

Introduction to tidyverse



<https://www.meetup.com/Lucerne-R-User-Group/>



https://github.com/Lucerne-R-User-Group/2020_06_24-inaugural-meeting

Dr Andrea De Angelis



24 June 2020

Structure of the workshop

1 R basics



2

tidyverse & R4DS

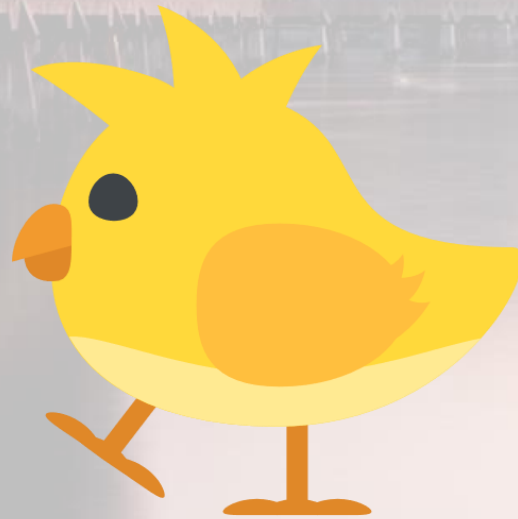


3

dplyr & ggplot2

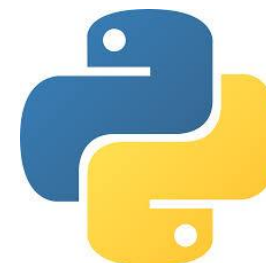


Basic notions





Point-and-click

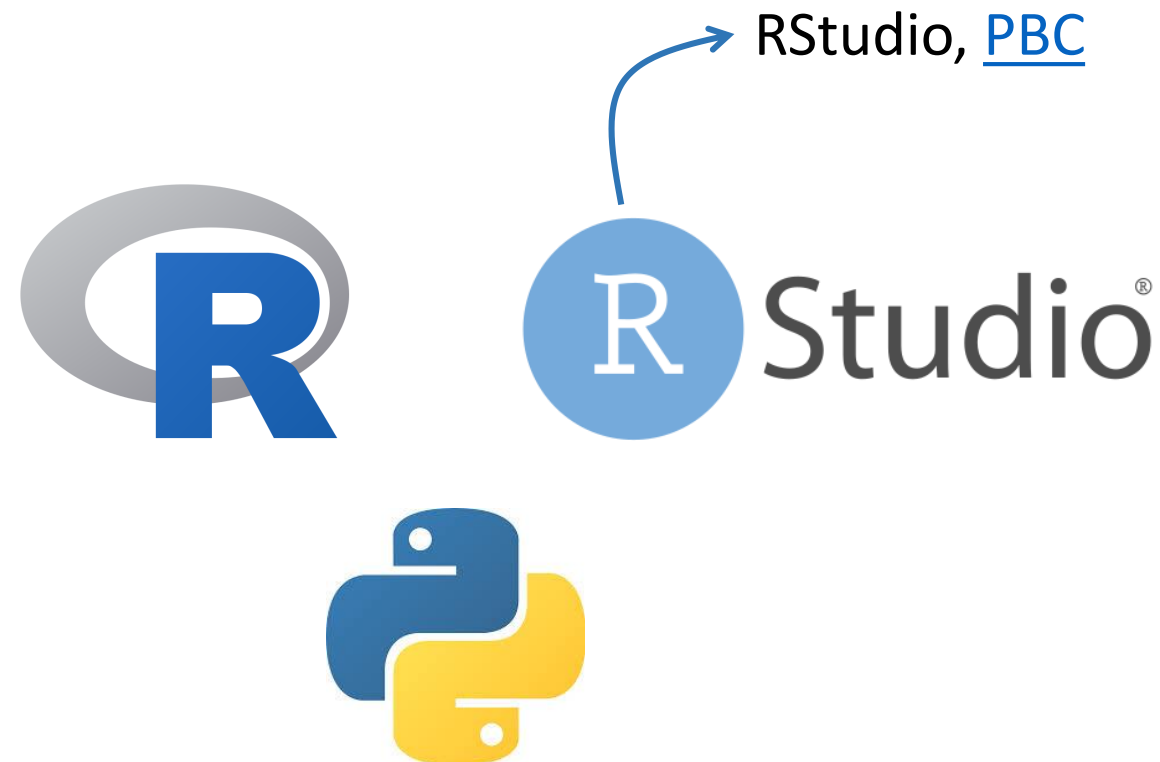


Coding (scripts)





**Proprietary
software**



Free software
(as in "free speech" not as "free beer")





A language



Writing software



R the language

Values

```
1, "Florida", "2010-01-25"
```



R the language

Values

```
1, "Florida", "2010-01-25"
```

Objects

```
x <- 10/3
```



R the language

Values

1, "Florida", "2010-01-25"

Objects

x <- 10/3

A name
without quotes

<-
(it looks like an arrow)

object, values, or
function result



R the language

Values

```
1, "Florida", "2010-01-25"
```

Objects

```
x <- 10/3
```

Available objects
appear in a box
called "global
environment"



The screenshot shows the R Studio Environment pane. At the top are tabs for 'Environment', 'History', and 'Connections'. Below the tabs is a toolbar with icons for file operations and a search bar. The main area is titled 'Global Environment' and contains a table of objects. The table has a header row labeled 'Values' and a single data row for object 'x' with the value '3.333333333333333'.

Values	
x	3.333333333333333

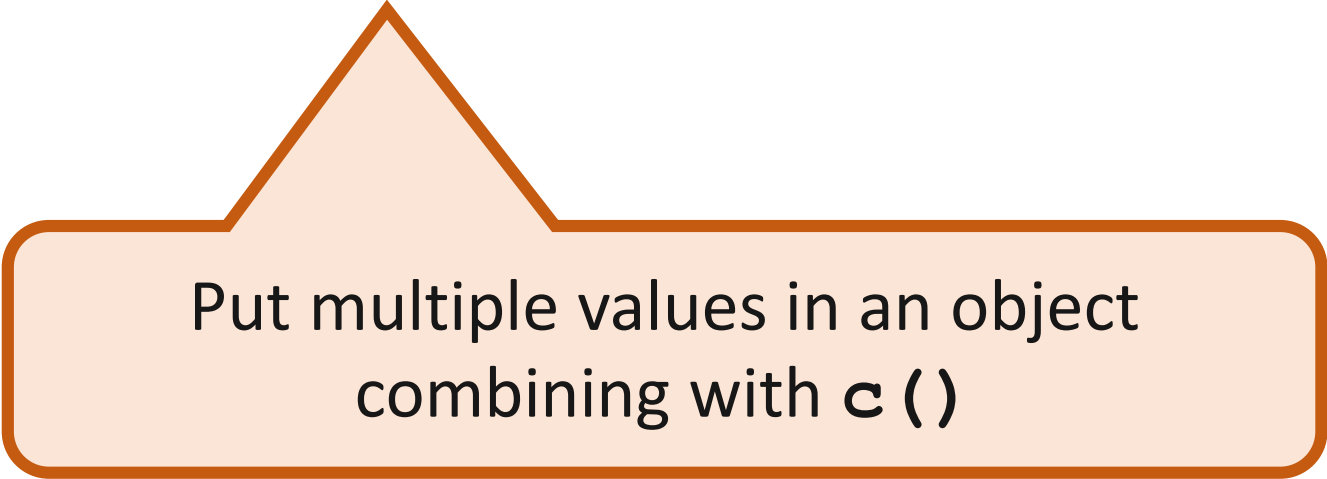
R the language

Values

```
1, "Florida", "2010-01-25"
```

Objects

```
x <- c(10/3, 2, 1.42)
```



Put multiple values in an object
combining with `c()`

R the language

Values

```
1, "Florida", "2010-01-25"
```

Objects

```
x <- c(10/3, 2, 1.42)
```

Functions

```
mean(x, na.rm = TRUE)
```



R the language

Values

```
1, "Florida", "2010-01-25"
```

Objects

```
x <- c(10/3, 2, 1.42)
```

Functions

```
mean(x, na.rm = TRUE)
```



Arguments

R the language

Values

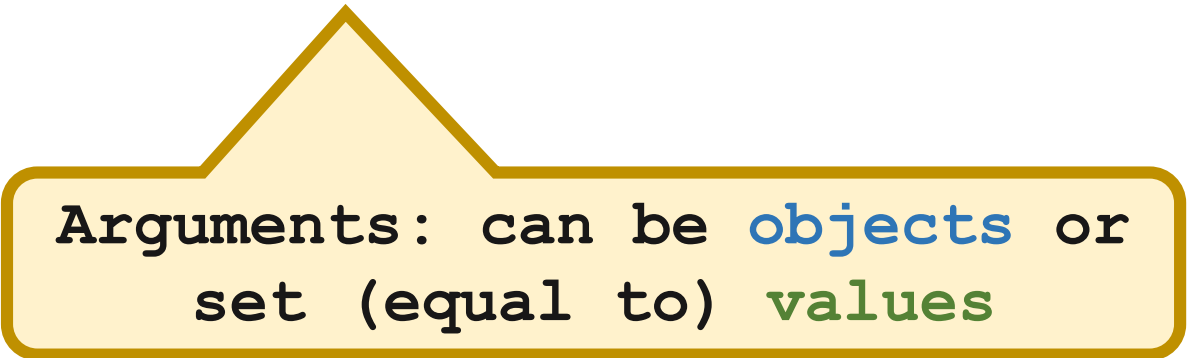
```
1, "Florida", "2010-01-25"
```

Objects

```
x <- c(10/3, 2, 1.42)
```

Functions

```
mean(x, na.rm = TRUE)
```



Arguments: can be **objects** or
set (equal to) **values**

Your Turn

Which of these are **numbers**?

A

1

B

"1"

C

"one"

D

one

00:15

Your Turn

Which of these are **numbers**?



A

1

number

B

"1"

strings



C

"one"

strings

D

one

object



Your Turn

Which of these are **numbers**?



These are
comments



A

1
number

B

"1"
strings



C

"one"
strings

D

one
object



Your Turn

Suppose `one <- 1`. Which of these **will work**?

A

`log(1)`

B

`log(one)`

C

`log("1")`

D

`log("one")`

00:15

Your Turn

Suppose `one <- 1`. Which of these **will work**?



A

`log(1)`

B

`log(one)`



C

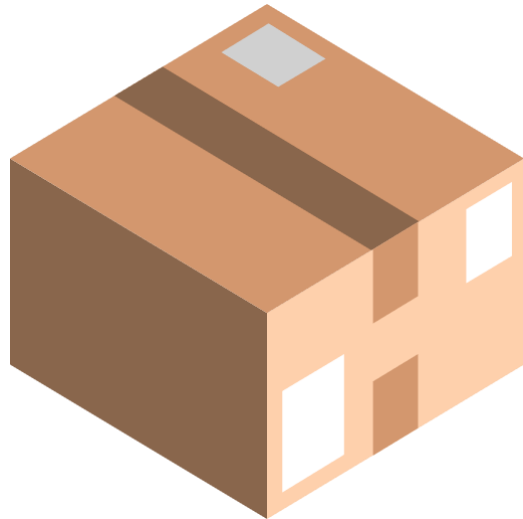
`log("1")`

D

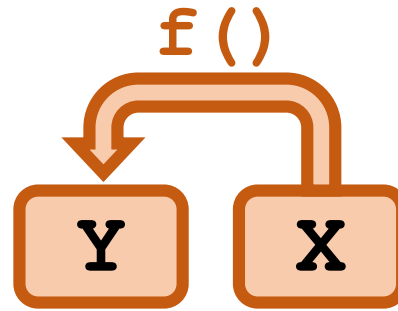
`log("one")`



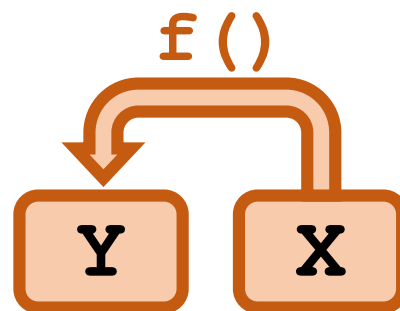
R 'packages'



Functions



Functions



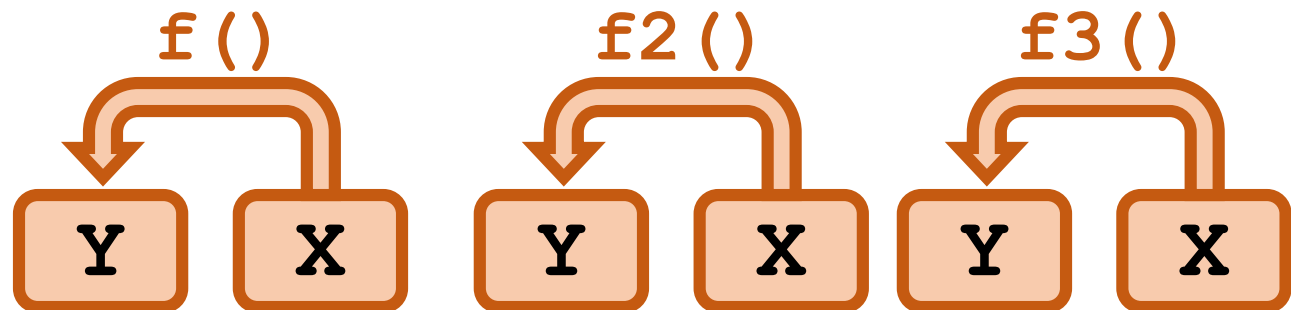
Objects

iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa



Many functions



Many objects

iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

mtcars

	mpg	cyl	displ	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valliant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440

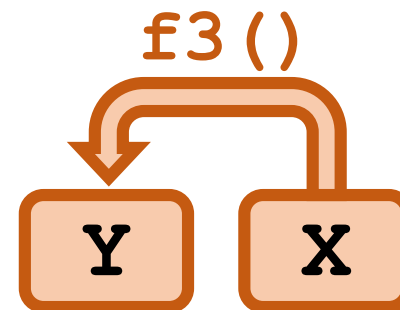
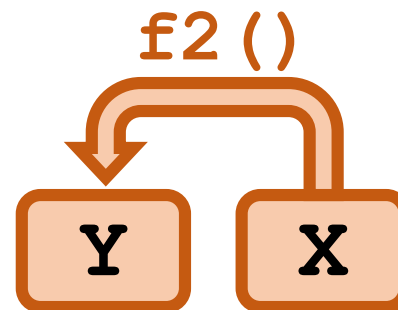
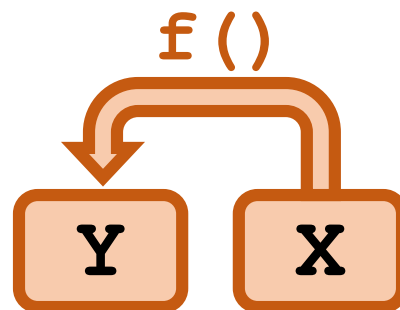
mpg

	manufacturer	model	displ	year	cyl
1	audi	a4	1.8	1999	4
2	audi	a4	1.8	1999	4
3	audi	a4	2.0	2008	4
4	audi	a4	2.0	2008	4
5	audi	a4	2.8	1999	6
6	audi	a4	2.8	1999	6
7	audi	a4	3.1	2008	6
8	audi	a4 quattro	1.8	1999	4



Many functions

Many objects



iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

mtcars

	mpg	cyl	displ	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440

mpg

	manufacturer	model	displ	year	cyl
1	audi	a4	1.8	1999	4
2	audi	a4	1.8	1999	4
3	audi	a4	2.0	2008	4
4	audi	a4	2.0	2008	4
5	audi	a4	2.8	1999	6
6	audi	a4	2.8	1999	6
7	audi	a4	3.1	2008	6
8	audi	a4 quattro	1.8	1999	4

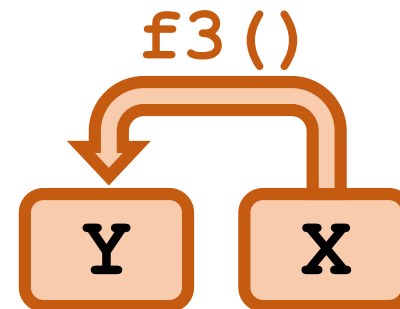
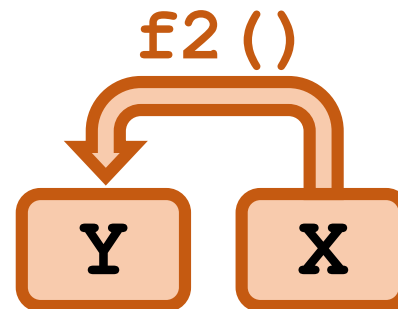
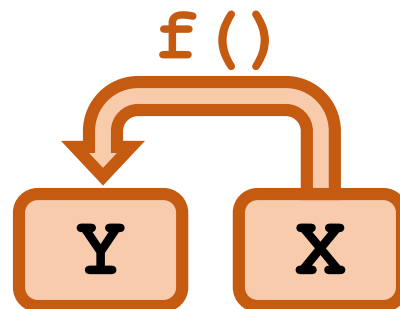
Package

foo



Many functions

Many objects



iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

mtcars

	mpg	cyl	displ	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440

mpg

	manufacturer	model	displ	year	cyl
1	audi	a4	1.8	1999	4
2	audi	a4	1.8	1999	4
3	audi	a4	2.0	2008	4
4	audi	a4	2.0	2008	4
5	audi	a4	2.8	1999	6
6	audi	a4	2.8	1999	6
7	audi	a4	3.1	2008	6
8	audi	a4 quattro	1.8	1999	4

Package

foo

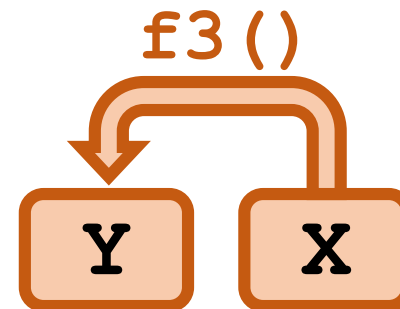
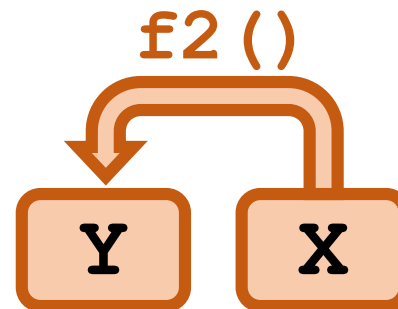
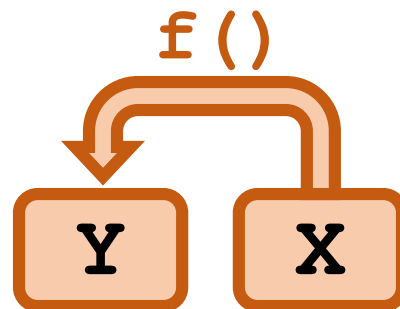


```
install.packages("foo")  
library("foo")  
f2(iris)
```



Many functions

Many objects



iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

mtcars

	mpg	cyl	displ	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440

mpg

	manufacturer	model	displ	year	cyl
1	audi	a4	1.8	1999	4
2	audi	a4	1.8	1999	4
3	audi	a4	2.0	2008	4
4	audi	a4	2.0	2008	4
5	audi	a4	2.8	1999	6
6	audi	a4	2.8	1999	6
7	audi	a4	3.1	2008	6
8	audi	a4 quattro	1.8	1999	4

Package

foo



```
install.packages("foo")  
library("foo")  
f2(iris)
```

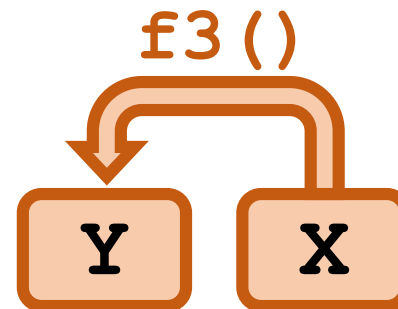
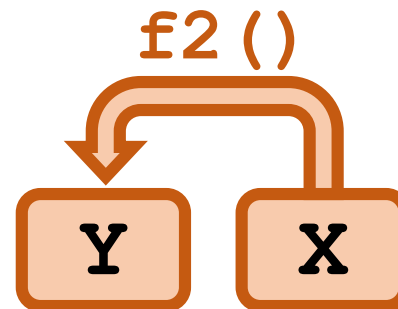
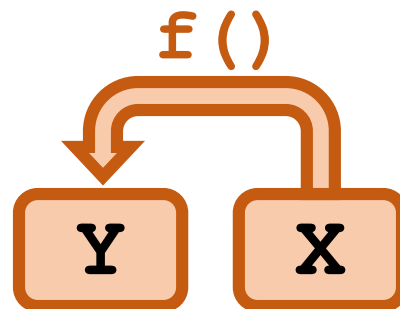
once per
computer

once per
session



Many functions

Many objects



iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

mtcars

	mpg	cyl	displ	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440

mpg

	manufacturer	model	displ	year	cyl
1	audi	a4	1.8	1999	4
2	audi	a4	1.8	1999	4
3	audi	a4	2.0	2008	4
4	audi	a4	2.0	2008	4
5	audi	a4	2.8	1999	6
6	audi	a4	2.8	1999	6
7	audi	a4	3.1	2008	6
8	audi	a4 quattro	1.8	1999	4

Package

foo



```
install.packages("foo")  
foo::f2(iris)
```

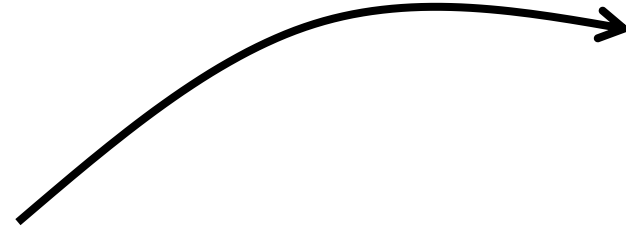
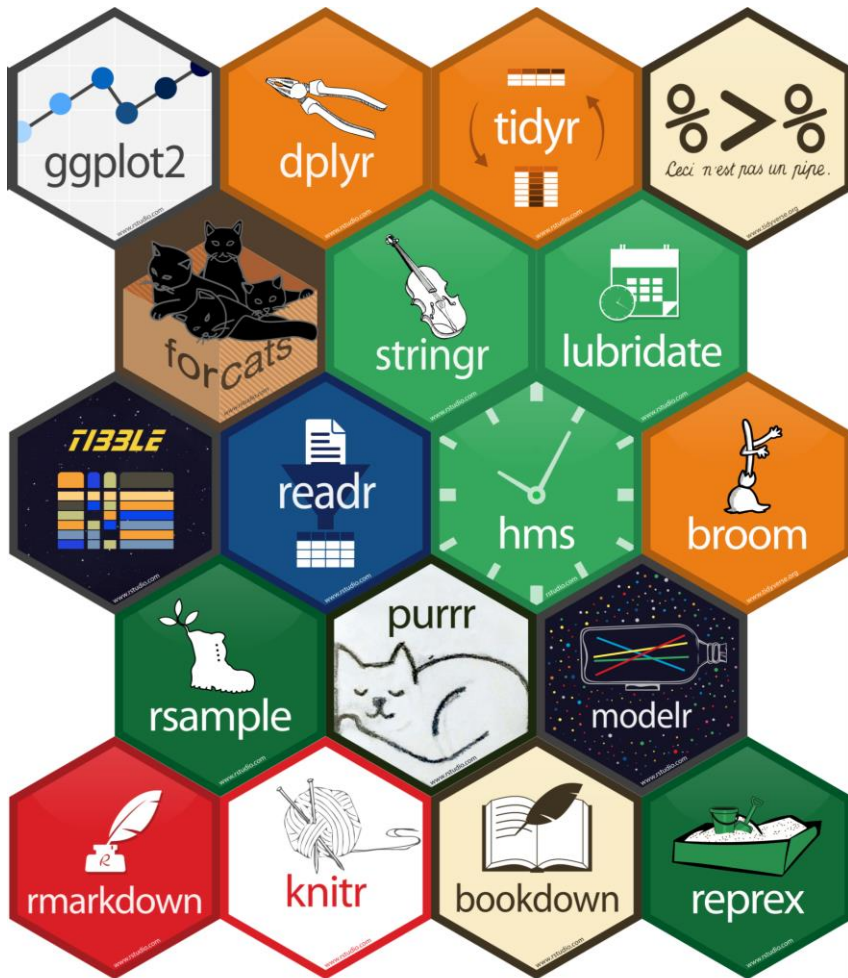
once per
computer



tidyverse & Data Science



Many packages

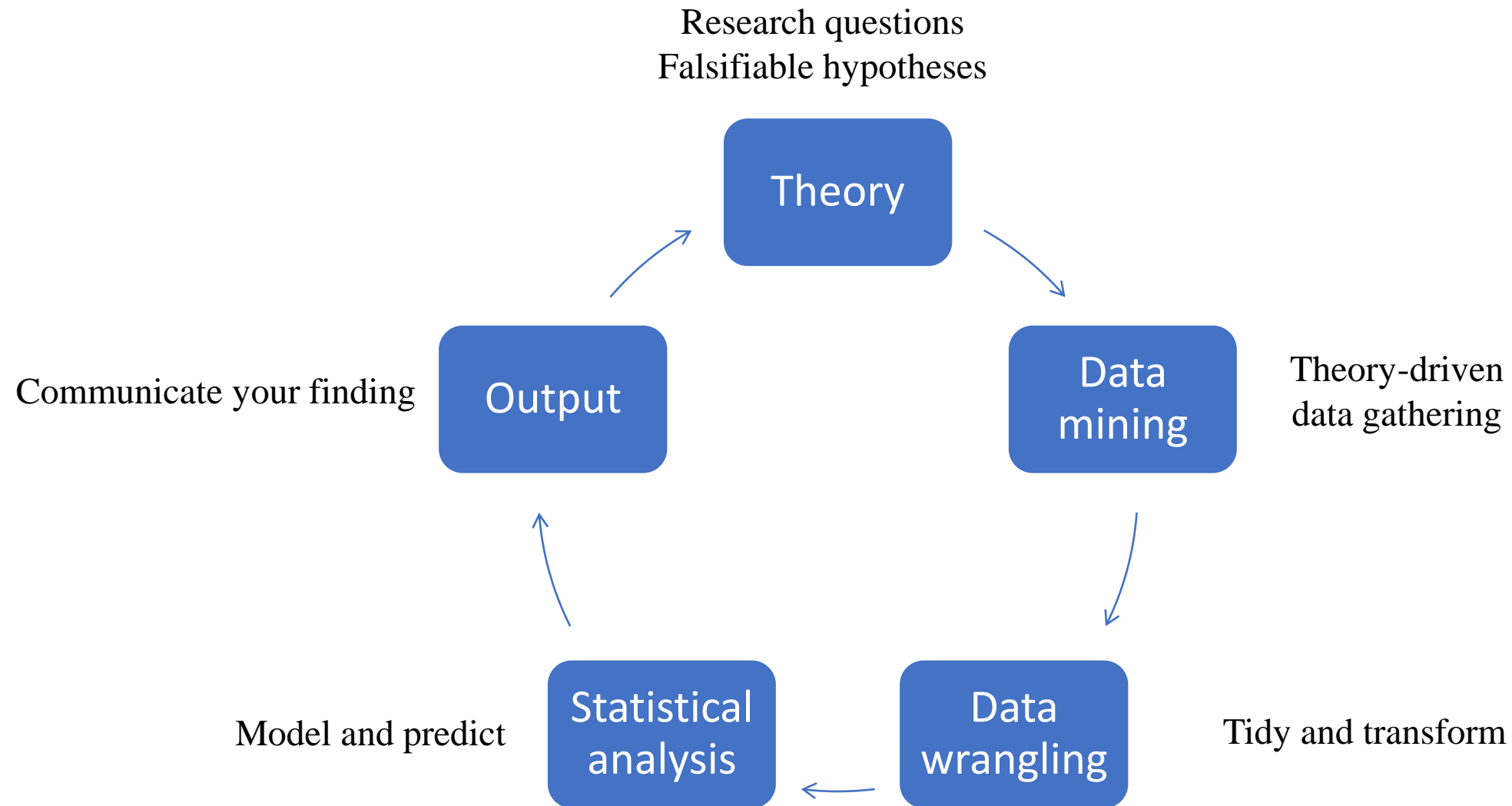


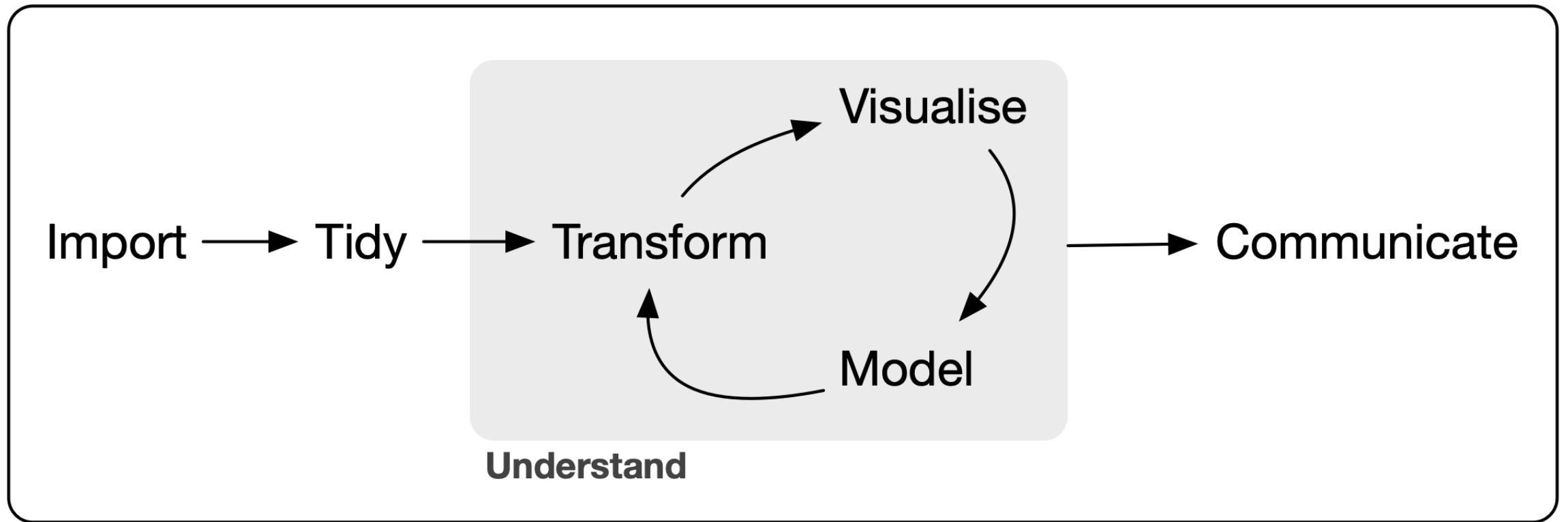
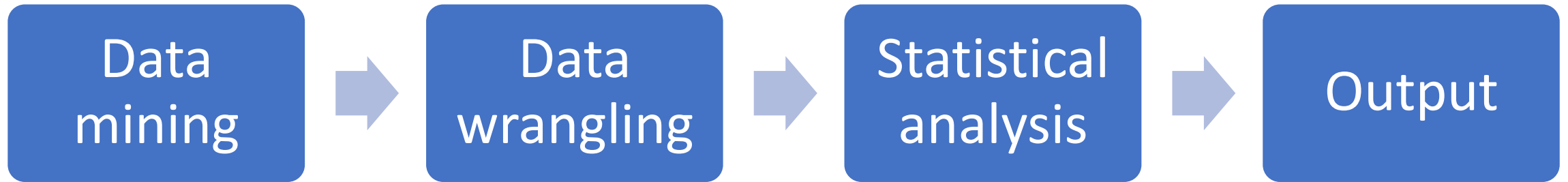
```
# install.packages("tidyverse")  
library("tidyverse")
```

a shortcut

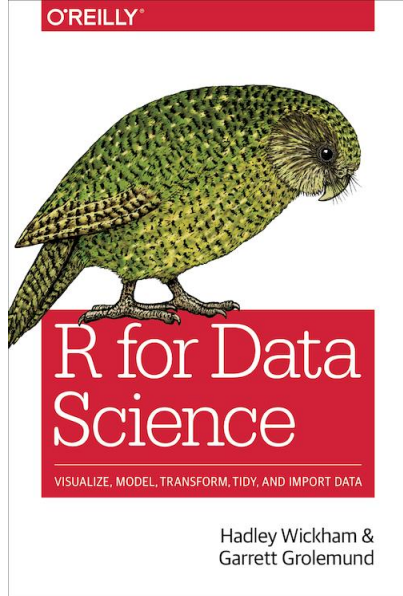


The Data Science Cycle



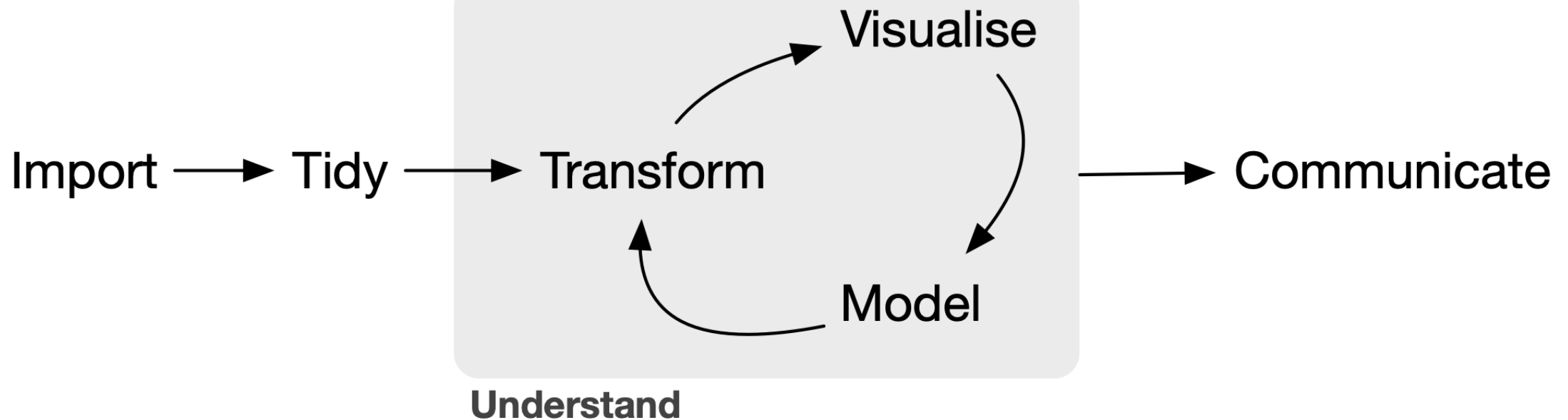


Program

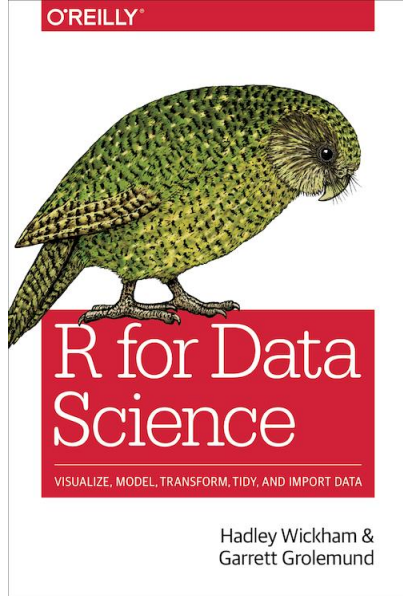


R for Data Science

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

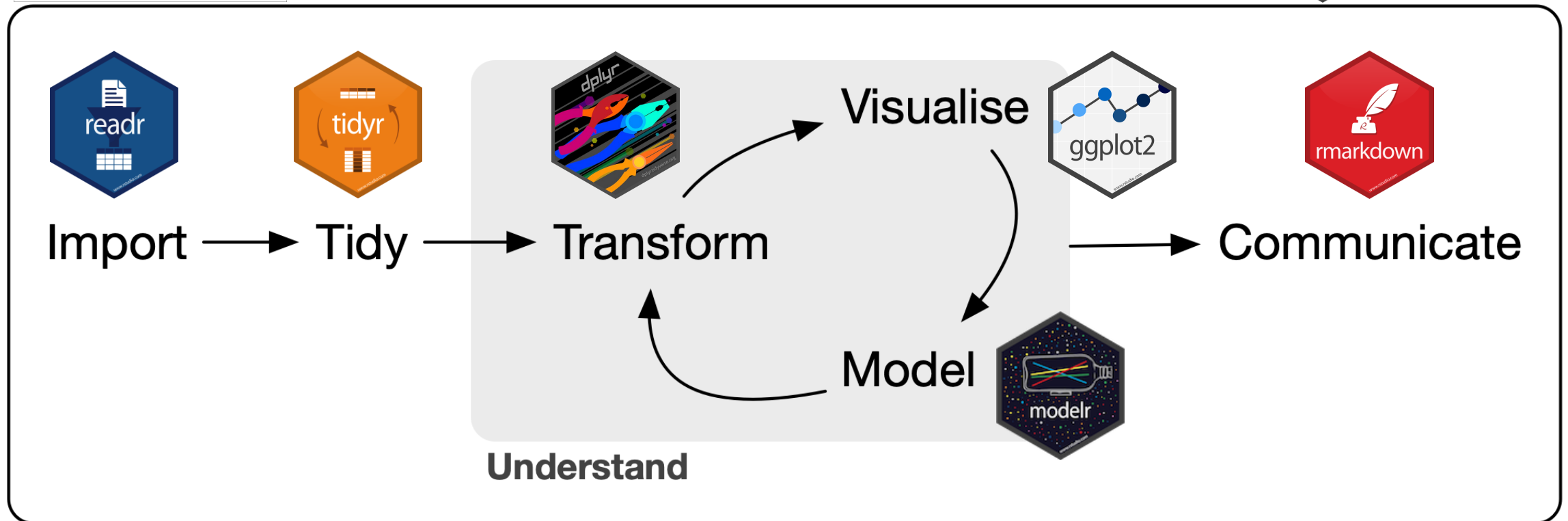


Program



R for Data Science

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



Program

Your Turn

If you don't have RStudio installed:

Access RStudio Cloud at [this link](#)

Register / Log in > Projects

If you do have RStudio installed:

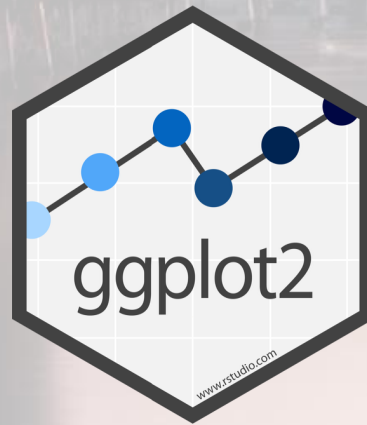
Clone this [GitHub repo](#)

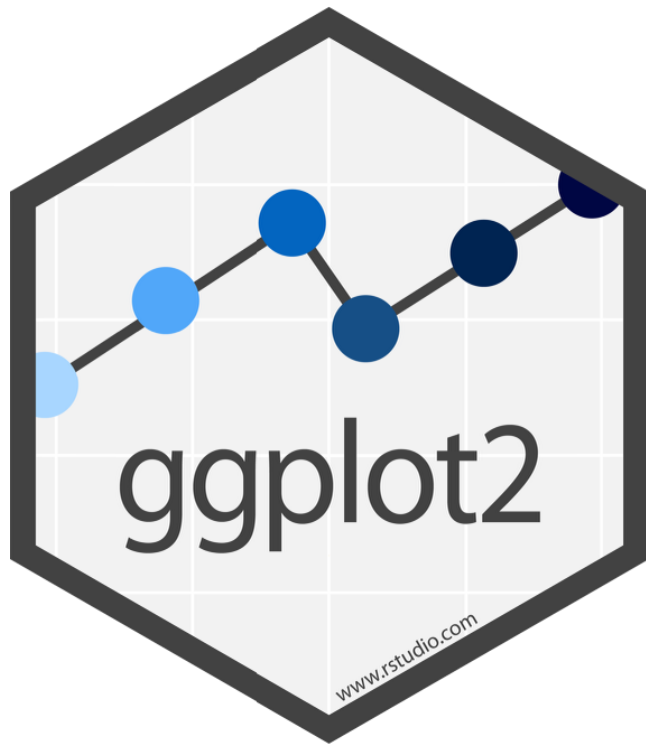
Then, open ``01_introduction.R``

Load the tidyverse packages

03:00

Data visualization with ggplot2





A package for creating graphics based on common principles (Grammar of Graphics).

You provide **data**, tell ggplot2 how to **map variables to aesthetics** and which **geometries** to use. ggplot2 takes care of the rest.

ggplot2 cheatsheet

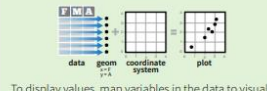
Data Visualization with ggplot2

Cheat Sheet

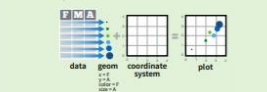


Basics

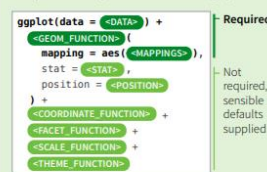
ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.



`ggplot(data = mpg, aes(x = cty, y = hwy))`
Begins a plot that you finish by adding layers to. Add one geom function per layer.

`geom_point()`
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

`last_plot()`
Returns the last plot

`ggsave("plot.png", width = 5, height = 5)`
Saves last plot as 5"x5" file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(cty, hwy))

a + geom_blank()
# Useful for expanding limits

b + geom_curve(aes(yend = lat + 1,
  xend = long + 1, curvature = 2)) ~ x, yend,
  alpha, angle, color, curvature, linetype, size

a + geom_path(linetype = "butt",
  linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat,
  xmax = long + 1, ymax = lat + 1)) ~ xmax, xmin,
  ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900,
  ymax = unemploy + 900)) ~ x, y, alpha, color, fill, group, linetype, size
```

Line Segments

```
common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))
b + geom_segment(aes(xend = lat + 1, yend = long + 1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))
```

One Variable

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, group, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight

Discrete
d <- ggplot(mpg, aes(fill))
d + geom_bar()
```

Two Variables

```
Continuous X, Continuous Y
e <- ggplot(diamonds, aes(carat, price))

e + geom_label(aes(label = cty), nudge_x = 1,
  nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface,
  hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size, stroke

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = "lm")
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1,
  nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface,
  hjust, lineheight, size, vjust

Discrete X, Continuous Y
f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha,
  color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y",
  stackdir = "center")
x, y, alpha, color, fill, group

f + geom_violin(scale = "area")
x, y, alpha, color, fill, group, linetype, size,
  weight

Maps
data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
k + geom_map(map_id = state, map = map) +
  expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size
```

Maps

```
Discrete X, Discrete Y
g <- ggplot(diamonds, aes(cut, color))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke
```

Three Variables

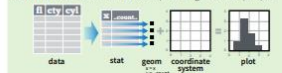
```
seals2 <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))

l + geom_raster(aes(fill = z), hjust = 0.5,
  vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

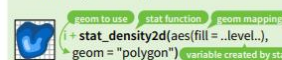
l + geom_tile(aes(fill = z))
x, y, alpha, color, fill, linetype, size, weight
```

Stats

An alternative way to build a layer



Visualize a stat by changing the default stat of a geom function, `geom_bar(stat = "count")` or by using a stat function, `stat_count(geom = "bar")`, which calls a default geom to make a layer (equivalent to a geom function). Use `..name..` syntax to map stat variables to aesthetics.



```
c + stat_bin(binwidth = 1, origin = 0)
x, y, count, ..ncount.., ..density..

c + stat_count(width = 1) ~ x, y, count, ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, count, ..density.., ..scaled..
```

```
e + stat_bin2d(bins = 30, drop = T)
x, y, fill, count, density

e + stat_bin_hex(bins = 30) ~ x, y, fill, count, density

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size, level

e + stat_ellipse(level = 0.95, segments = 51, type = "t")
```

```
l + stat_contour(aes(z = z)) ~ x, y, z, order | level

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | value

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | value
```

```
f + stat_boxplot(coef = 1.5)
x, y, lower, middle, upper, width, ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area")
x, y, density, ..scaled.., ..width..
```

```
e + stat_ecdf(n = 40) ~ x, y, .., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9),
  formula = y ~ log(x), method = "rq") ~ x, y, ..quantile..

f + stat_smooth(method = "lm", formula = y ~ x,
  se = T, level = 0.95) ~ x, y, .., .., ..ymin.., ..ymax..
```

```
ggplot() + stat_function(aes(x = -3:3), n = 99,
  fun = dnorm, args = list(sd = 0.5)) ~ x, .., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  dparam = list(df = 5)) ~ sample, x, y | sample, theoretical

e + stat_sum(x, y, size | .., .., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary(fun.y = "mean", geom = "bar")
e + stat_unique()
```

```
ggplot() + stat_function(aes(x = -3:3), n = 99,
  fun = dnorm, args = list(sd = 0.5)) ~ x, .., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  dparam = list(df = 5)) ~ sample, x, y | sample, theoretical

e + stat_sum(x, y, size | .., .., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary(fun.y = "mean", geom = "bar")
e + stat_unique()
```

```
ggplot() + stat_function(aes(x = -3:3), n = 99,
  fun = dnorm, args = list(sd = 0.5)) ~ x, .., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  dparam = list(df = 5)) ~ sample, x, y | sample, theoretical

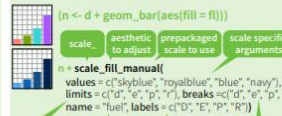
e + stat_sum(x, y, size | .., .., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary(fun.y = "mean", geom = "bar")
e + stat_unique()
```

General Purpose

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



```
n <- d + geom_bar(aes(fill = fill))

scale_fill_manual(values = c("skyblue", "royalblue", "blue", "navy"),
  limits = c("d", "e", "p", "r"), breaks = c("d", "e", "p", "r"),
  name = "fuel", labels = c("D", "E", "P", "R"))
```

General Purpose scales
Use with most aesthetics

```
scale_continuous() - map cont' values to visual ones
scale_discrete() - map discrete values to visual ones
scale_identity() - use data values as visual ones
scale_manual(values = c()) - map discrete values to manually chosen labels
scale_date(date_labels = "%m/%d",
  date_breaks = "2 weeks") - treat data values as dates.
scale_datetime() - treat data x values as dates.
Use same arguments as scale_x_date().
See strftime for label formats.
```

X and Y location scales
Use with x or y aesthetics (x shown here)

```
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale
```

Color and fill scales (Discrete)
n <- d + geom_bar(aes(fill = fill))

n + scale_fill_brewer(palette = "Blues")
For palette choices: RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

Color and fill scales (Continuous)
o <- c + geom_dotplot(aes(fill = ..x..))

o + scale_fill_distiller(palette = "Blues")

o + scale_fill_gradient(low = "red", high = "yellow")

o + scale_fill_gradient2(low = "red", high = "blue",
 mid = "white", midpoint = 25)

o + scale_fill_github(colors = topo.colors(6),
 cm.colors(), RColorBrewer::brewer.pal(1))

Shape and size scales
p <- e + geom_point(aes(shape = fl, size = cyl))

p + scale_shape() + scale_size()
For point choices: RColorBrewer::display.brewer.all()

p + scale_shape_manual(values = c(3:7))

o + stat_sum(x, y, size | .., .., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary(fun.y = "mean", geom = "bar")
e + stat_unique()

General Purpose

Coordinate Systems

r <- d + geom_bar()
xlim, ylim

The default cartesian coordinate system
r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction = 1)
theta, start, direction
Polar coordinates

r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

n + coord_quickmap()
n + coord_map(projection = "ortho",
 orientation = c(41, -74, 0))
projection, orientation, xlim, ylim
Map projections from the mapr package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height

e + geom_point(position = "jitter")
Add random noise to x and y position of each element to avoid overlapping

e + geom_label(position = "nudge")
Nudge labels away from points

s + geom_bar(position = "stack")
Stack elements on top of one another
```

Each position adjustment can be recast as a function with manual **width** and **height** arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

Themes

```
r + theme_bw()
White background with grid lines

r + theme_gray()
Grey background (default theme)

r + theme_dark()
Dark for contrast
```

```
r + theme_classic()
Minimal theme

r + theme_minimal()
Minimal theme

r + theme_void()
Empty theme
```

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(~ fl)
Facet into columns based on fl

t + facet_grid(row ~ .)
Facet into rows based on year

t + facet_grid(row ~ fl)
Facet into both rows and columns

t + facet_wrap(~ fl)
Wrap facets into a rectangular layout
```

Set scales to let axis limits vary across facets
t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets

Set labeller to adjust facet labels
t + facet_grid(drv ~ fl, labeller = label_both)

```
fl: c fl: d fl: e fl: p fl: r
t + facet_grid(fl ~ fl, labeller = label_both, alpha = .8)
fl: c fl: d fl: e fl: p fl: r
t + facet_grid(fl ~ fl, labeller = label_paired)
```

Labels
t + labs(x = "New x axis label", y = "New y axis label",
 title = "Add a title above the plot",
 subtitle = "Add a subtitle below plot",
 caption = "Add a caption below plot",
 <AES> = "New <AES> legend title")

Use scale functions to update legend labels
geom to place manual values for geom's aesthetics

Legends
n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

n + scale_fill_discrete(name = "Title",
 labels = c("A", "B", "C", "D", "E"))
Set legend title and labels with a scale function.

Zooming
Without clipping (preferred)
t + coord_cartesian(
 xlim = c(0, 100), ylim = c(10, 20))

With clipping (removes unseen data points)
t + xlim(0, 100) + ylim(10, 20)
t + scale_x_continuous(limits = c(0, 100)) +
 scale_y_continuous(limits = c(10, 20))



<https://www.meetup.com/Lucerne-R-User-Group/>

Three elements of visualization

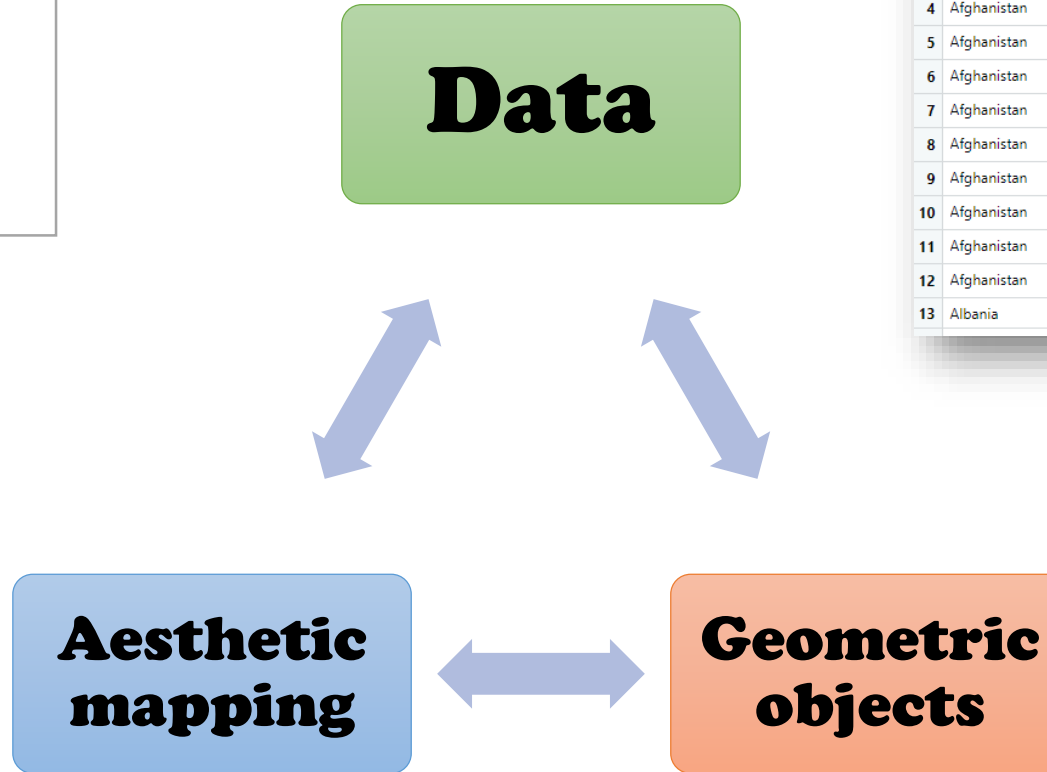
You provide **data**, tell
ggplot2 how to **map**
variables to **aesthetics** and
which **geometries** to use.

Three elements of visualization

You provide **data**, tell ggplot2 how to **map** variables to **aesthetics** and which **geometries** to use.

	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.80100	8425333	779.4453
2	Afghanistan	Asia	1957	30.33200	9240934	820.8530
3	Afghanistan	Asia	1962	31.99700	10267083	853.1007
4	Afghanistan	Asia	1967	34.02000	11537966	836.1971
5	Afghanistan	Asia	1972	36.08800	13079460	739.9811
6	Afghanistan	Asia	1977	38.43800	14880372	786.1134
7	Afghanistan	Asia	1982	39.85400	12881816	978.0114
8	Afghanistan	Asia	1987	40.82200	13867957	852.3959
9	Afghanistan	Asia	1992	41.67400	16317921	649.3414
10	Afghanistan	Asia	1997	41.76300	22227415	635.3414
11	Afghanistan	Asia	2002	42.12900	25268405	726.7341
12	Afghanistan	Asia	2007	43.82800	31889923	974.5803
13	Albania	Europe	1952	55.23000	1282697	1601.0561

Aesthetic	Variable
x	year
y	lifeExp
size	pop
shape	Continent
...	...



Geometry	
point	
line	
smoother	
...	

mpg data

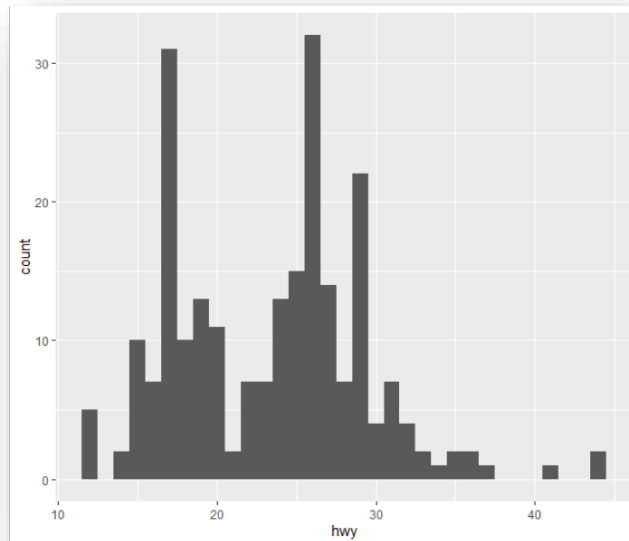
Fuel economy data for 38 models of cars

```
mpg # print data
```

```
?mpg # access documentation
```

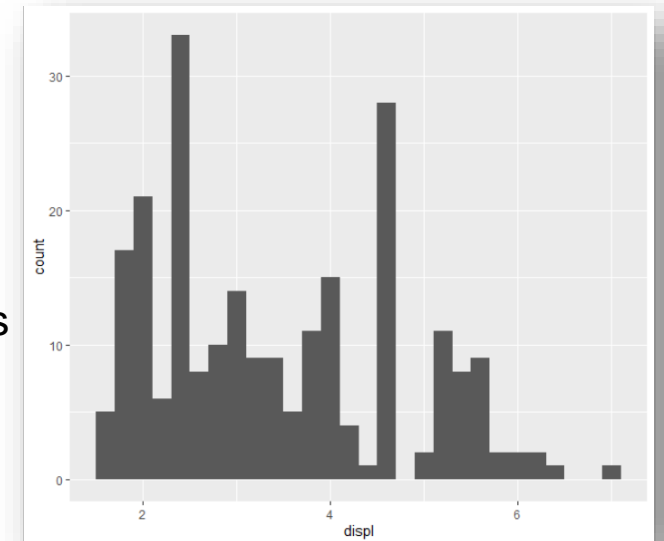
hwy

highway miles per gallon



displ

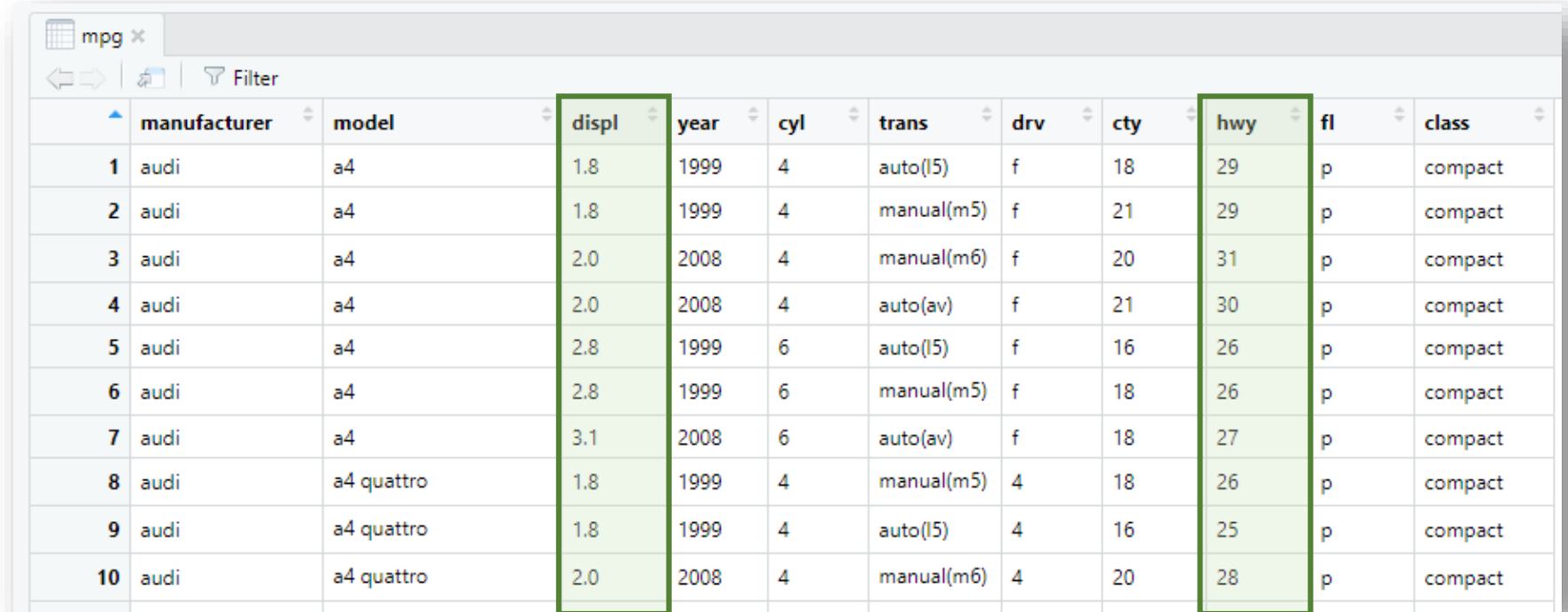
engine displacement, in litres



mpg data

Fuel economy data for 38 models of cars

View (mpg)



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

Your Turn

Read the following code and **mentally predict** the resulting plot.

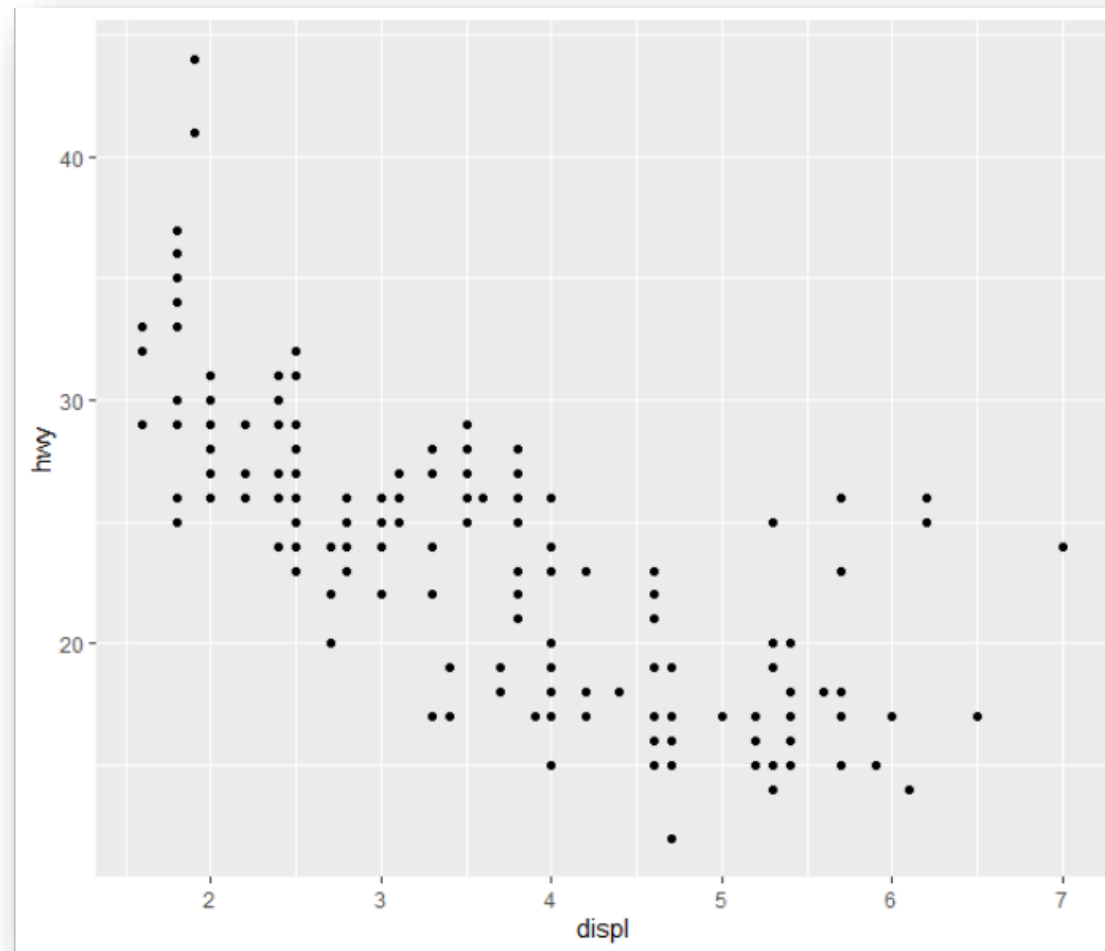
Only then, type it in the **01_introduction** script and execute.

Tip: pay attention to spelling, capitalization, and parentheses!

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

02:00

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```



Data

+ before new line

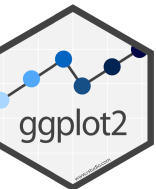
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

type of layer

aes()

x variable

y variable



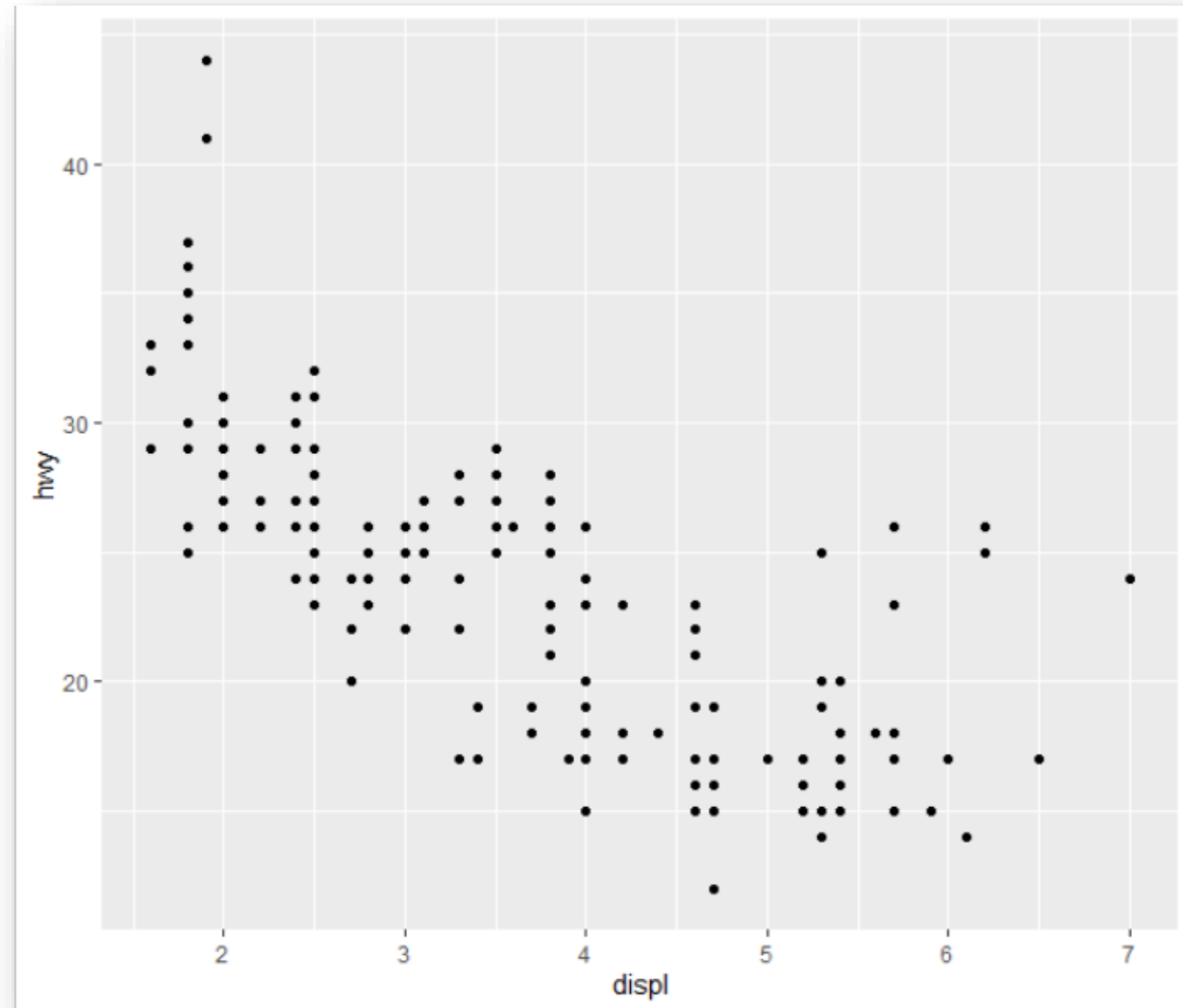
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

1. Initialize a plot with `ggplot()`
2. Add layers with `geom_***()`
3. Map variables with `aes()`

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

1. Initialize a plot with `ggplot(data)`
2. Add layers with `geom_***(mapping)`
3. Map variables with `aes(x, y, size, ...)`

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```



Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I **draw a "smoother"** instead of the points?

Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I draw a "smoother" instead of the points?

A

Changing the
mapping variables

B

Changing the data

C

Changing the geometry

D

Changing one of the
aesthetics

00:15

Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I draw a smoother instead of the points?



A

Changing the
mapping variables

B

Changing the data



C

Changing the geometry

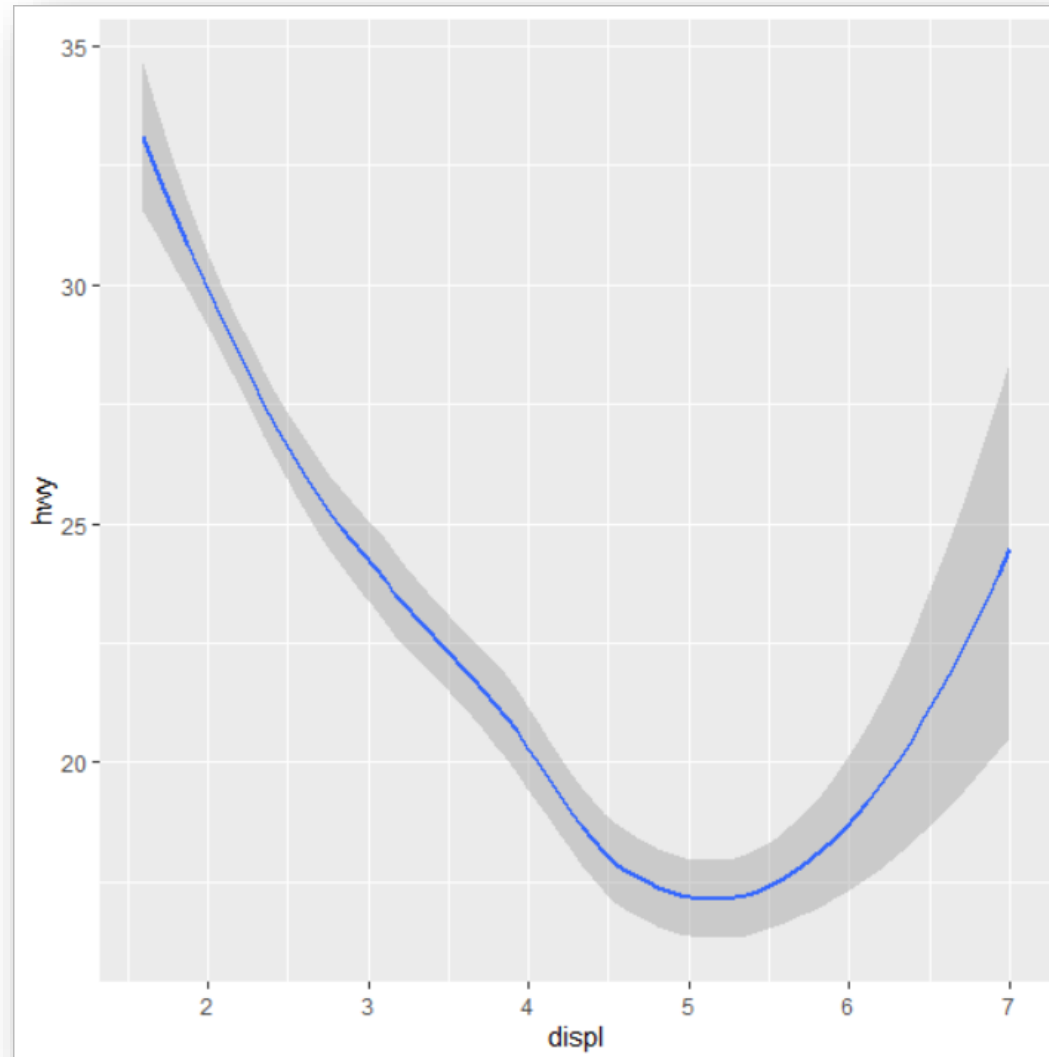
D

Changing one of the
aesthetics



00:15

```
ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I **add a smoother** on top of the points?

Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I add a smoother on top of the points?

A

Adding one
mapping variable

B

Adding one more
data set

C

Adding one
geometry

D

Adding one
aesthetic

00:15

Your Turn

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

How can I add a smoother on top of the points?



A

Adding one
mapping variable

B

Adding one more
data set



C

Adding one
geometry

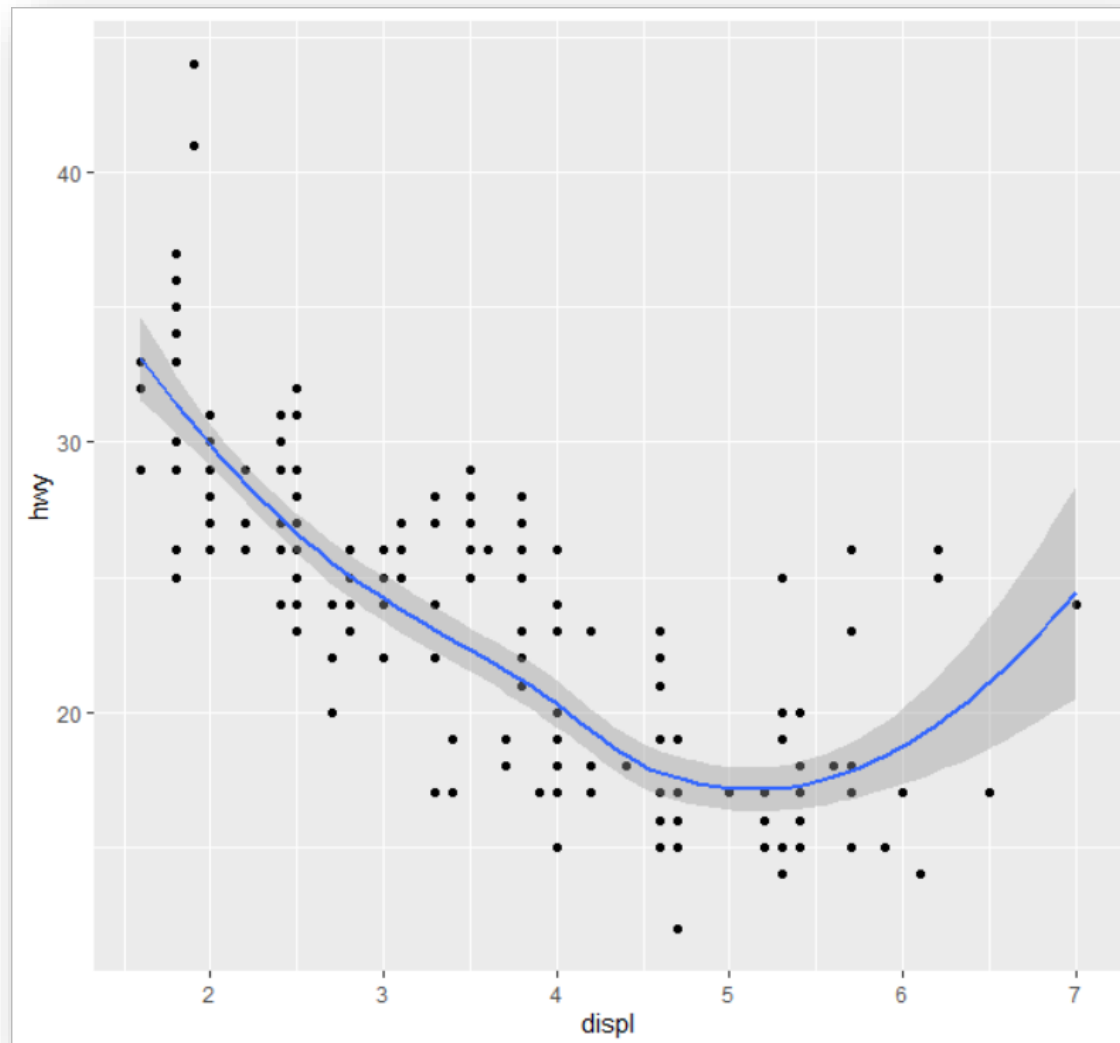
D

Adding one
aesthetic



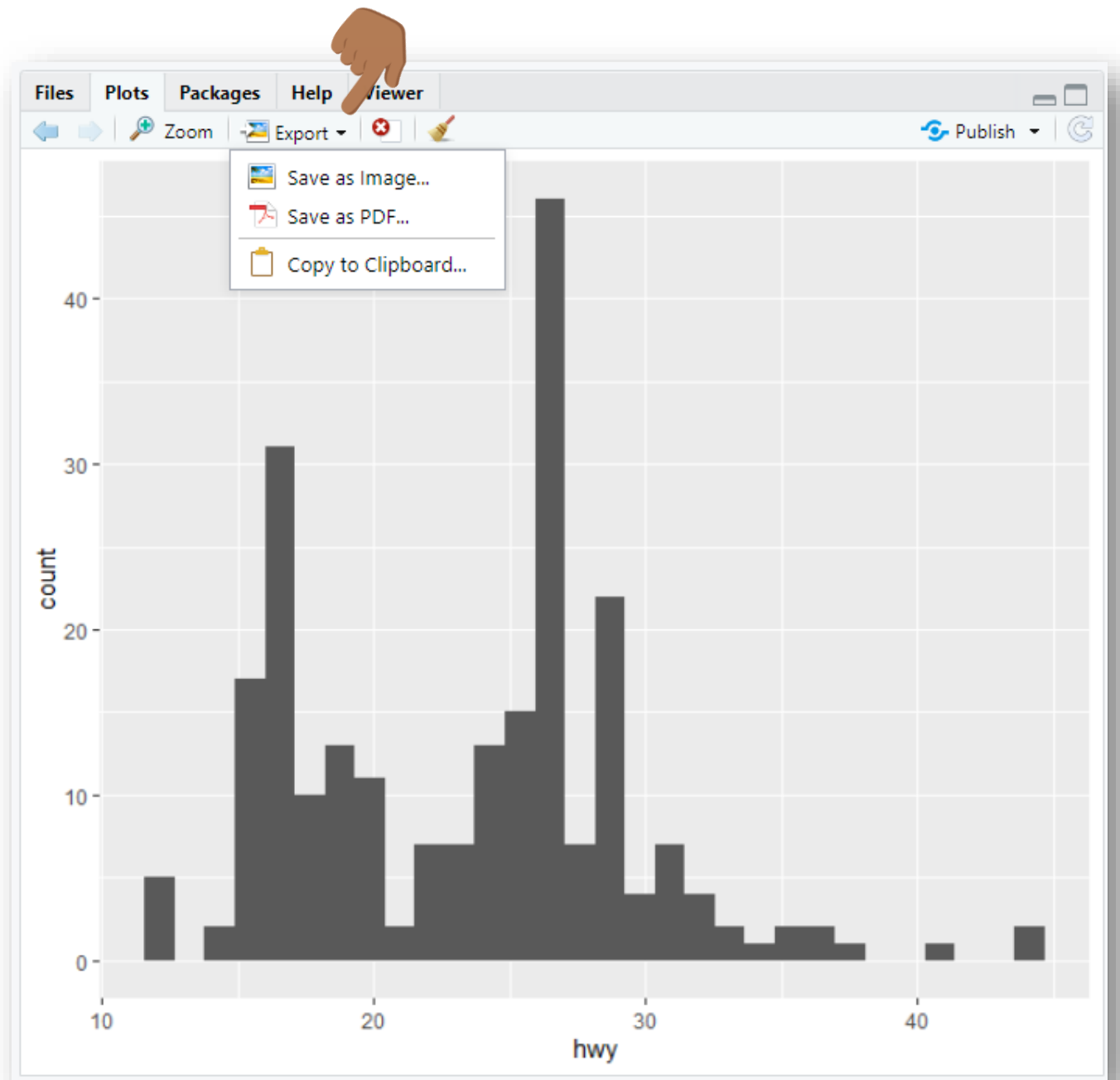
00:15

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



Saving your plots

Tip: `?ggsave()`



Questions

?

data manipulation with dplyr





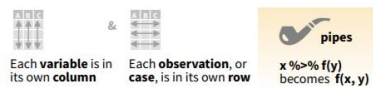
A package for data transformation.

It provides six simple "verbs" corresponding to the most common data manipulation tasks.

Data transformation cheat sheet

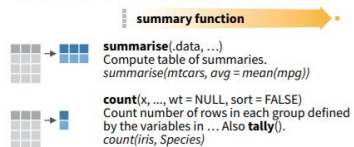
Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect tidy data. In tidy data:



Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



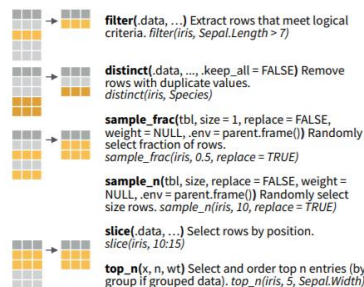
group_by(data, ..., add = FALSE)
Returns copy of table grouped by ...
`g_iris <- group_by(iris, Species)`

ungroup(x, ...)
Returns ungrouped copy of table.
`ungroup(g_iris)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

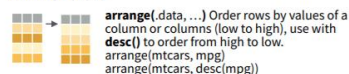


Logical and boolean operators to use with filter()

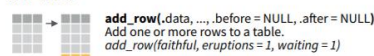
`<` `<=` `is.na()` `%in%` `|` `xor()`
`>` `>=` `!is.na()` `!` `&`

See ?base::Logic and ?Comparison for help.

ARRANGE CASES



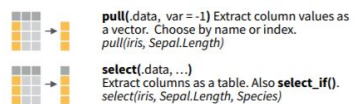
ADD CASES



Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

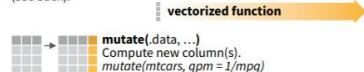


Use these helpers with **select()**, e.g. `select(iris, starts_with("Sepal"))`

contains(match) **num_range(prefix, range)** **ends_with(match)** **one_of(...)** **starts_with(match)**
e.g. `mpg:cyl` `Species`

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).



transmute(data, ...)
Compute new column(s), drop others.
`transmute(mtcars, gpm = 1/mpg)`

mutate_all(tbl, funs, ...) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
`mutate_all(faithful, funs(log(), log2(), log10()))`
`mutate_if(iris, is.numeric, funs(log(), log2(), log10()))`

mutate_at(tbl, cols, funs, ...) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for **select()**.
`mutate_at(iris, vars(Species), funs(log(), log2(), log10()))`

add_column(data, ..., before = NULL, after = NULL) Add new column(s). Also **add_count()**, **add_tally()**, **add_column(mtcars, new = 1:32)**

rename(data, ...) Rename columns.
`rename(iris, Length = Sepal.Length)`

Vector Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS

dplyr::lag() - Offset elements by 1
dplyr::lead() - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::cumall() - Cumulative all()
dplyr::cumany() - Cumulative any()
dplyr::cummax() - Cumulative max()
dplyr::cummean() - Cumulative mean()
dplyr::cummin() - Cumulative min()
dplyr::cumprod() - Cumulative prod()
dplyr::cumsum() - Cumulative sum()

RANKINGS

dplyr::cume_dist() - Proportion of all values <=
dplyr::dense_rank() - rank w ties = min, no gaps
dplyr::min_rank() - rank with ties = min
dplyr::ntile() - bins into n bins
dplyr::percent_rank() - min_rank scaled to [0,1]
dplyr::row_number() - rank with ties = "first"

MATH

+, **-**, *****, **/**, **^**, **%/%**, **%%** - arithmetic ops
log(), **log2()**, **log10()** - logs
<, **<=**, **>**, **>=**, **!=** - logical comparisons
dplyr::between() - `x >= left & x <= right`
dplyr::near() - safe == for floating point numbers

MISC

dplyr::case_when() - multi-case if_else()
`iris %>% mutate(Species = case_when(Species == "versicolor" ~ "vers", Species == "virginica" ~ "virg", TRUE ~ Species))`

dplyr::coalesce() - first non-NA values by element across a set of vectors
dplyr::if_else() - element-wise if() + else()
dplyr::na_if() - replace specific values with NA
dplyr::pmax() - element-wise max()
dplyr::pmin() - element-wise min()
dplyr::recode() - Vectorized switch()
dplyr::recode_factor() - Vectorized switch() for factors

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNTS

dplyr::n() - number of values/rows
dplyr::n_distinct() - # of uniques
sum(is.na()) - # of non-NA's

LOCATION

mean() - mean, also **mean(is.na())**
median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

dplyr::first() - first value
dplyr::last() - last value
dplyr::nth() - value in nth location of vector

RANK

quantile() - nth quantile
min() - minimum value
max() - maximum value

SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

rownames_to_column()
Move row names into col.
`a <- rownames_to_column(iris, var = "C")`

column_to_rownames()
Move col in row names.
`column_to_rownames(a, var = "C")`

Also **has_rownames()**, **remove_rownames()**

Combine Tables

COMBINE VARIABLES



Use **bind_cols()** to paste tables side by side as they are.

bind_cols(...) Returns tables placed side by side as a single table. BE SURE THAT ROWS ALIGN.

Use a "Mutating Join" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)
Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)
Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)
Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)
Join data. Retain all values, all rows.

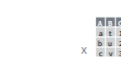
semi_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)
Join data. Retain only rows that have a match in y.

Use by = c("col1", "col2", ...) to specify one or more common columns to match on.
`left_join(x, y, by = "A")`

Use a named vector, **by = c("col1" = "col2")**, to match on columns that have different names in each table.
`left_join(x, y, by = c("C" = "D"))`

Use **suffix** to specify the suffix to give to unmatched columns that have the same name in both tables.
`left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))`

COMBINE CASES



Use **bind_rows()** to paste tables below each other as they are.

bind_rows(..., id = NULL)
Returns tables one on top of the other as a single table. Set **id** to a column name to add a column of the original table names (as pictured)

intersect(x, y, ...)
Rows that appear in both x and y.

setdiff(x, y, ...)
Rows that appear in x but not y.

union(x, y, ...)
Rows that appear in x or y. (Duplicates removed). **union_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

EXTRACT ROWS



Use a "Filtering Join" to filter one table against the rows of another.

semi_join(x, y, by = NULL, ...)
Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

anti_join(x, y, by = NULL, ...)
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.



RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tidbtle")) • dplyr 0.7.0 • tidbtle 1.2.0 • Updated: 2019-08



RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tidbtle")) • dplyr 0.7.0 • tidbtle 1.2.0 • Updated: 2019-08



<https://www.meetup.com/Lucerne-R-User-Group/>

dplyr general syntax

1. Call any **dplyr::function()**
2. First argument is a **data frame (tibble)**
3. Arguments describing what to do using **variable names** (no quotes)
4. Result is a **data frame (tibble)**

function(**data**, ...)

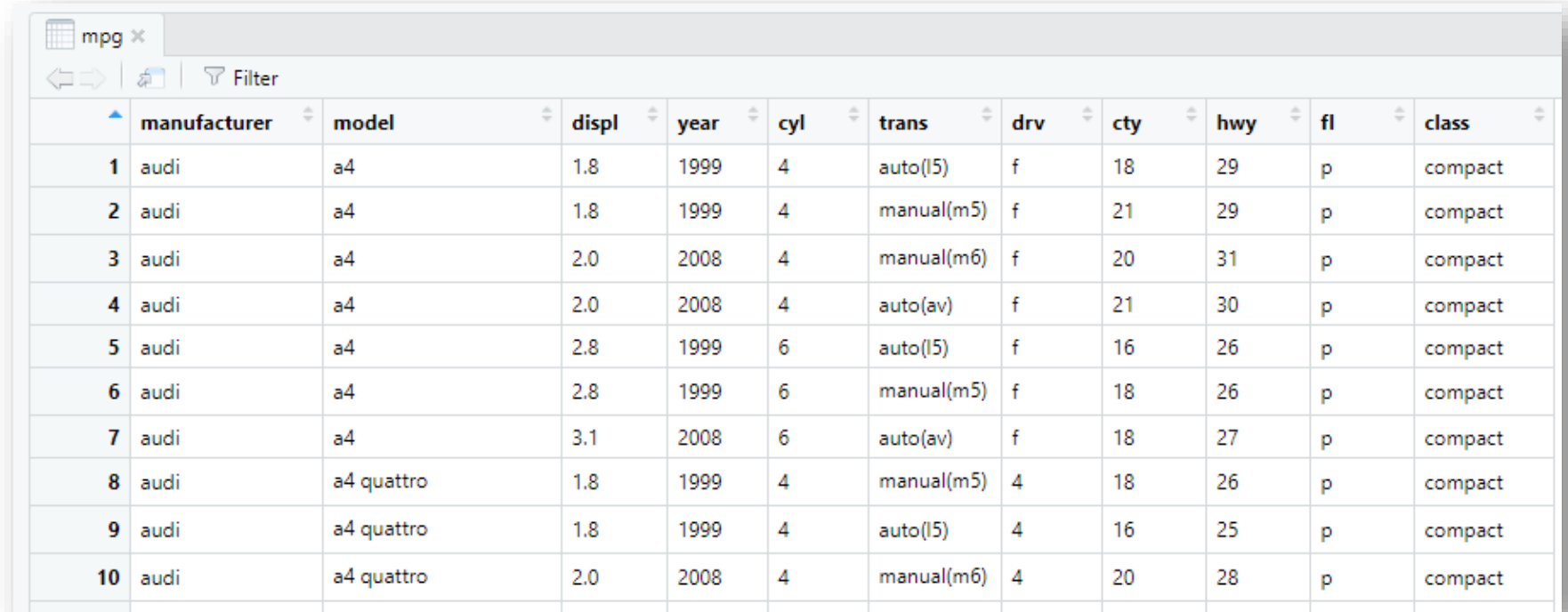
↑ ↑
data **function-specific
argument**



mpg data

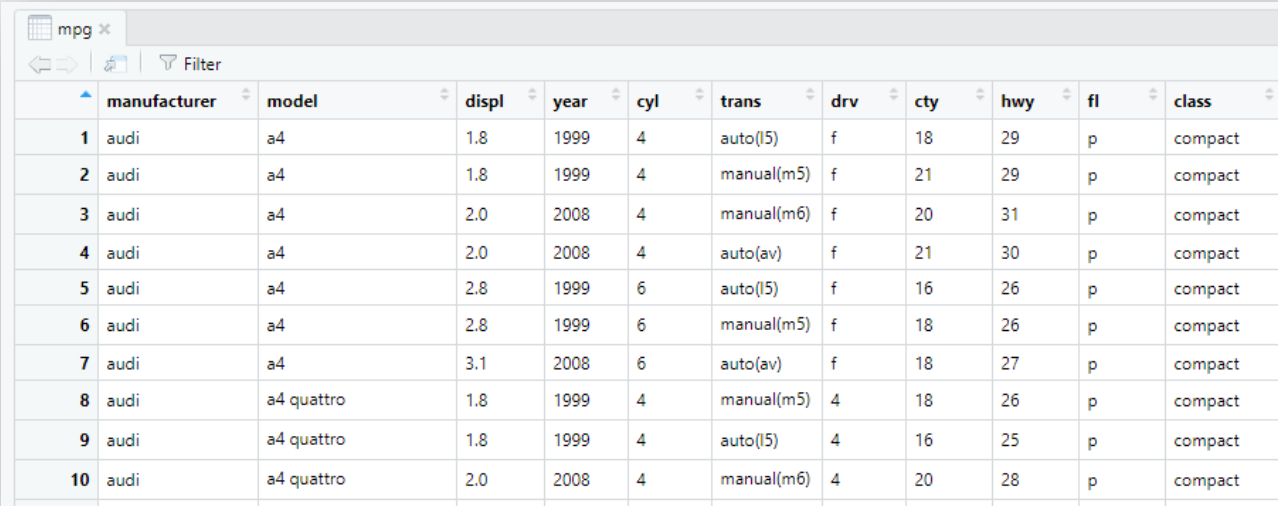
Fuel economy data for 38 models of cars

View (mpg)



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

Your Turn



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that you want
to **keep only variables**
displ and **hwy**?

Your Turn

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that you want
to **keep only variables**
displ and **hwy**?

A

`filter`

B

`mutate`

C

`arrange`

D

`select`

00:15

Your Turn

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that we want
to **keep only variables**
displ and **hwy**?



A

`filter`

B

`mutate`



C

`arrange`

D

`select`



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

```
mpg_rec <- select(mpg, displ, hwy)
```

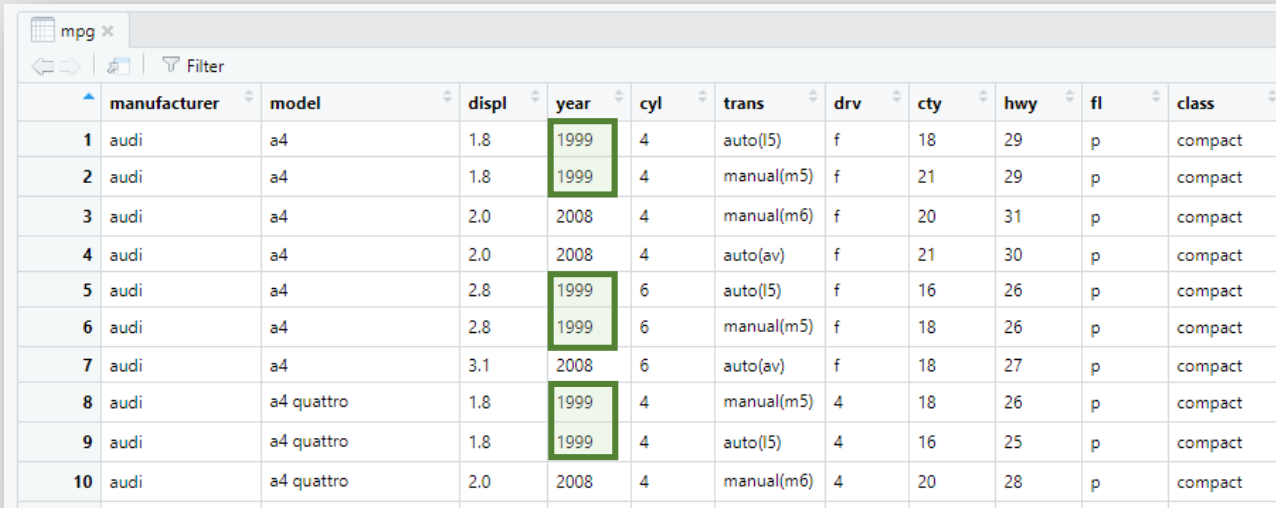
↑
new data

↑
data

↑
variables

	displ	hwy
1	1.8	29
2	1.8	29
3	2.0	31
4	2.0	30
5	2.8	26
6	2.8	26
7	3.1	27
8	1.8	26
9	1.8	25
10	2.0	28

Your Turn



mpg ✕

Filter

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that you want
to **keep only certain**
observations?
(e.g. cars from 1999)

Your Turn

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that you want
to **keep only certain**
observations?
(e.g. cars from 1999)

A

`filter`

B

`mutate`

C

`arrange`

D

`select`

00:15

Your Turn

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

How do you say that you want
to **keep only certain**
observations?
(e.g. cars from 1999)



A

`filter`

B

`mutate`



C

`arrange`

D

`select`



00:15

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact

```
mpg_rec <- filter(mpg, year==1999)
```

↑
new data

↑
data

↑
filtering
condition

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
4	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
5	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
6	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
7	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
8	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
9	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
10	chevrolet	c1500 suburban 2wd	5.7	1999	8	auto(l4)	r	13	17	r	suv

Questions

?

Your Turn

What about **mutate()** and **arrange()**?

Check out the code in ``01_introduction.R``
and try to find it out.

Your Turn

Open the file ``02_markdown_piping.R`` to learn about R Markdown and piping.



Introduction to tidyverse



<https://www.meetup.com/Lucerne-R-User-Group/>



https://github.com/Lucerne-R-User-Group/2020_06_24-inaugural-meeting

Dr Andrea De Angelis



24 June 2020