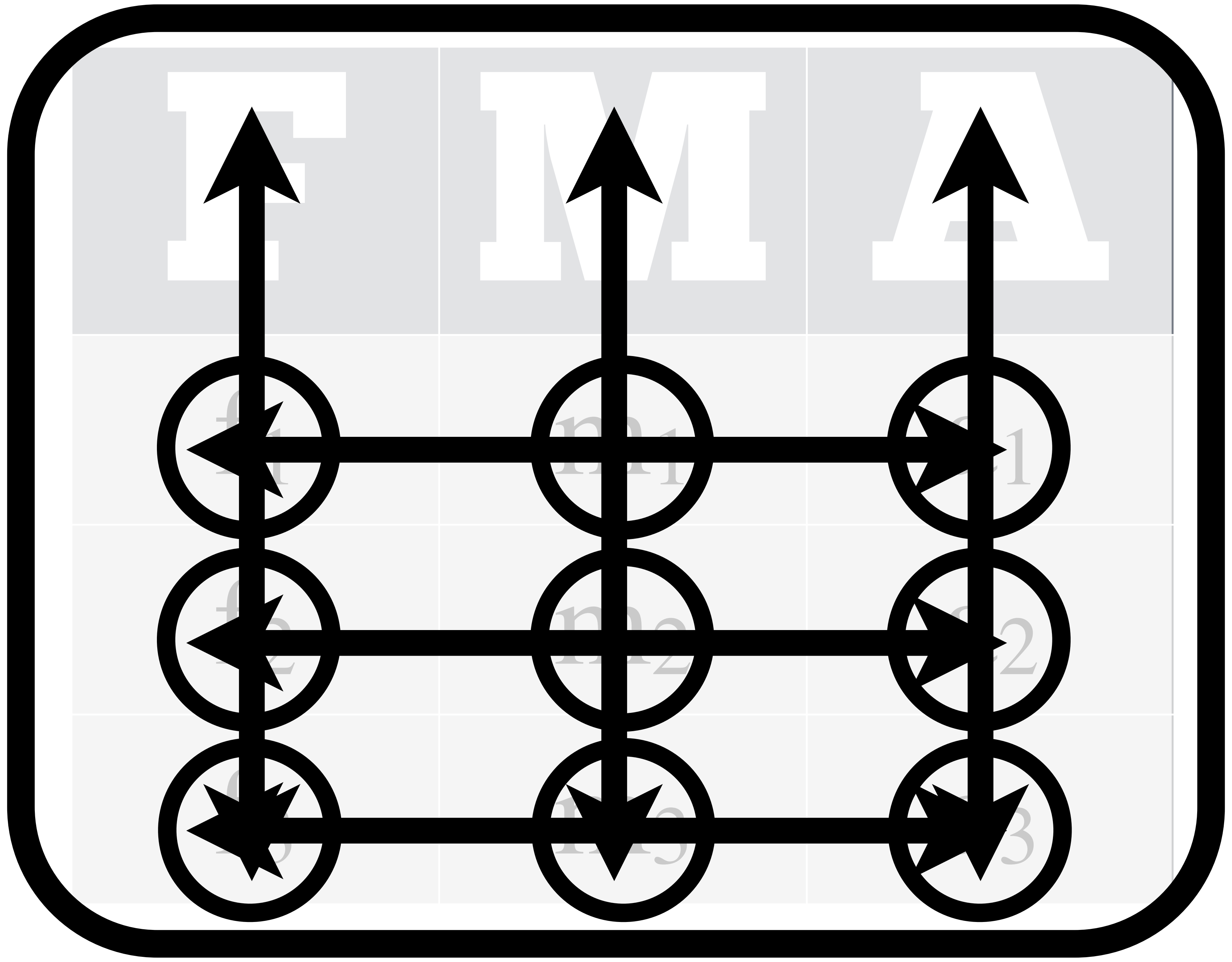
Changing the variables, values, and
units of analysis contained in the data set.

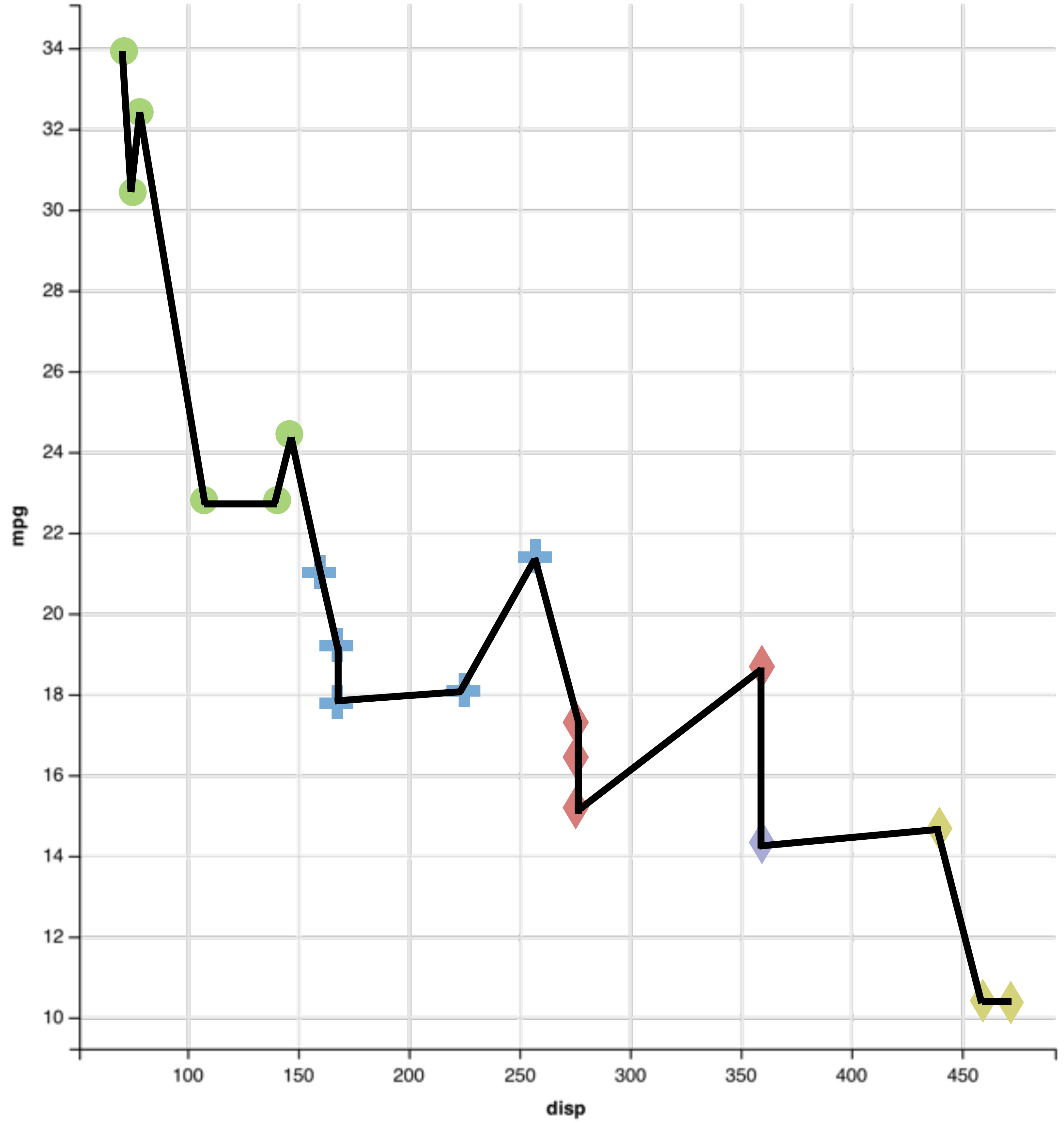
Data Tidying

Changing the layout of tabular data to make it
suitable for a particular piece of software (R).

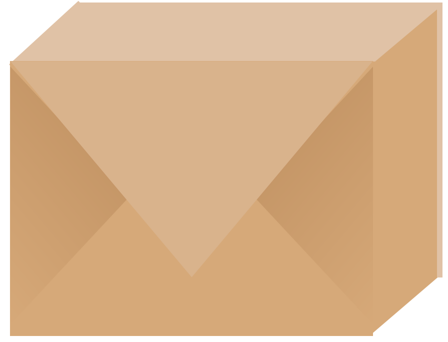
Data Visualization

Transforming the data to a visual format that
reveals visual patterns.

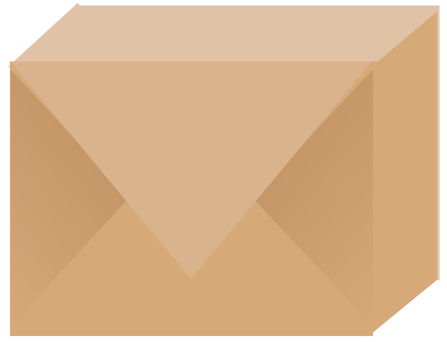

F
M
A


mpg
disp
34
32
30
28
26
24
22
20
18
16
14
12
10
100
150
200
250
300
350
400
450

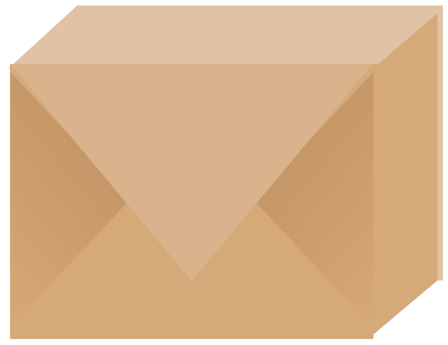
Three packages to help you work with the structure of data.



dplyr

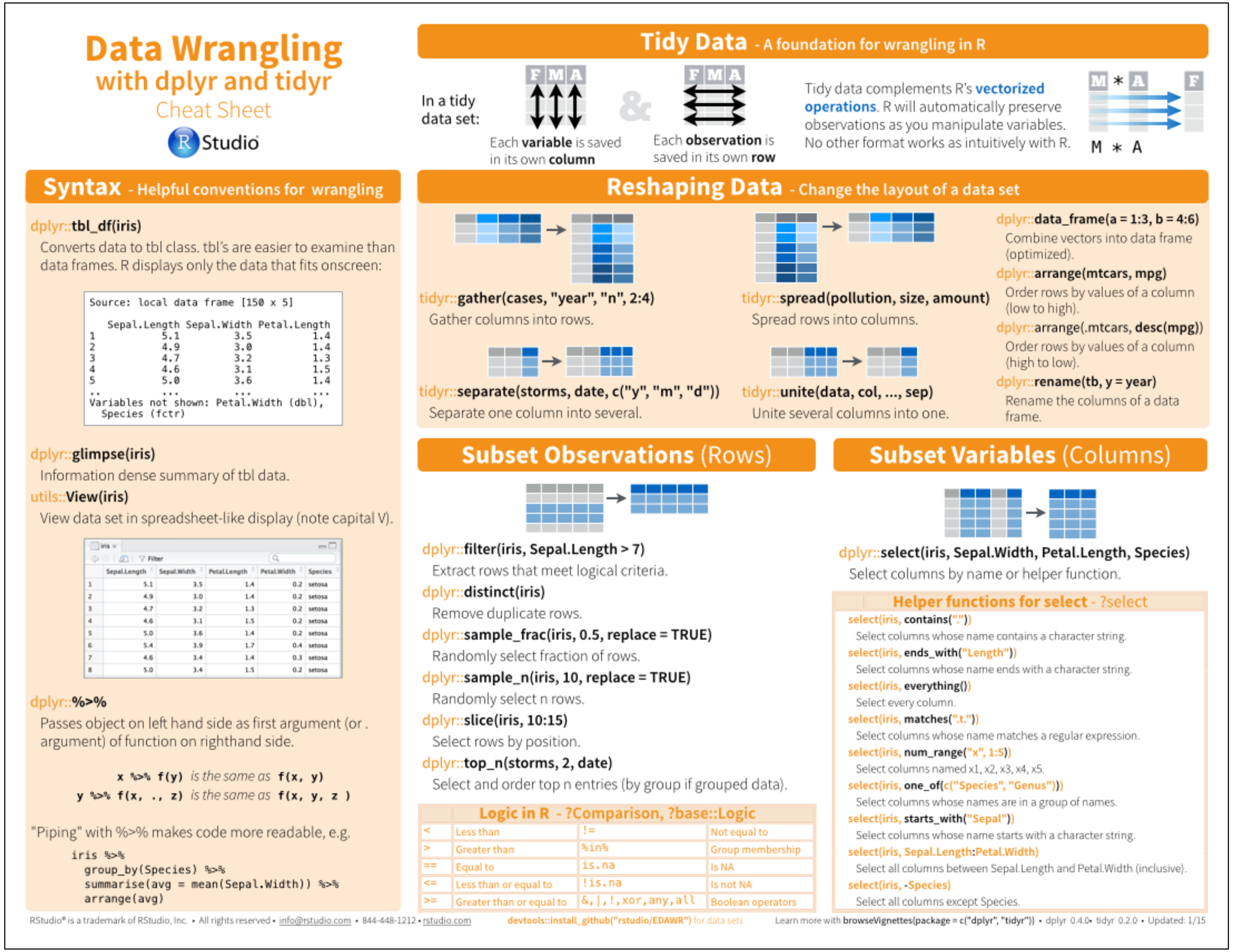


tidyr



ggvis

RStudio



Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column** & Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

M * A

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

  Sepal.Length Sepal.Width Petal.Length
1          5.1         3.5          1.4
2          4.9         3.0          1.4
3          4.7         3.2          1.3
4          4.6         3.1          1.5
5          5.0         3.6          1.4
..         ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` *is the same as* `f(x, y)`

`y %>% f(x, ., z)` *is the same as* `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(.mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, **Sepal.Length:Petal.Width**)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, **-Species**)
Select all columns except Species.

 devtools::install_github("rstudio/EDAWR") for data sets. Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

http://www.rstudio.com/resources/cheatsheets/

R Packages with example data sets

- EDAWR
- nycflights13

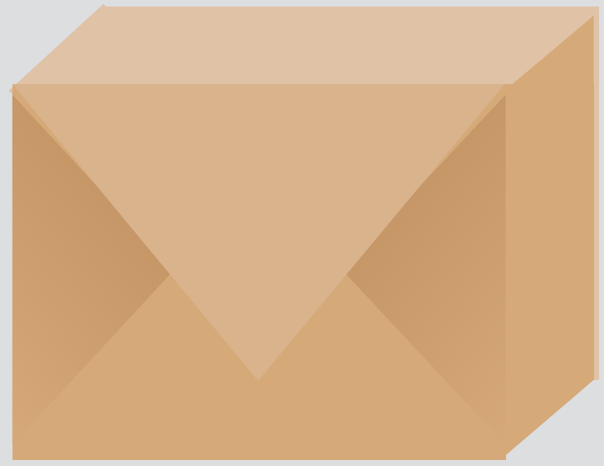
Case Study

Explore the spread of TB from 1995 to 2013 in 100 countries.

country	year	cases	population	rate
Afghanistan	1997	128	19021226	0.07
Afghanistan	1998	1778	19496836	0.91
Afghanistan	1999	745	19987071	0.37
Afghanistan	2000	2666	20595360	1.29
Afghanistan	2001	4639	21347782	2.17
Afghanistan	2002	6509	22202806	2.93
Afghanistan	2003	6528	23116142	2.82

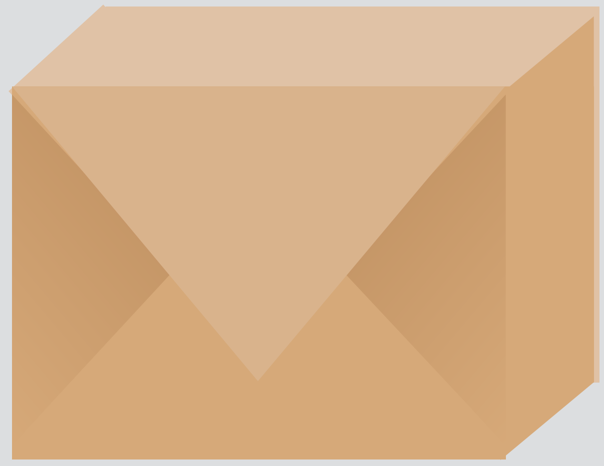
Your Turn

Open R. Then download the packages that we will use today by running the following code.



EDAWR

```
install.packages("devtools")
devtools::install_github("rstudio/EDAWR")
```

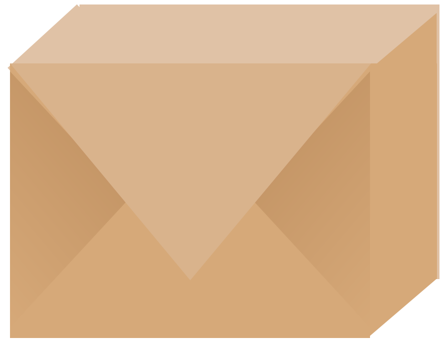


dplyr, tidyr, ggvis, nycflights13

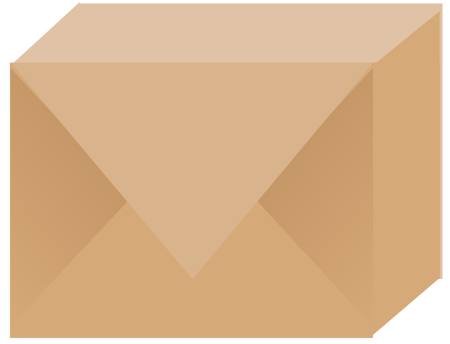
```
install.packages(c("dplyr", "tidyr",
   "ggvis", "nycflights13"))
```

Two new conventions

Install packages



```r
library(dplyr)
```



```r
library(nycflights13)
```

nycflights13

Arrival, departure, meterological, etc. data for 300,000+ flights that flew in and out of New York City in 2013.

tbl’s

tbl's

Just like data frames, but play better with the console window.

```
Source: local data frame [336,776 x 16]

   year month day dep_time dep_delay arr_time
1  2013     1   1      517         2      830
2  2013     1   1      533         4      850
3  2013     1   1      542         2      923
4  2013     1   1      544        -1     1004
5  2013     1   1      554        -6      812
6  2013     1   1      554        -4      740
7  2013     1   1      555        -5      913
8  2013     1   1      557        -3      709
9  2013     1   1      557        -3      838
10 2013     1   1      558        -2      753
..  ...   ... ...      ...       ...      ...
Variables not shown: arr_delay (dbl), carrier
  (chr), tailnum (chr), flight (int), origin
  (chr), dest (chr), air_time (dbl), distance
  (dbl), hour (dbl), minute (dbl)
```

tbl

```
1978  2013     1   3      848         -2
1979  2013     1   3      850         -5
1980  2013     1   3      850         21
1981  2013     1   3      851         24
1982  2013     1   3      851        111
1983  2013     1   3      851         -4
1986  2013     1   3      853         -5
1987  2013     1   3      854        144
1988  2013     1   3      855         -3
1989  2013     1   3      855         -5
1990  2013     1   3      855         -5
1991  2013     1   3      856        -12
1992  2013     1   3      856         36
1993  2013     1   3      857         -3
1994  2013     1   3      857         -3
1995  2013     1   3      857         -3
1996  2013     1   3      858         -2
1997  2013     1   3      858         -2
1998  2013     1   3      859         39
1999  2013     1   3      859         -1
2000  2013     1   3      900          3
 [ reached getOption("max.print") --
omitted 334776 rows ]
```

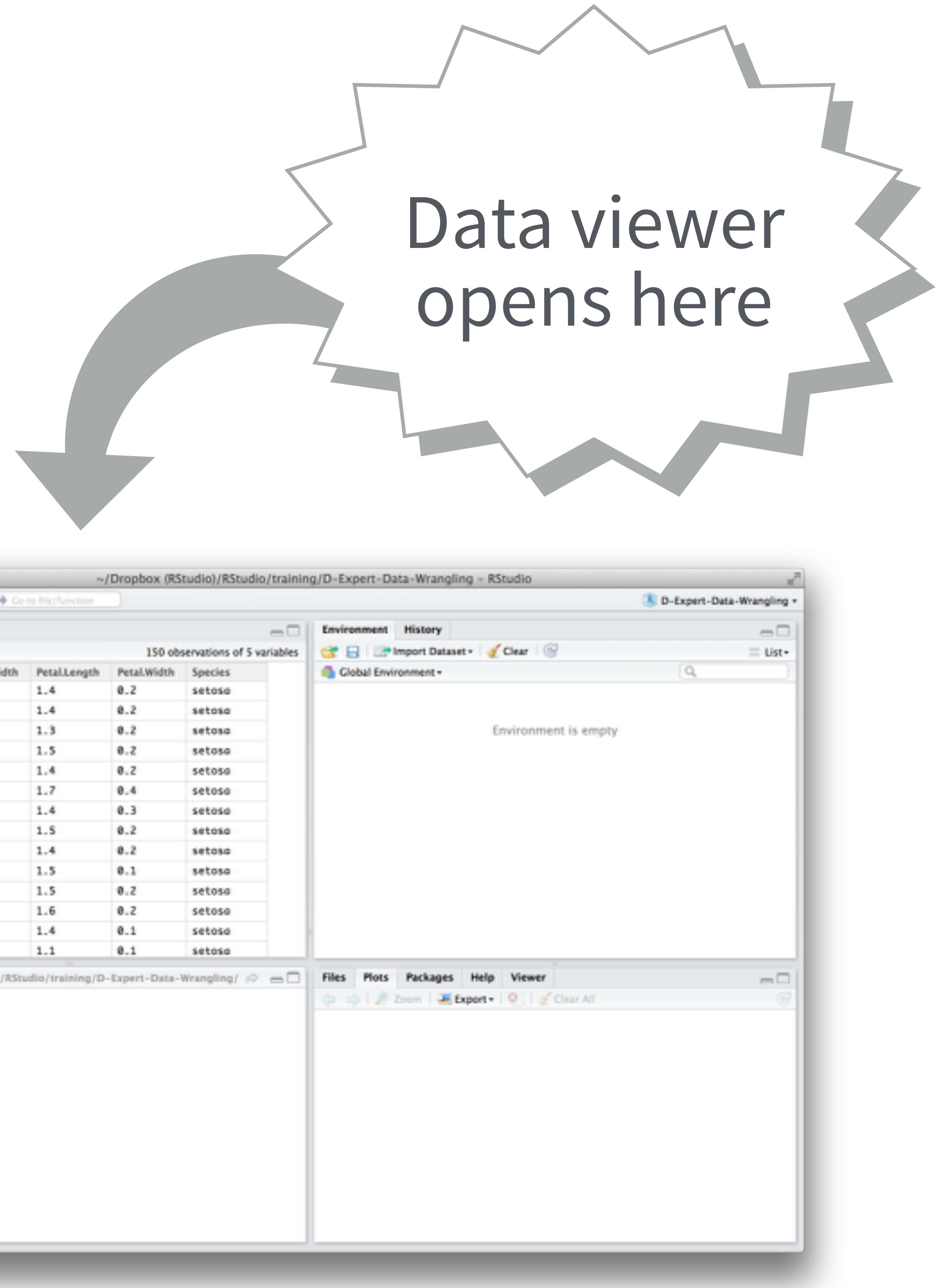
data.frame

```r
flights <- tbl_df(flights)
# Can undo with
# flights <- as.data.frame(flights)
```

View()

Examine any data set with the View() command (Capital V)

```
View(flights)
View(iris)
View(mtcars)
```



glimpse()

Examine values and structure at command line

```
glimpse(flights)
glimpse(iris)
glimpse(mtcars)
```

%>%

Ceci n'est pas une pipe.

The pipe operator %>%

```
dd <- flights$dep_delay
mean(dd, na.rm = TRUE)
dd %>% mean(na.rm = TRUE)
```

These do the same thing
Try it!


```
dd                mean( ______, na.rm = TRUE)
```

```
little_bunny.foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field.mouse) %>%
  bop_on(head)
```

Data Science*

* for Data Wranglers

"Music theory is simply an attempt to explain what musicians have been playing intuitively for thousands of years"

- Mike Iverson

Science

1 The best way to learn about the world is to **observe** it.

2 Nature behaves according to natural **laws** (e.g. $E = MC^2$, $F = MA$, …).

$$F = MA$$

variable variable variable

Variable - A quantity, quality, or property that you can measure.

$$F = MA$$

$$\begin{array}{lll} f_1 & m_1 & a_1 \\ f_2 & m_2 & a_2 \\ f_3 & m_3 & a_3 \end{array}$$

Variable - A quantity, quality, or property that you can measure.

Value - The state of a variable when you measure it.

(The value can change from measurement to measurement)

$$F = MA$$

$$f_1 = m_1 \cdot a_1$$
$$f_2 = m_2 \cdot a_2$$
$$f_3 = m_3 \cdot a_3$$

Variable - A quantity, quality, or property that you can measure.

Value - The state of a variable when you measure it.

Observation - The values of several variables measured under similar conditions.

RStudio

$$F = MA$$

$$f_1 = m_1 \cdot a_1$$
$$f_2 = m_2 \cdot a_2$$
$$f_3 = m_3 \cdot a_3$$

Structure of Natural Laws

Natural laws deal with **variables**,
but they operate on **values**
that appear in the same **observation**.

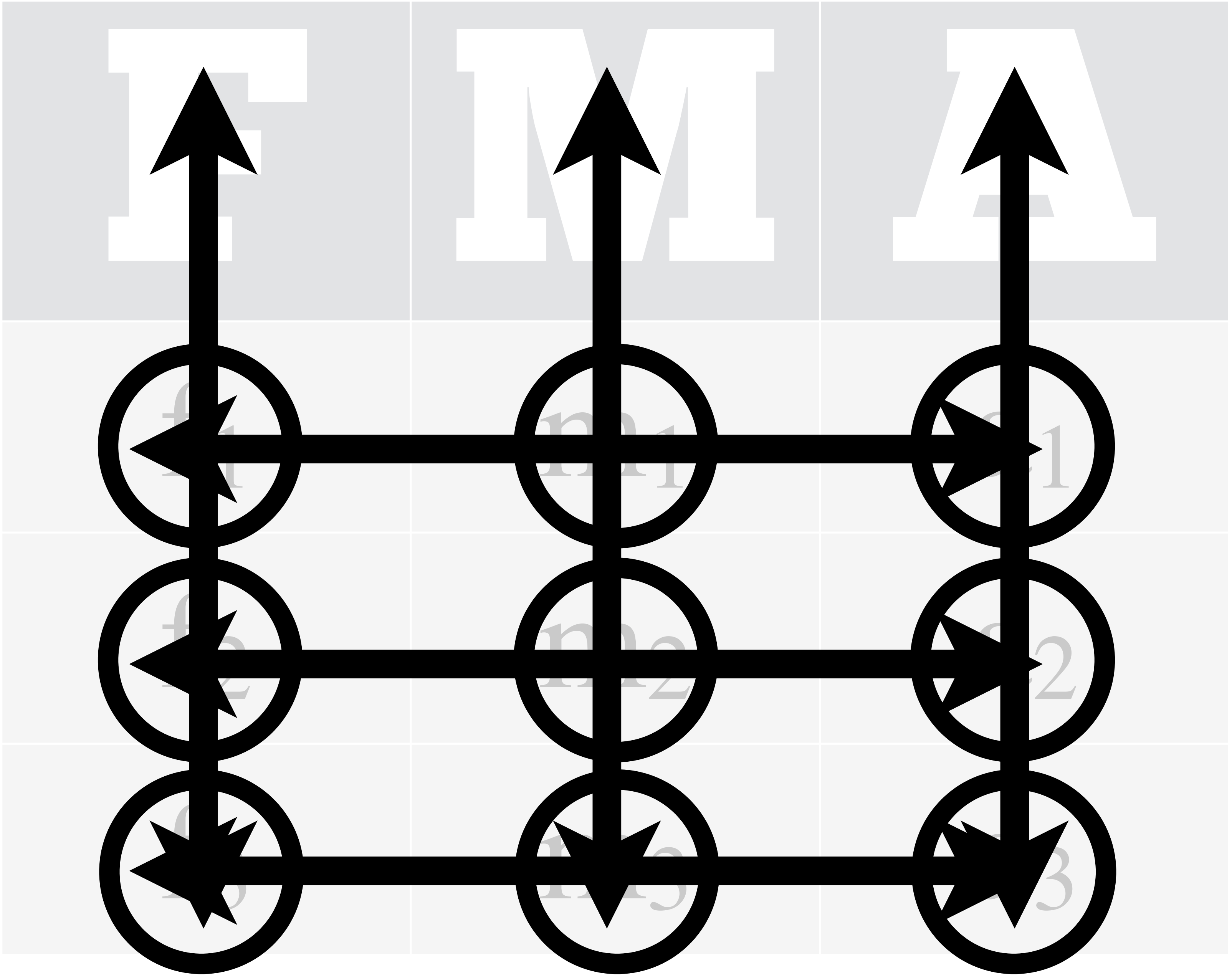
F	M	A
f_1	m_1	a_1
f_2	m_2	a_2
f_3	m_3	a_3

$$F = MA$$

$$= m_1 \cdot a_1$$

$$= m_2 \cdot a_2$$

$$= m_3 \cdot a_3$$



$$F = MA$$

$$f_1 = m_1 \cdot a_1$$

$$f_2 = m_2 \cdot a_2$$

$$f_3 = m_3 \cdot a_3$$

- Variables
- Values
- Observations

Data = Values associated with variables and observations

F	M	A
3.01	0.98	3.08
2.35	0.91	2.58
5.57	1.01	5.52

$$F = MA$$

$$f_1 = m_1 \cdot a_1$$

$$f_2 = m_2 \cdot a_2$$

$$f_3 = m_3 \cdot a_3$$

Laws appear as **patterns** in data.

RStudio

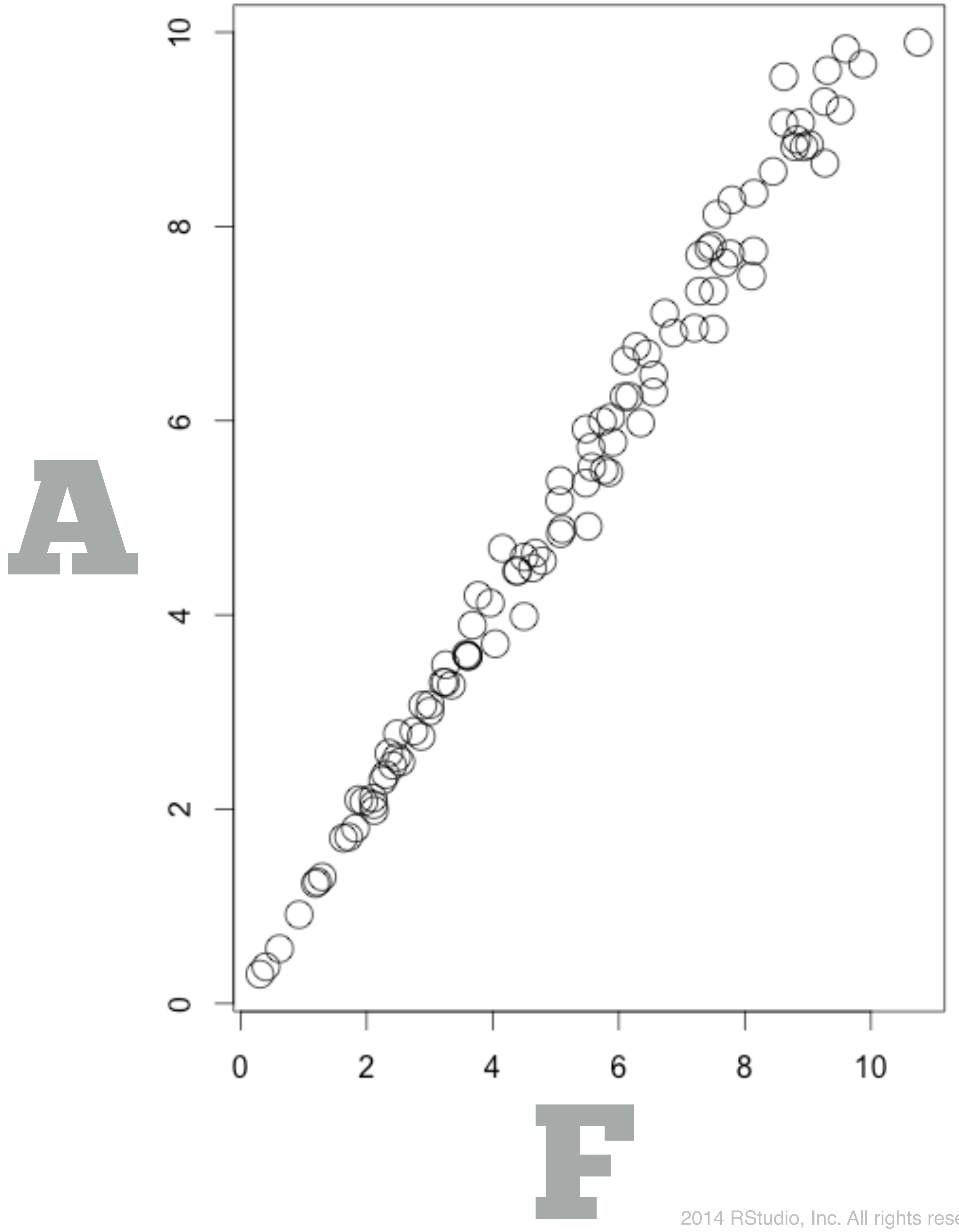
F	M	A
3.01	0.98	3.08
2.35	0.91	2.58
5.57	1.01	5.52
0.62	1.09	0.56
4.15	0.89	4.69
5.07	1.05	4.84
7.56	0.93	8.12

$$F = MA$$

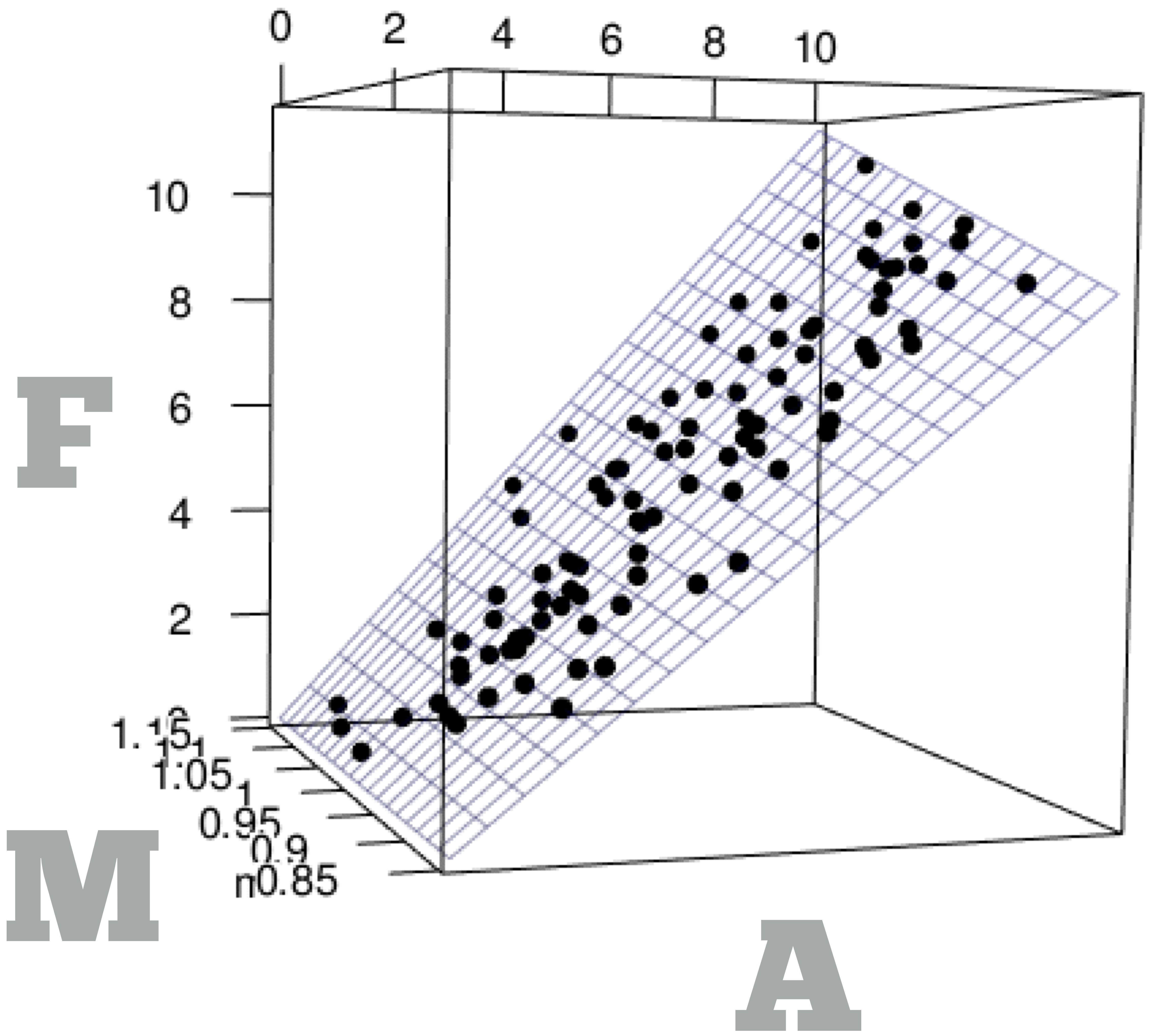
$$f_1 = m_1 \cdot a_1$$
$$f_2 = m_2 \cdot a_2$$
$$f_3 = m_3 \cdot a_3$$

Laws appear as **patterns** in data.

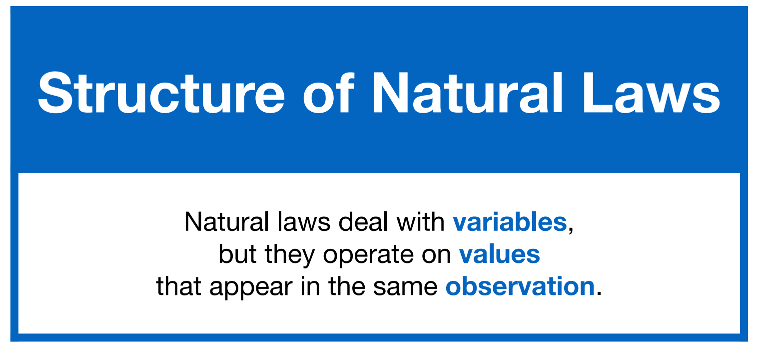
F	M	A
0.62	1.09	0.56
1.30	0.99	1.30
1.63	0.96	1.70
1.72	1.00	1.71
1.82	1.01	1.80
1.95	0.94	2.08
2.11	1.03	2.05



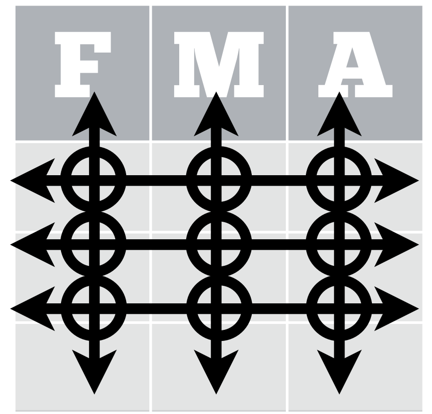
F	M	A
0.62	1.09	0.56
1.30	0.99	1.30
1.63	0.96	1.70
1.72	1.00	1.71
1.82	1.01	1.80
1.95	0.94	2.08
2.11	1.03	2.05



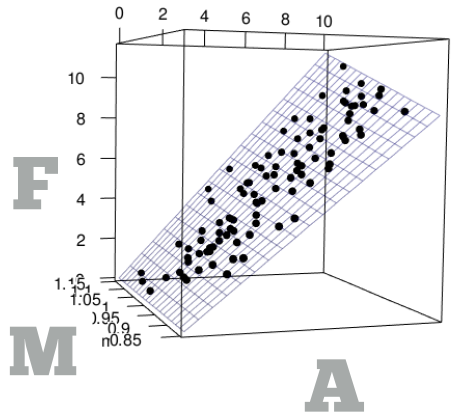
Recap: Data science



Natural laws deal with **variables**, but operate on **values** that appear in the same **observation**.

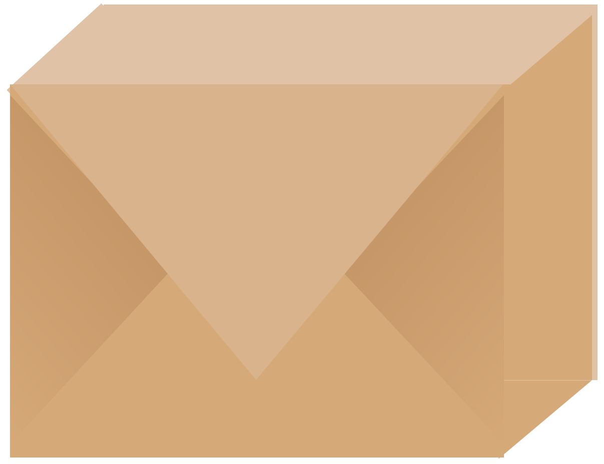


Data contains **values** grouped into **variables** and **observations**.



Laws appear as patterns in data.

EDAWR



An R package with toy data sets that we will use today.

```
# install.packages("devtools")
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
?tb
?population
```

Warm up - Identify the variables

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

cases

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

pollution

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

- Storm name
- Wind Speed (mph)
- Air Pressure
- Date

cases

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

- Country
- Year
- Count

pollution

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

- City
- Amount of large particles
- Amount of small particles

Your Turn - Identify the observations

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

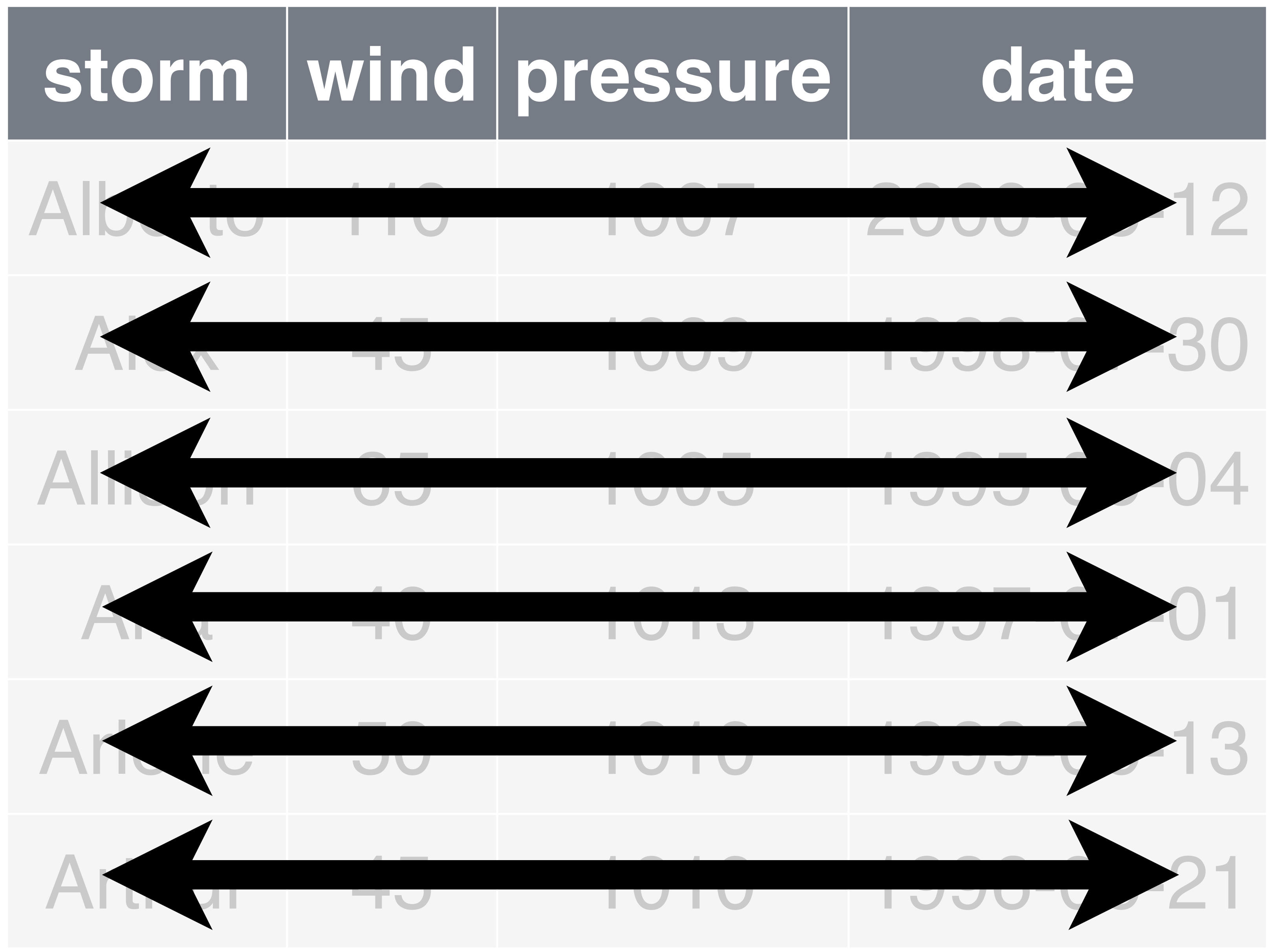
cases

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

pollution

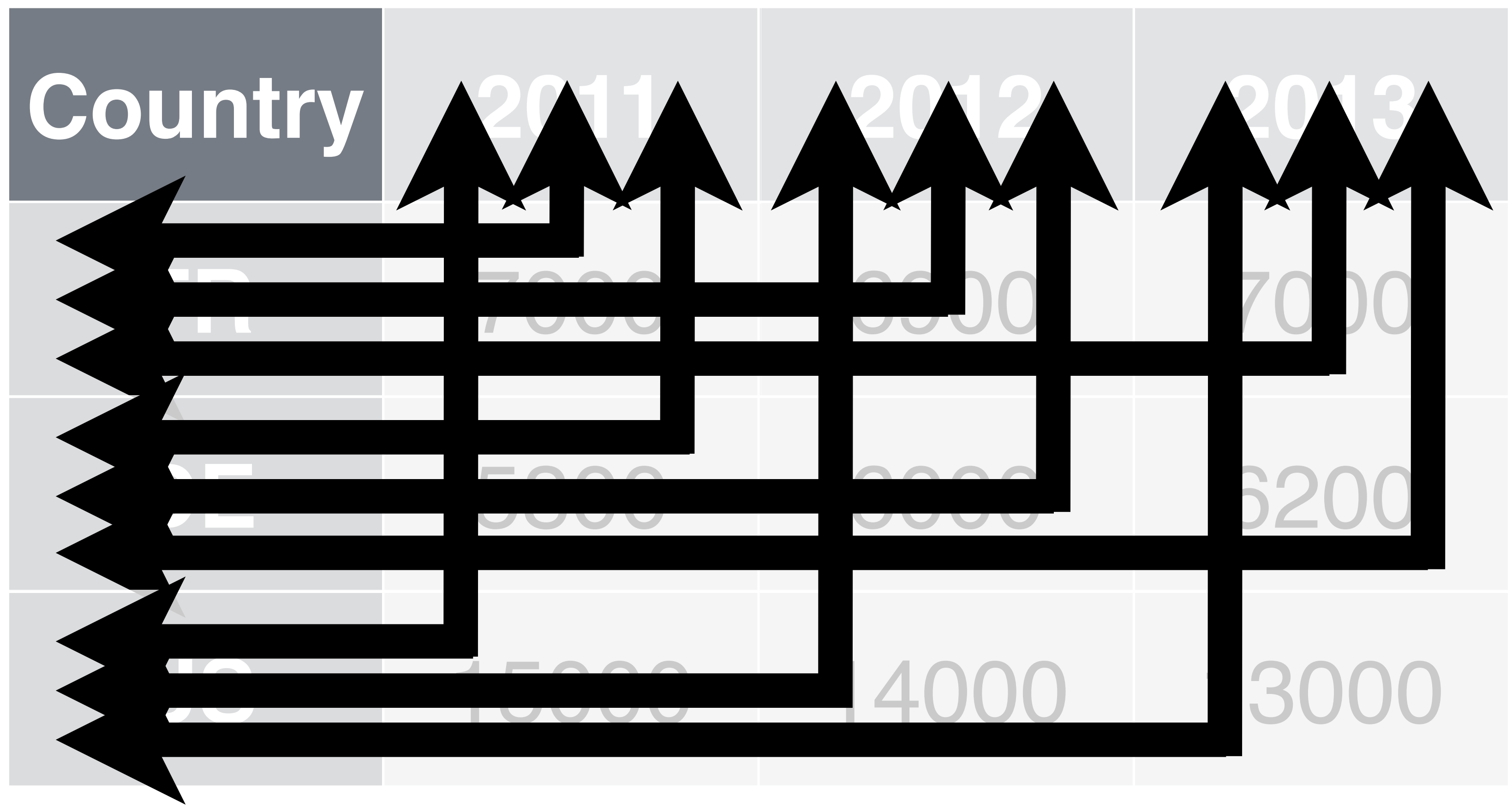
city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

storms



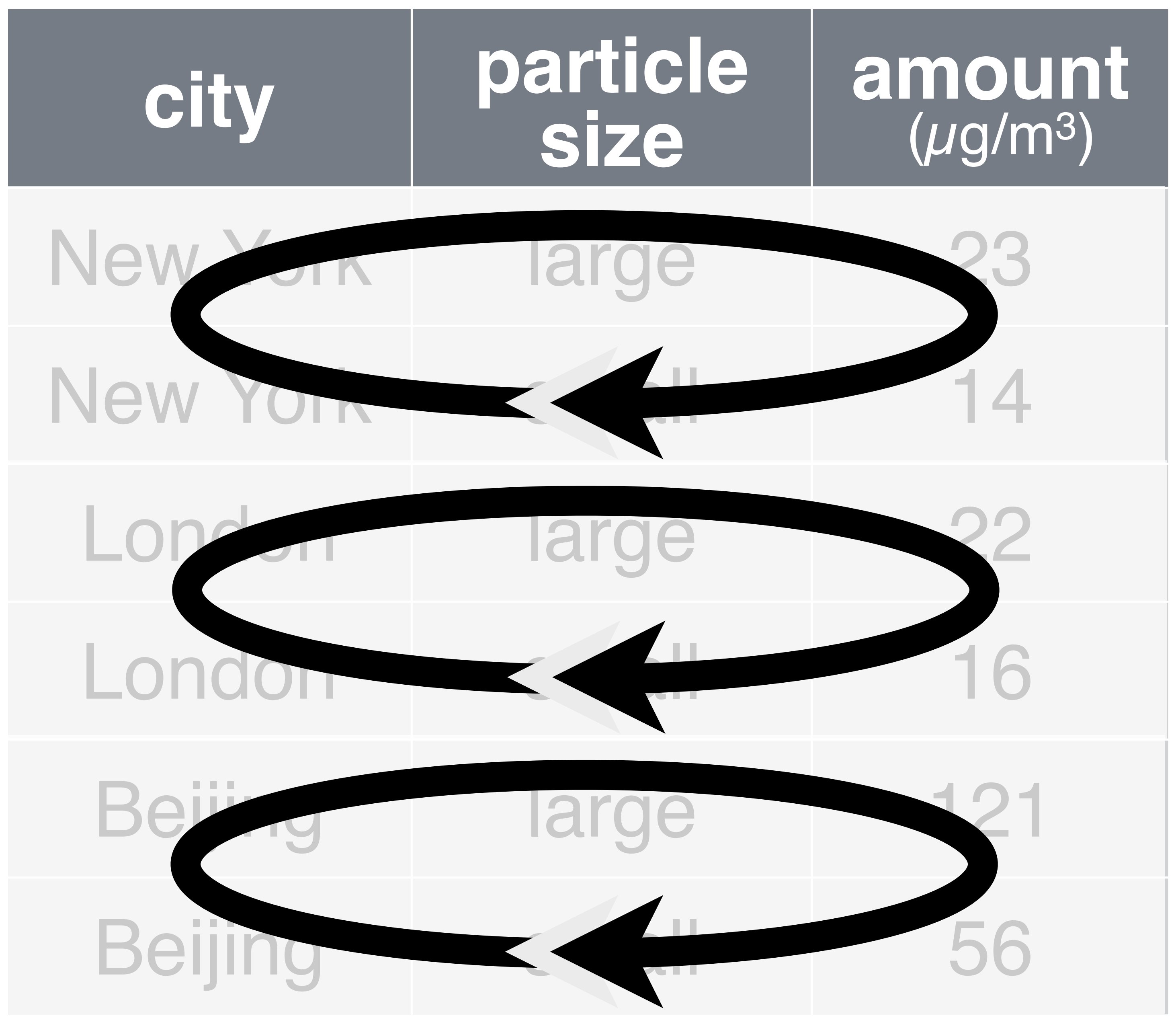
(per storm)

cases



(per country per year)

pollution



(per city)