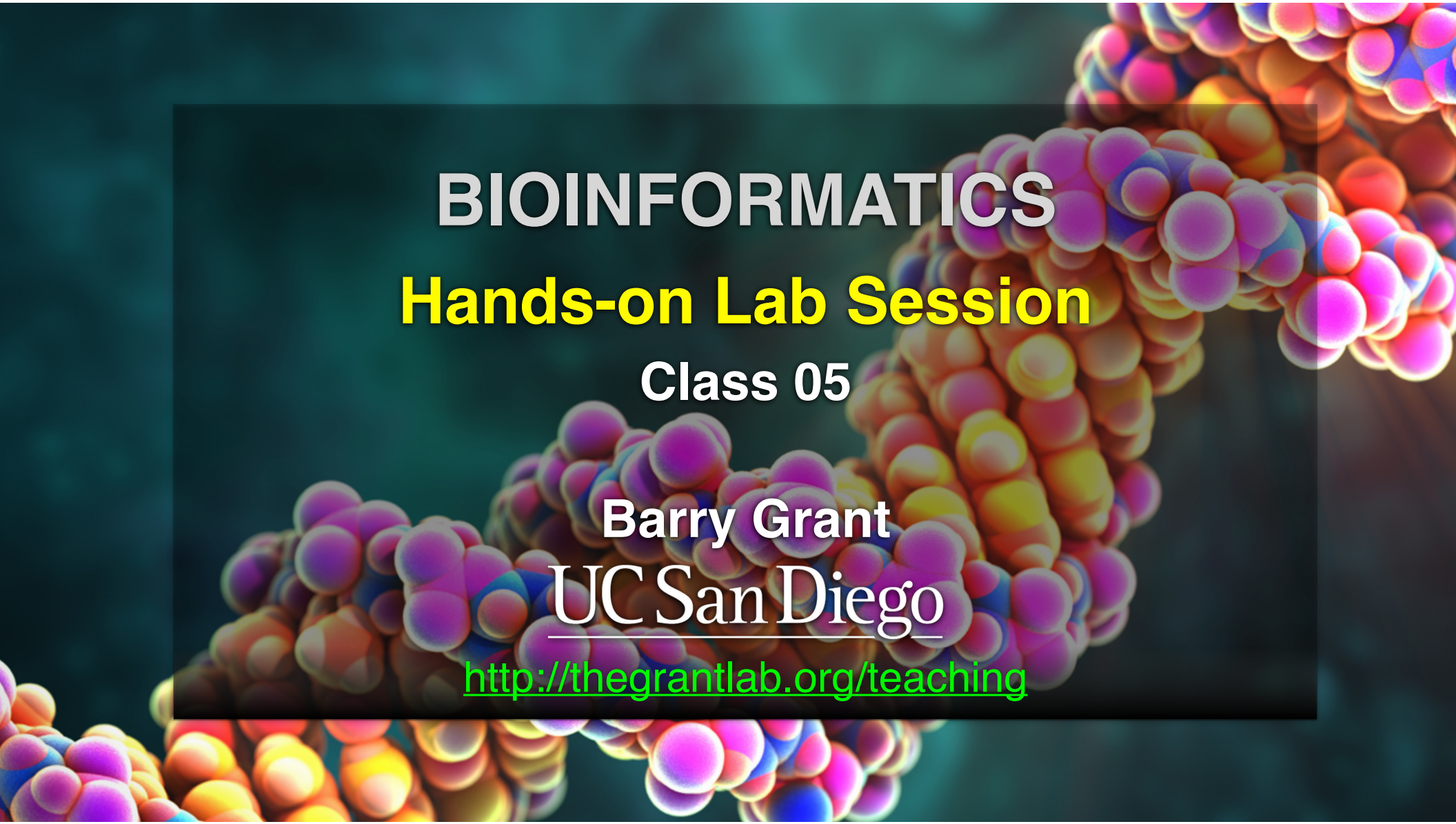


BIOINFORMATICS

Hands-on Lab Session

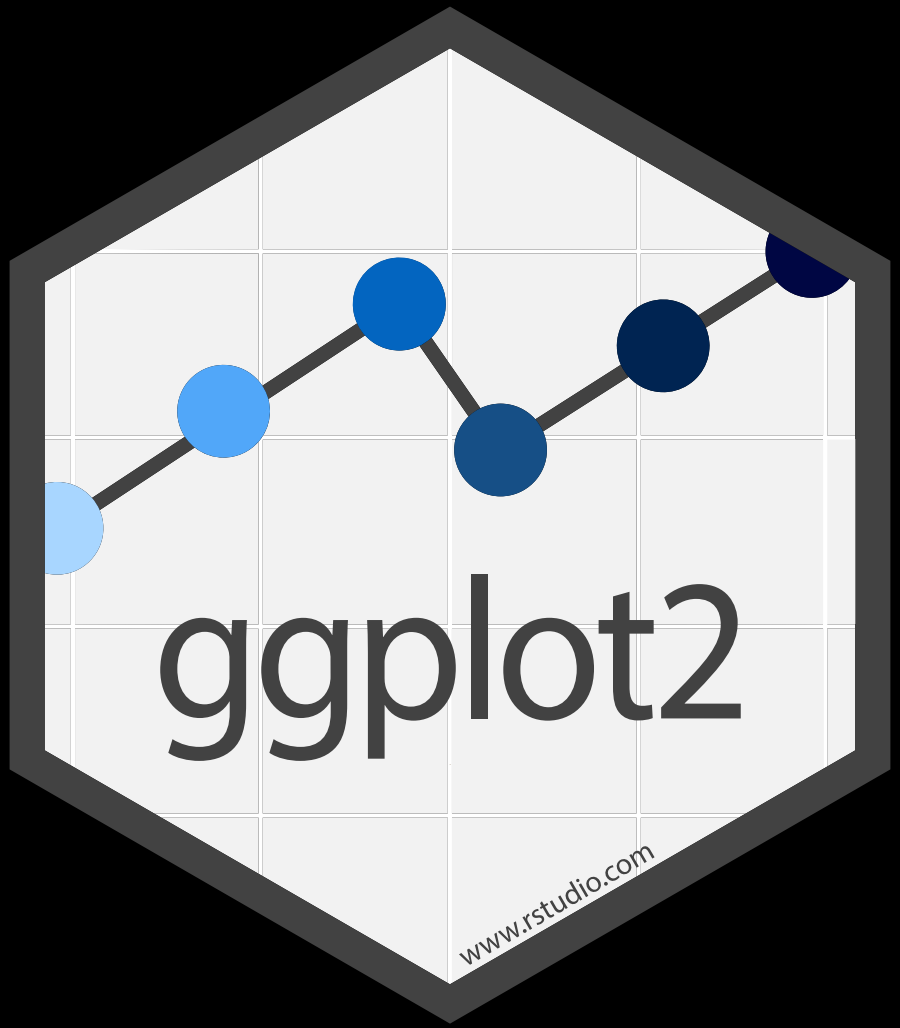
Class 05

Barry Grant

UC San Diego

http://thegrantlab.org/teaching

How do we make informative
and compelling figures?

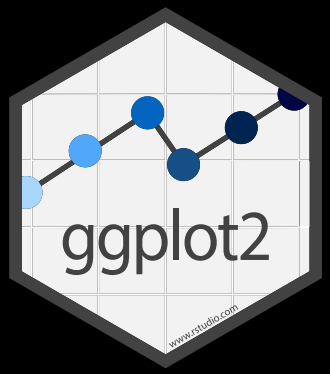

ggplot2
www.rstudio.com


ggplot2
Currently the premier plotting
library on the planet!

Key Insight: All visualizations map data into quantifiable aesthetic features of the resulting graphic

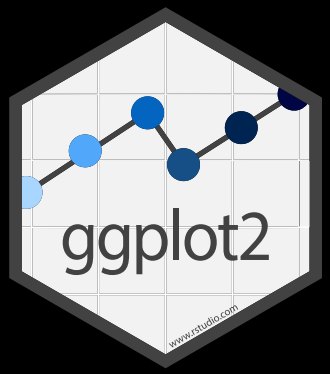
Key Insight: All visualizations map data into quantifiable aesthetics features of the resulting graphic


data
aesthetics



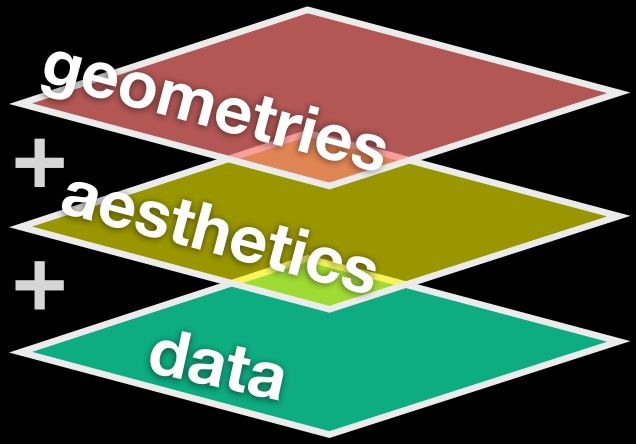
data + **aes**thetics + **geom**etrys

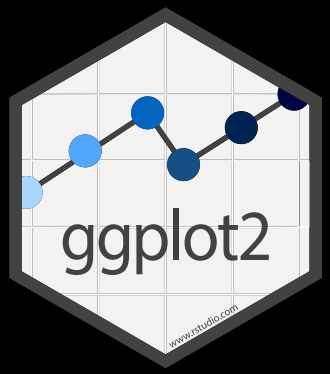
Three main "**layers**"
that are in every ggplot



data + aesthetics + geometrys

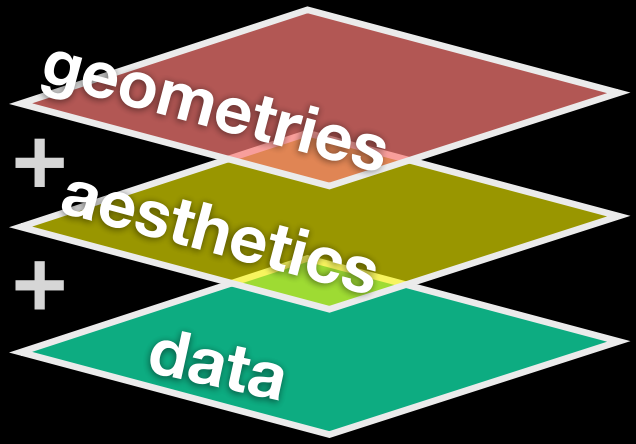
Three main "**layers**" that are in every ggplot

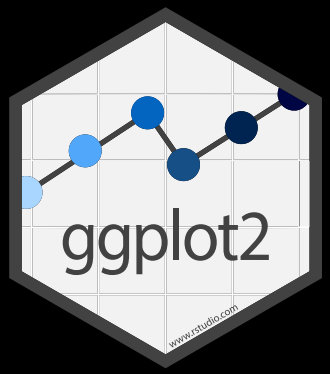




data + aesthetics + geometrys

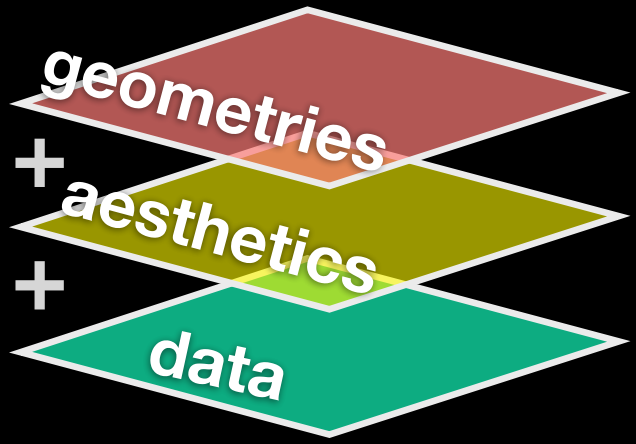
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy, color=class) +
  geom_point()
```



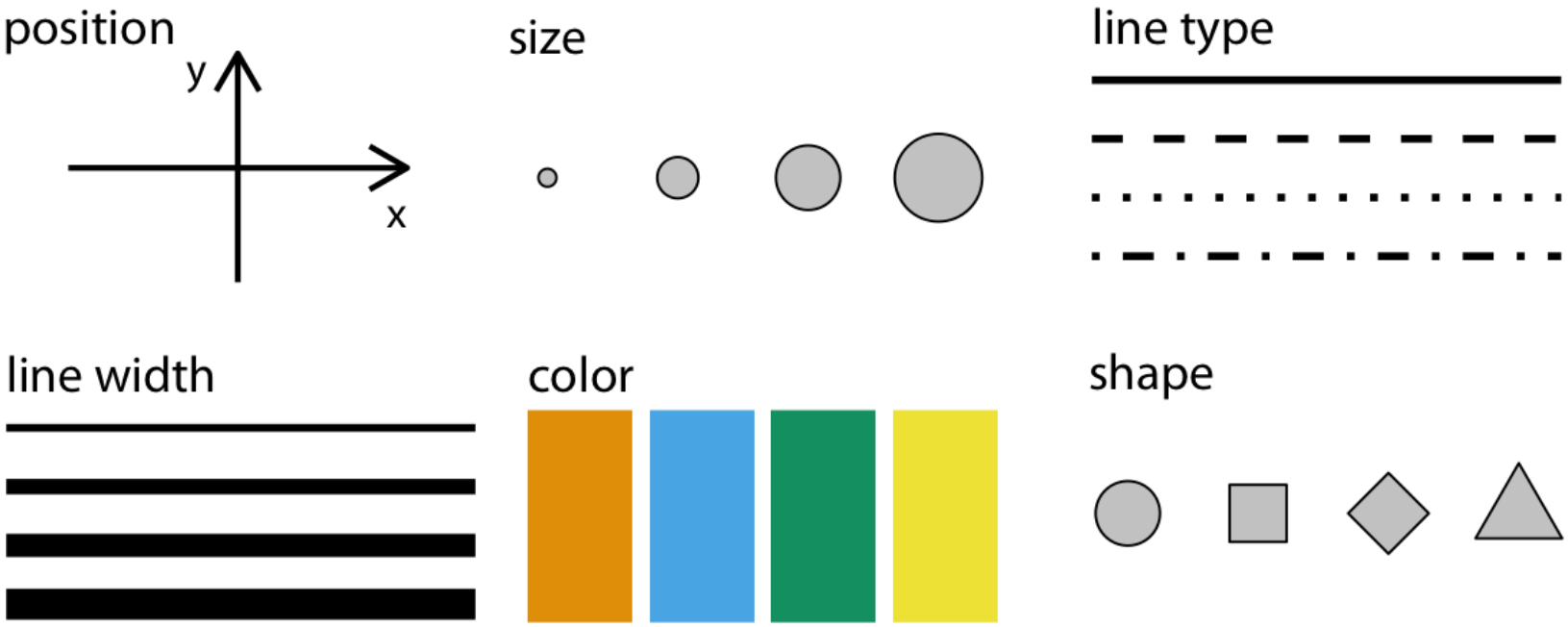


data + aesthetics + geometrys

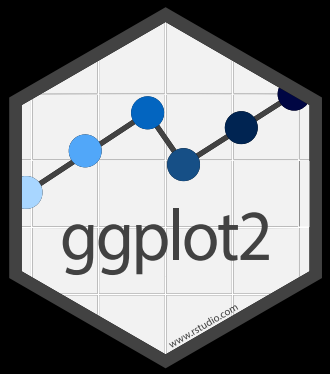
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy, color=class) +
  geom_point()
```



Common aesthetics include

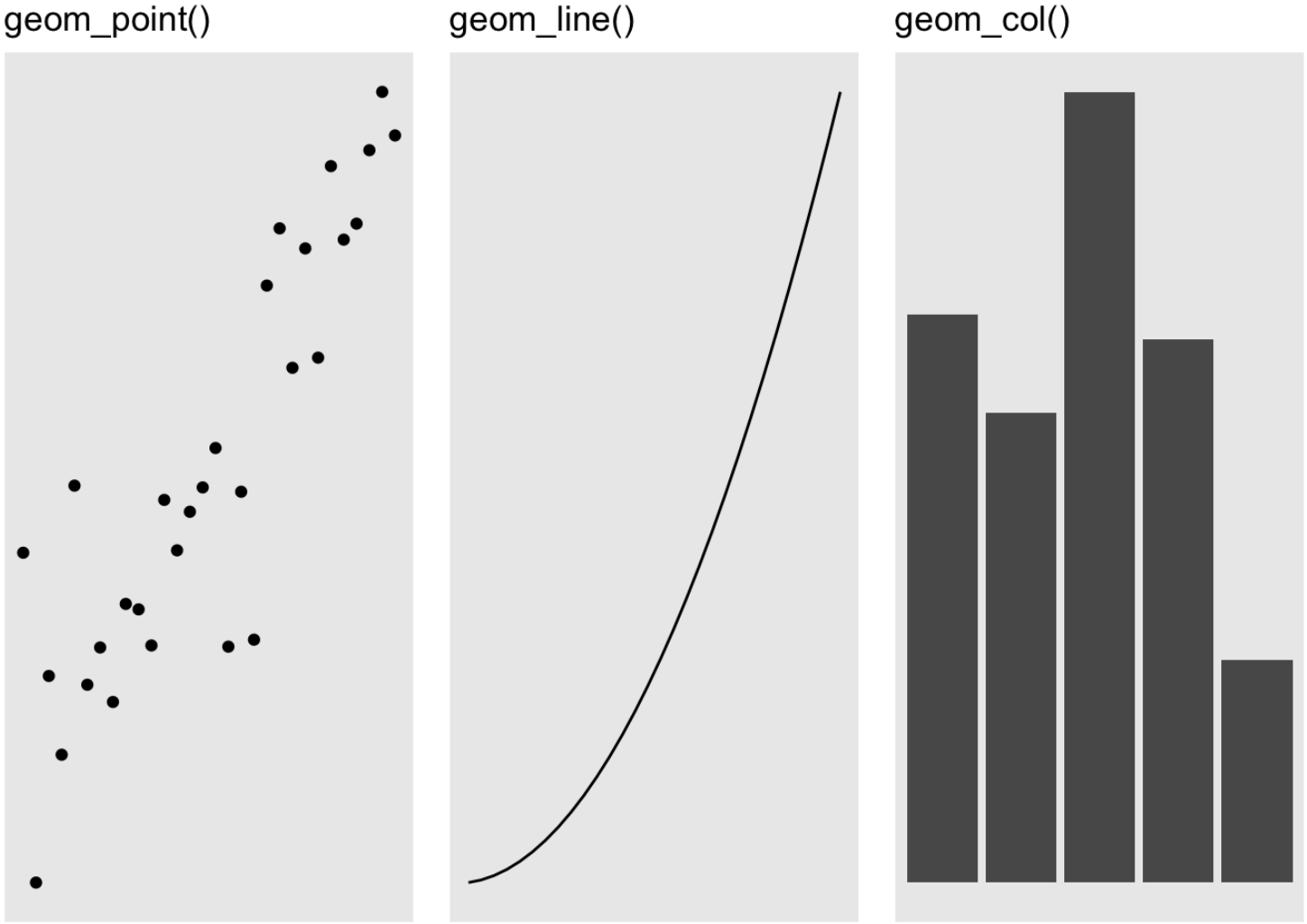


Modified from: Wilke (2019)



data + **aes**thetics + **geom**etrys

Three main "**layers**"
that are in every ggplot


geom_point()
geom_line()
geom_col()

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 = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, 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' x 5' 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(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

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

a + geom_path(lineend="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, ymax, ymin, 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(yend=lat+1, xend=long+1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

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, linetype, size, weight

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

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

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

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

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

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

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map**)**
+ expand_limits(x = map$long, y = map$lat**)**, map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

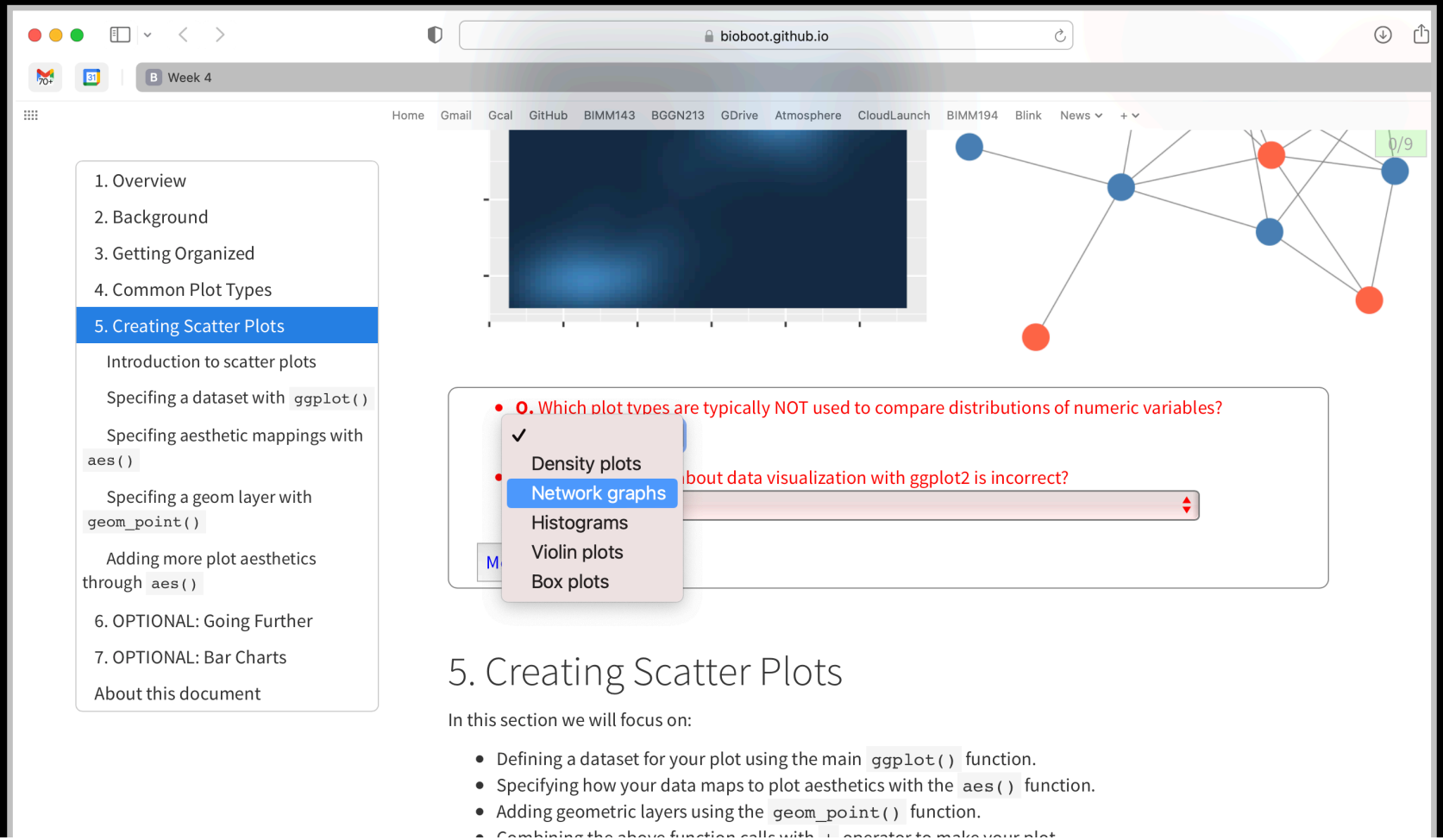
l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

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

Learn more about core `geom_FUNCTIONS()`

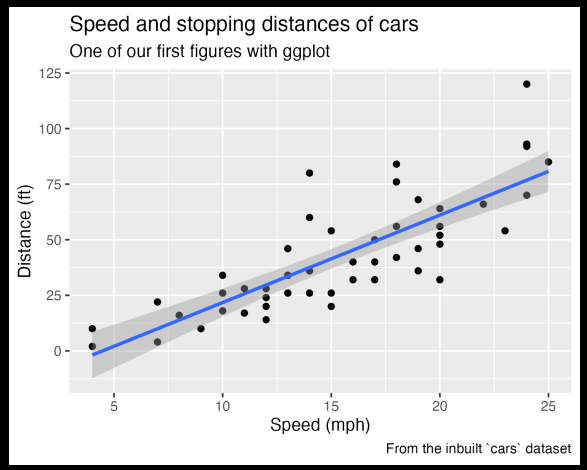
There are > 40 core "geom" functions. See cheat-sheet link on class website!

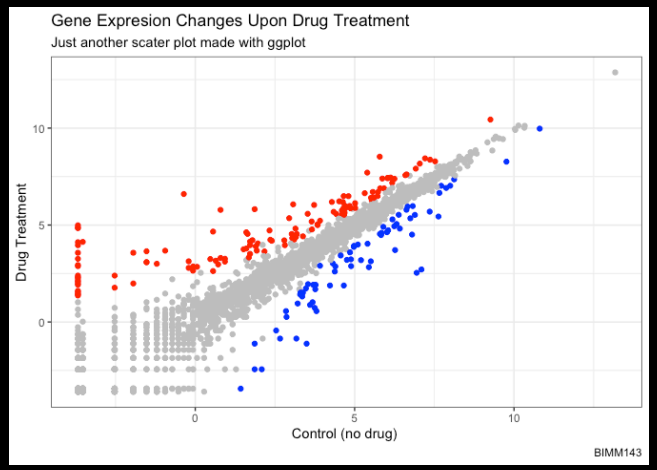
In addition to your **PDF lab report** answer the **inbuilt questions**

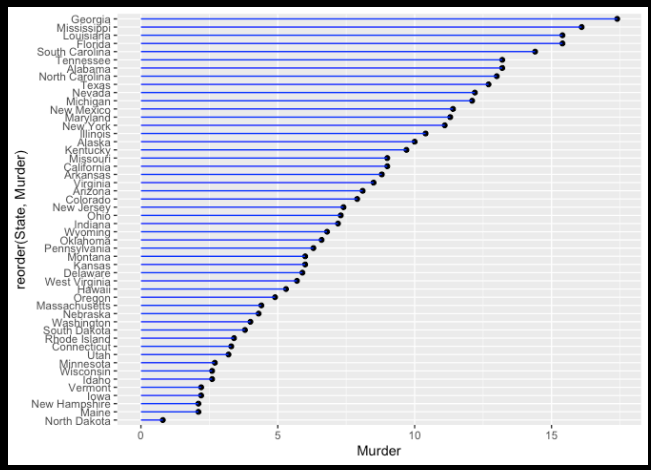


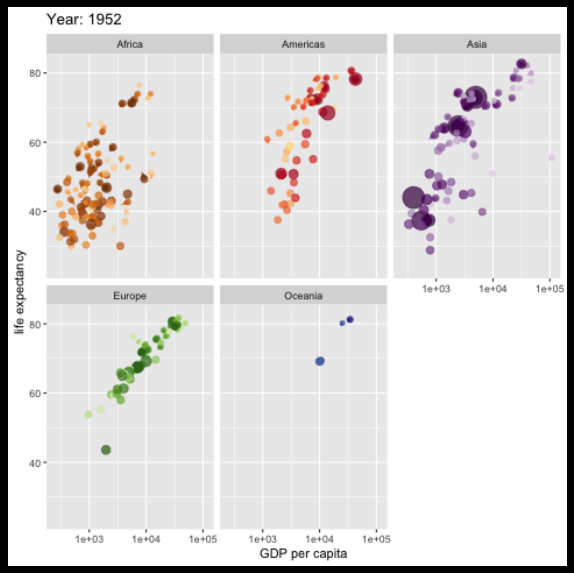
Question Counter

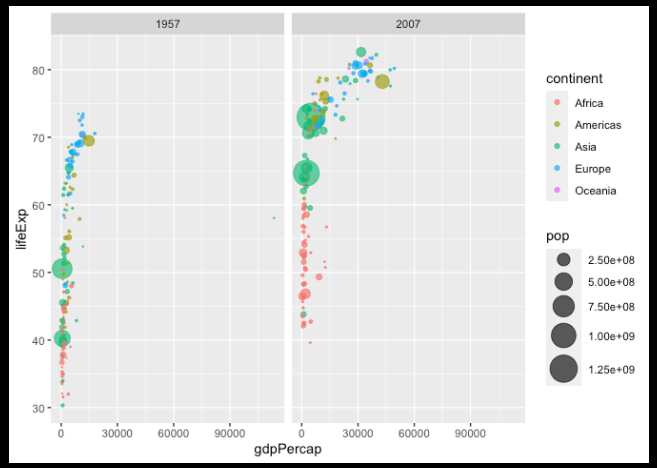
Questions

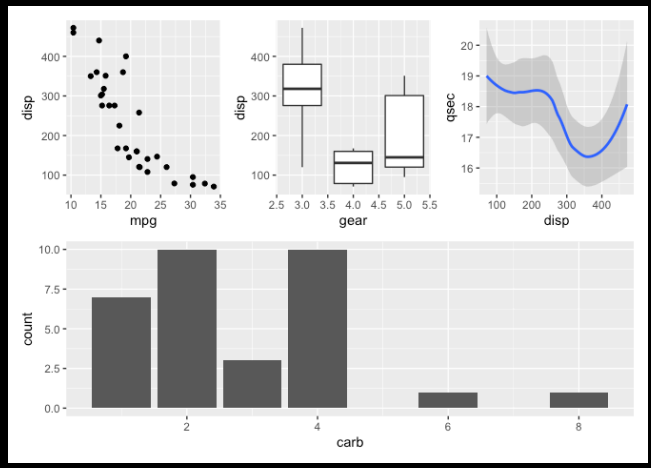

Speed and stopping distances of cars
One of our first figures with ggplot
Distance (ft)
Speed (mph)
From the inbuilt `cars` dataset


Gene Expresion Changes Upon Drug Treatment
Just another scater plot made with ggplot
Drug Treatment
Control (no drug)
BIMM143


reorder(State, Murder)
Murder


Year: 1952
Africa
Americas
Asia
Europe
Oceania
life expectancy
GDP per capita


1957
2007
lifeExp
gdpPercap
continent
Africa
Americas
Asia
Europe
Oceania
pop

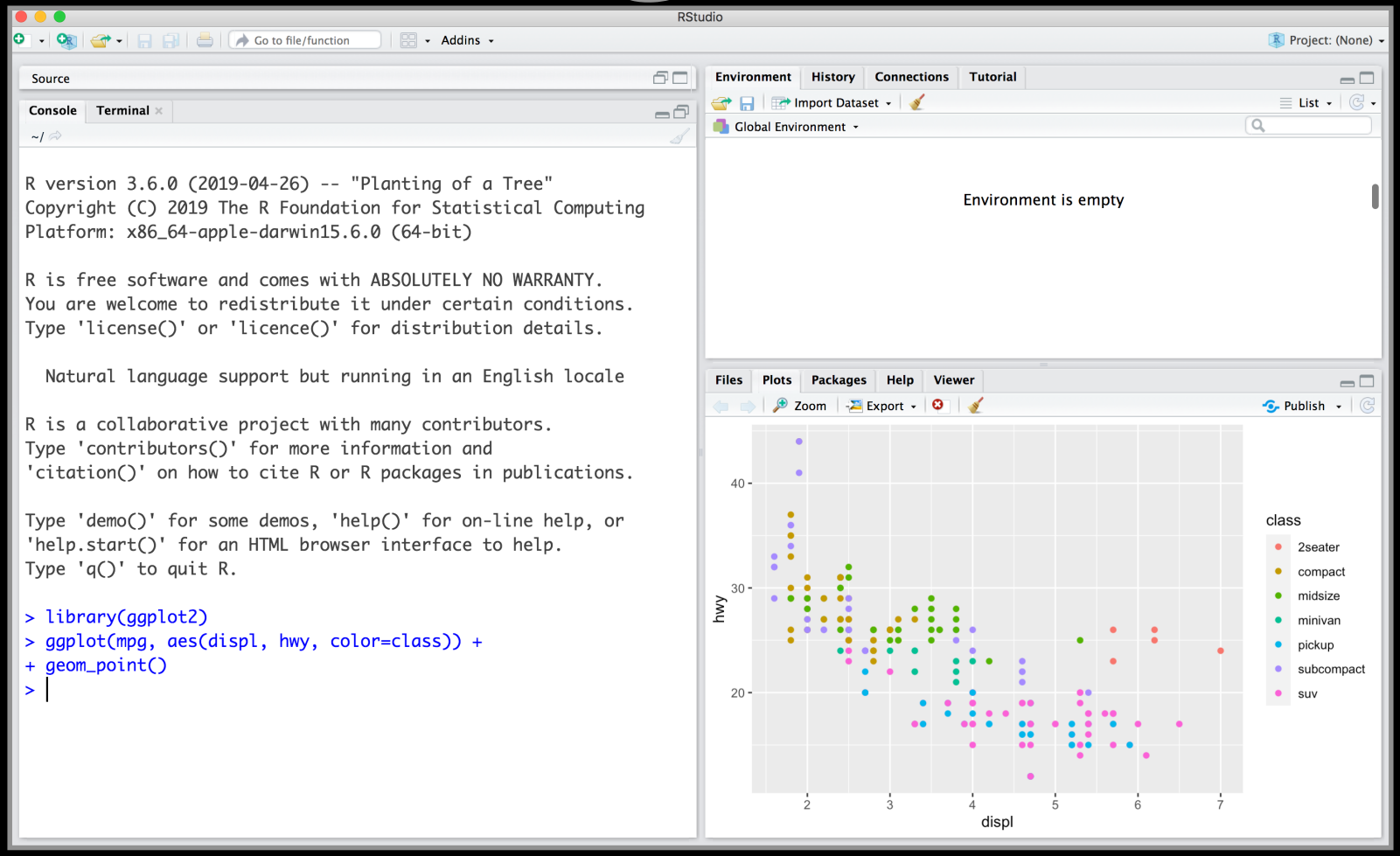

disp
mpg
gear
qsec
count
carb

Follow Along!

Please Open RStudio®

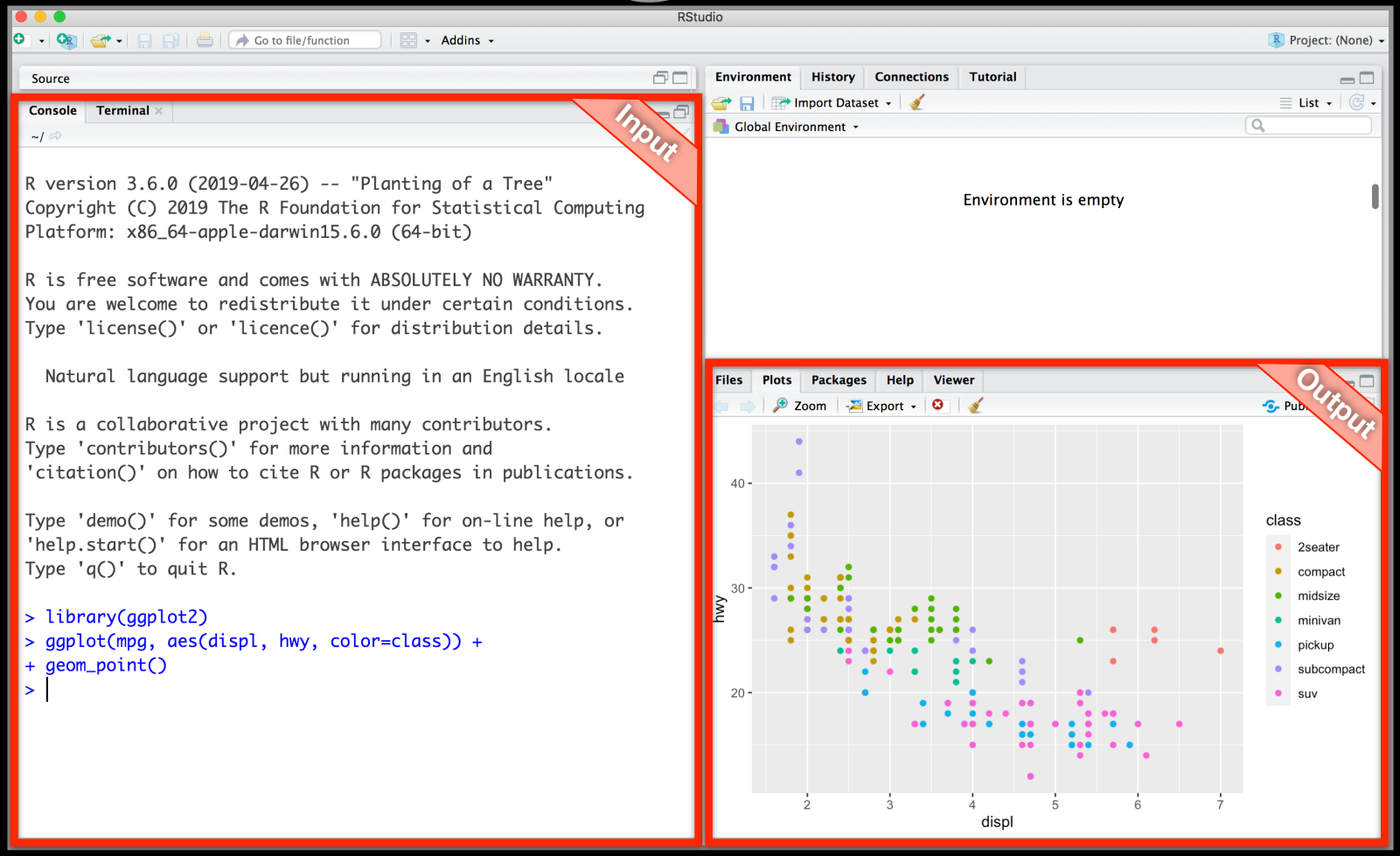
Please Open RStudio®

Follow Along!



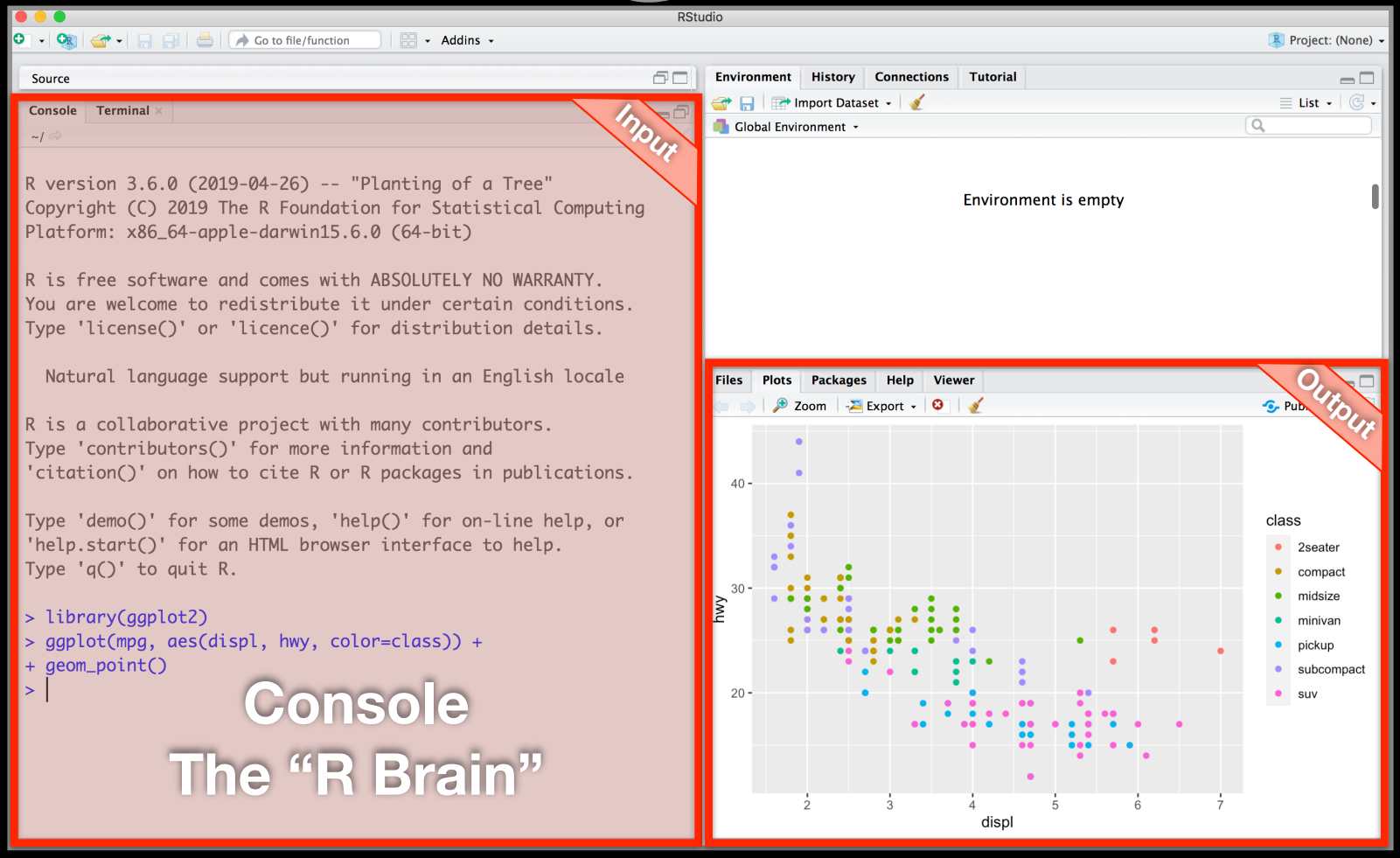
Please Open RStudio®

Follow Along!

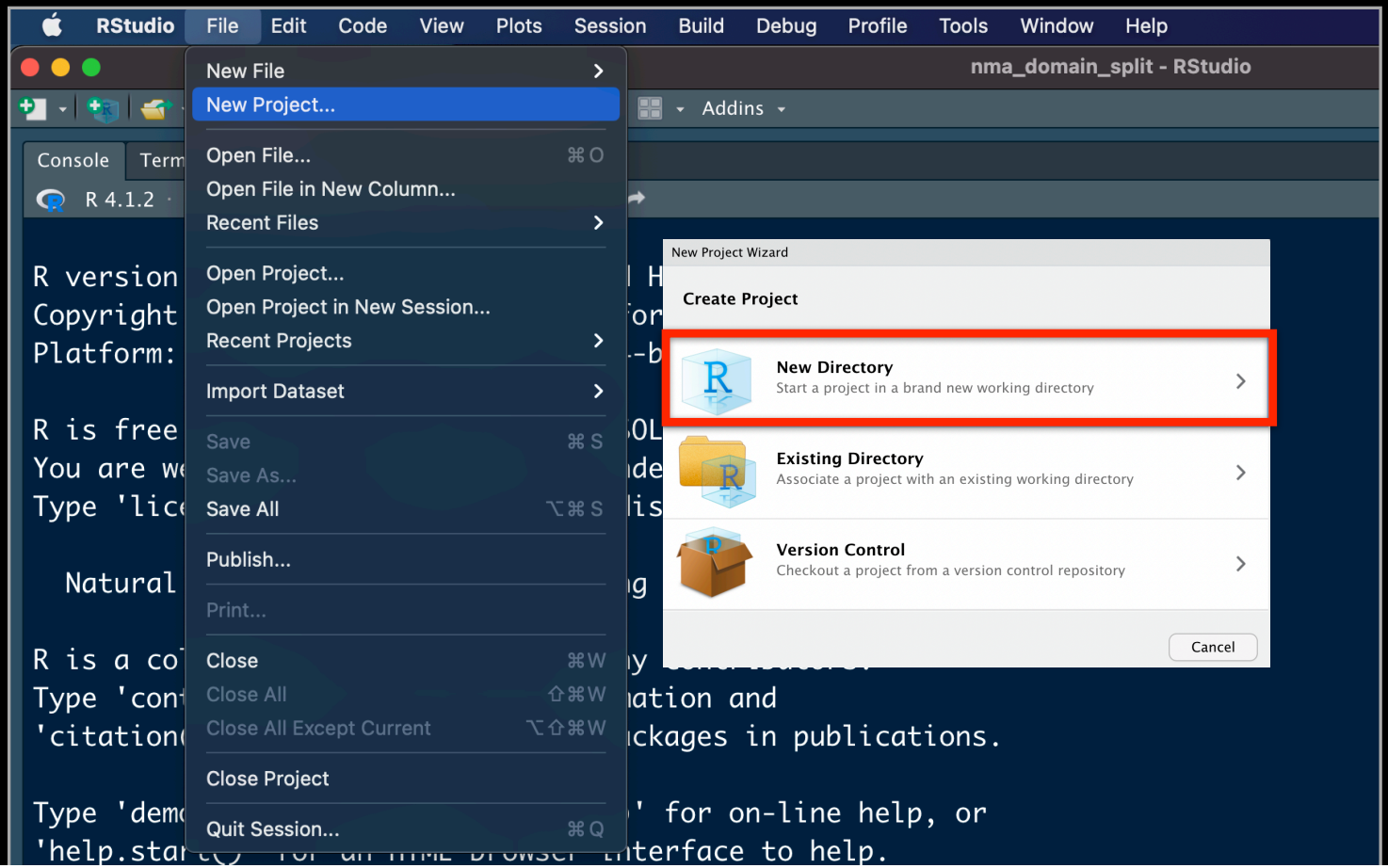


Please Open RStudio®

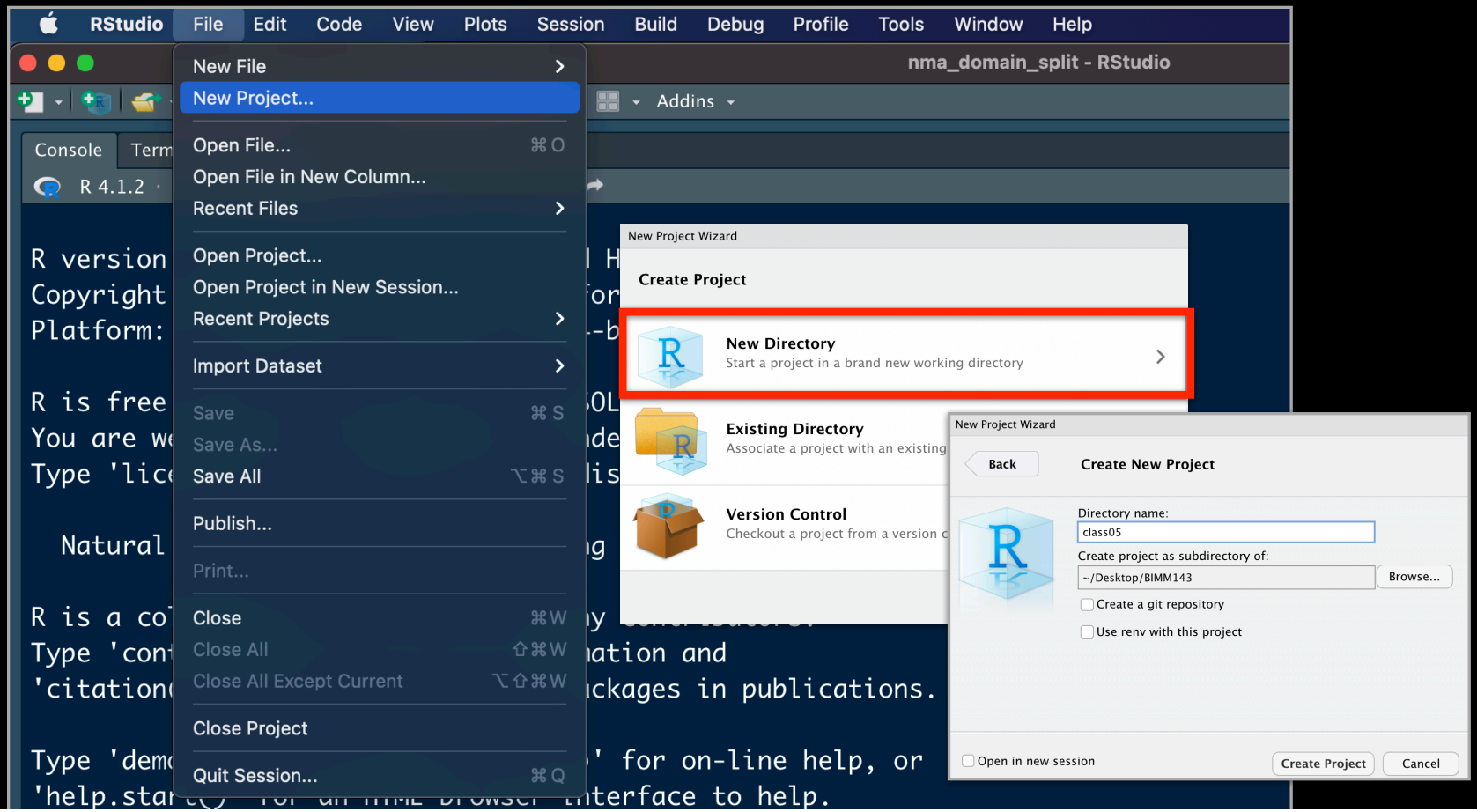
Follow Along!



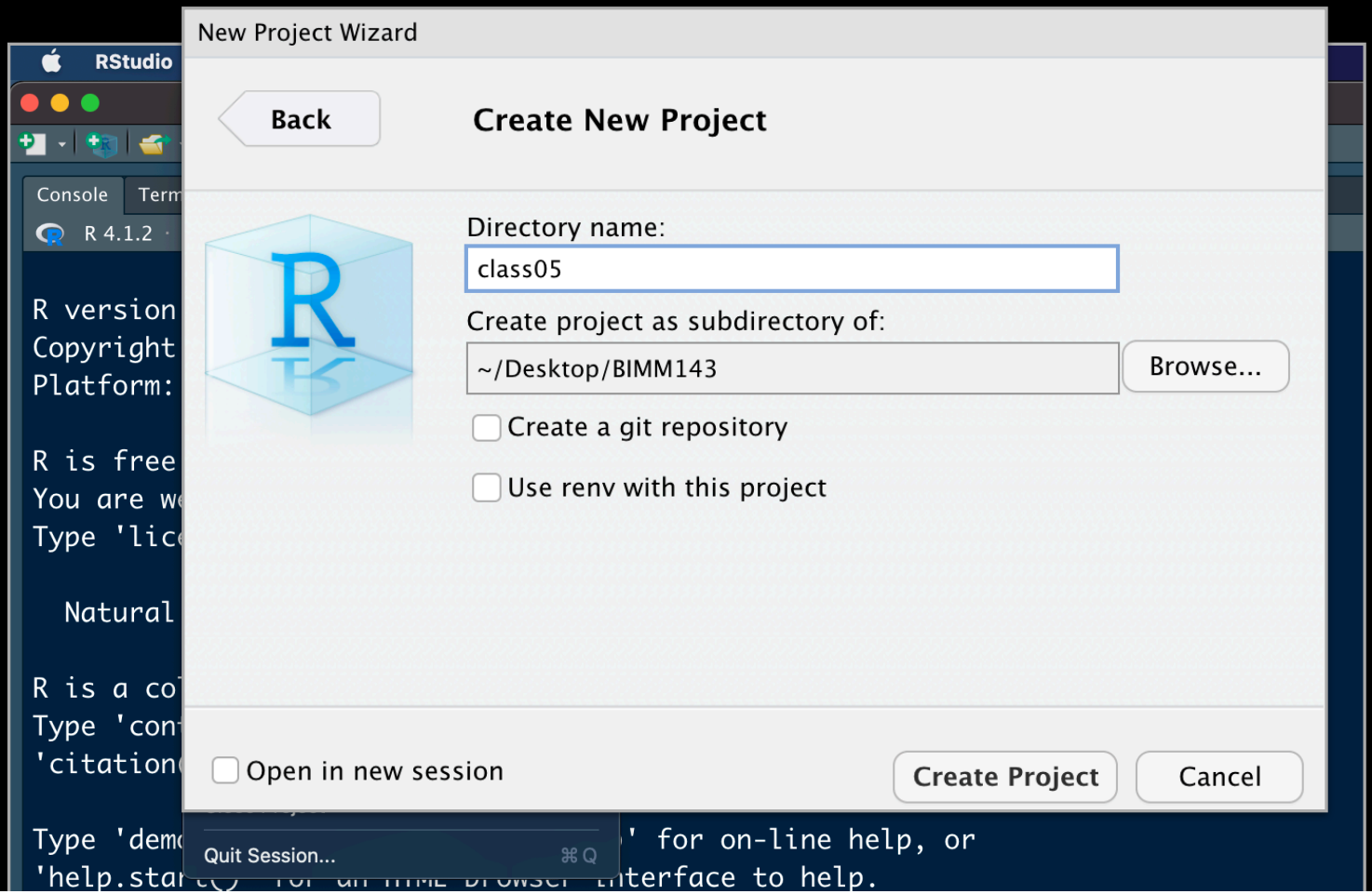
RStudio > File > New Project > New Directory



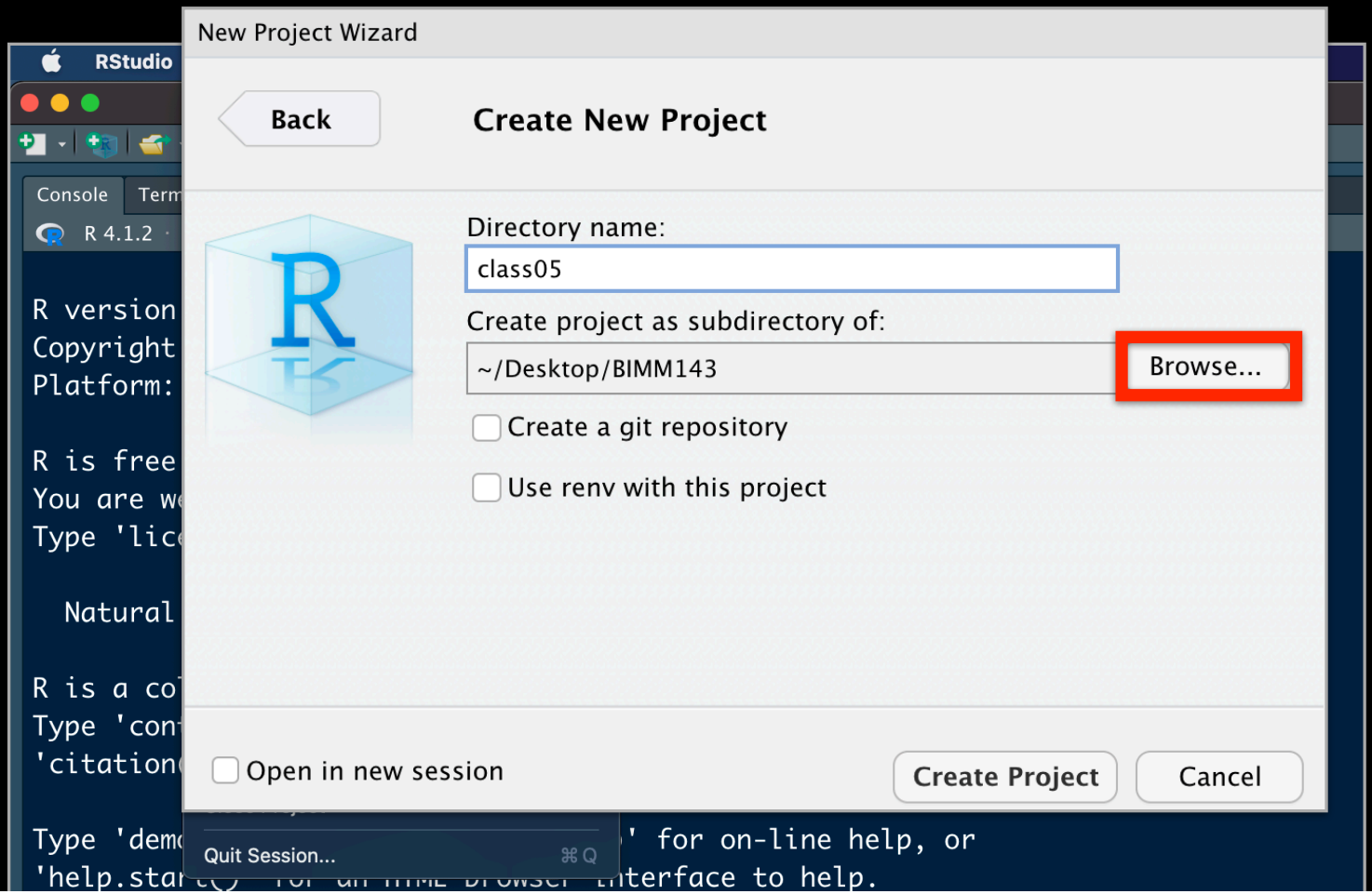
RStudio > File > New Project > New Directory



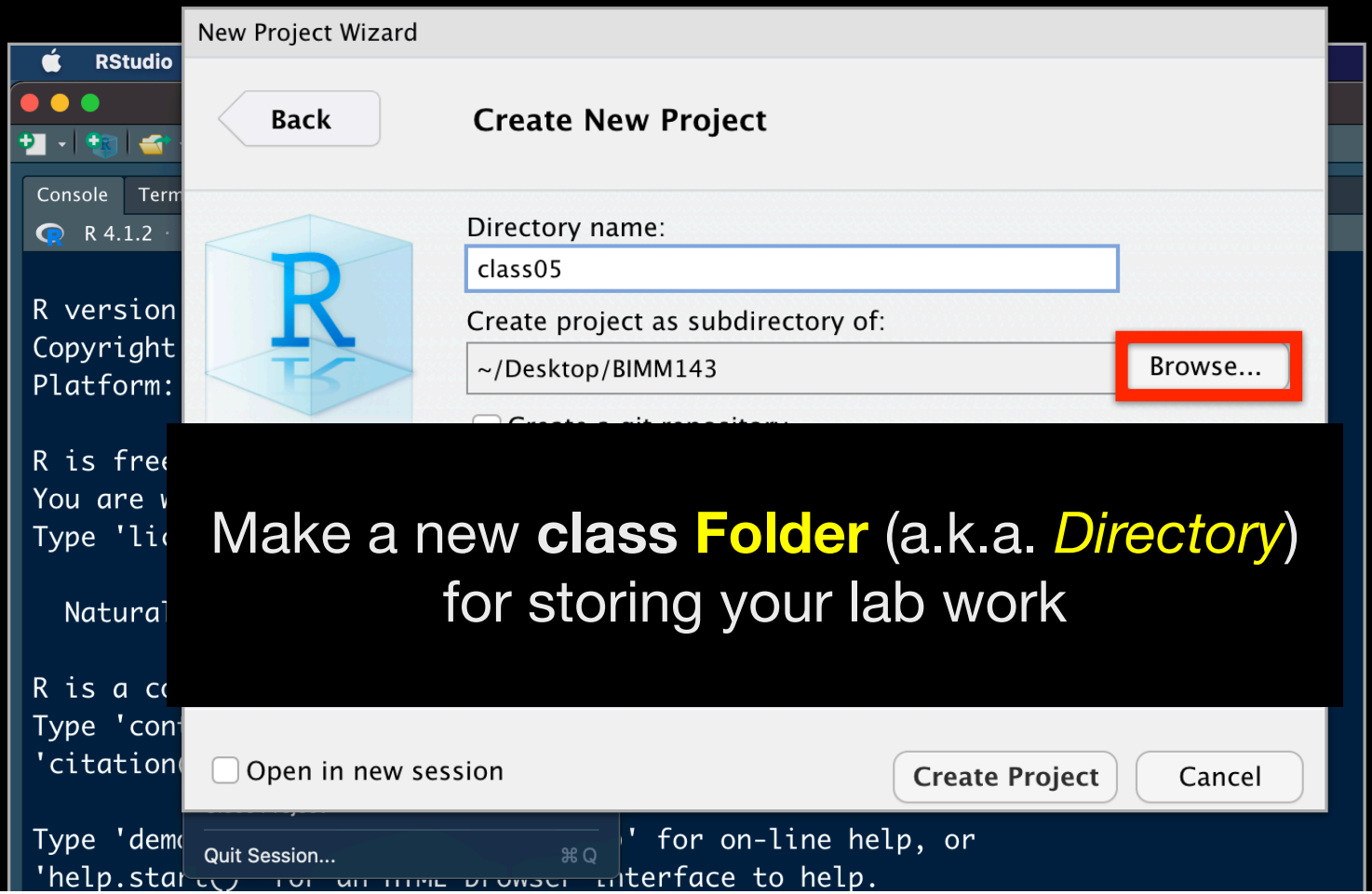
RStudio > File > New Project > New Directory



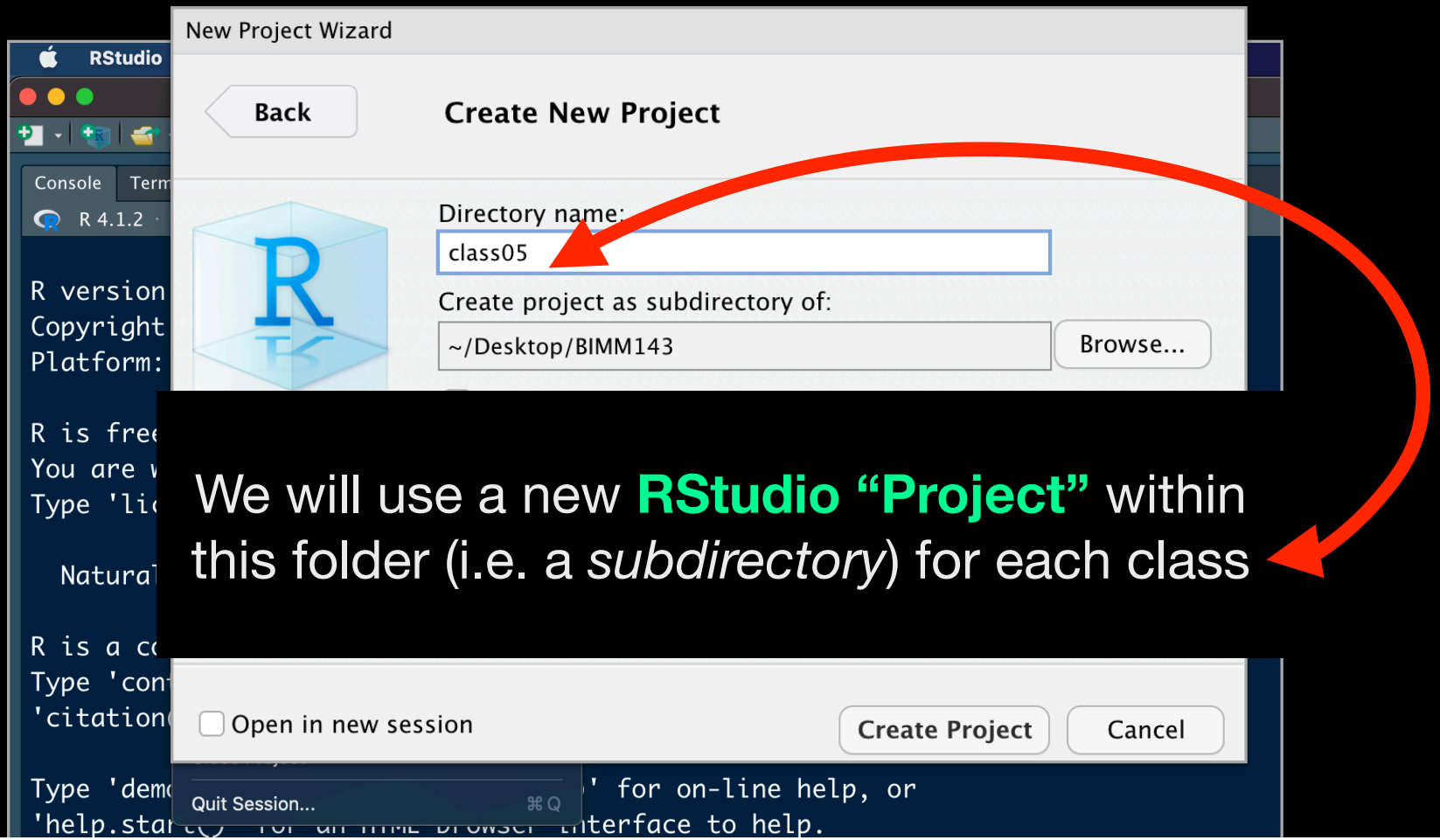
RStudio > File > New Project > New Directory



RStudio > File > New Project > New Directory



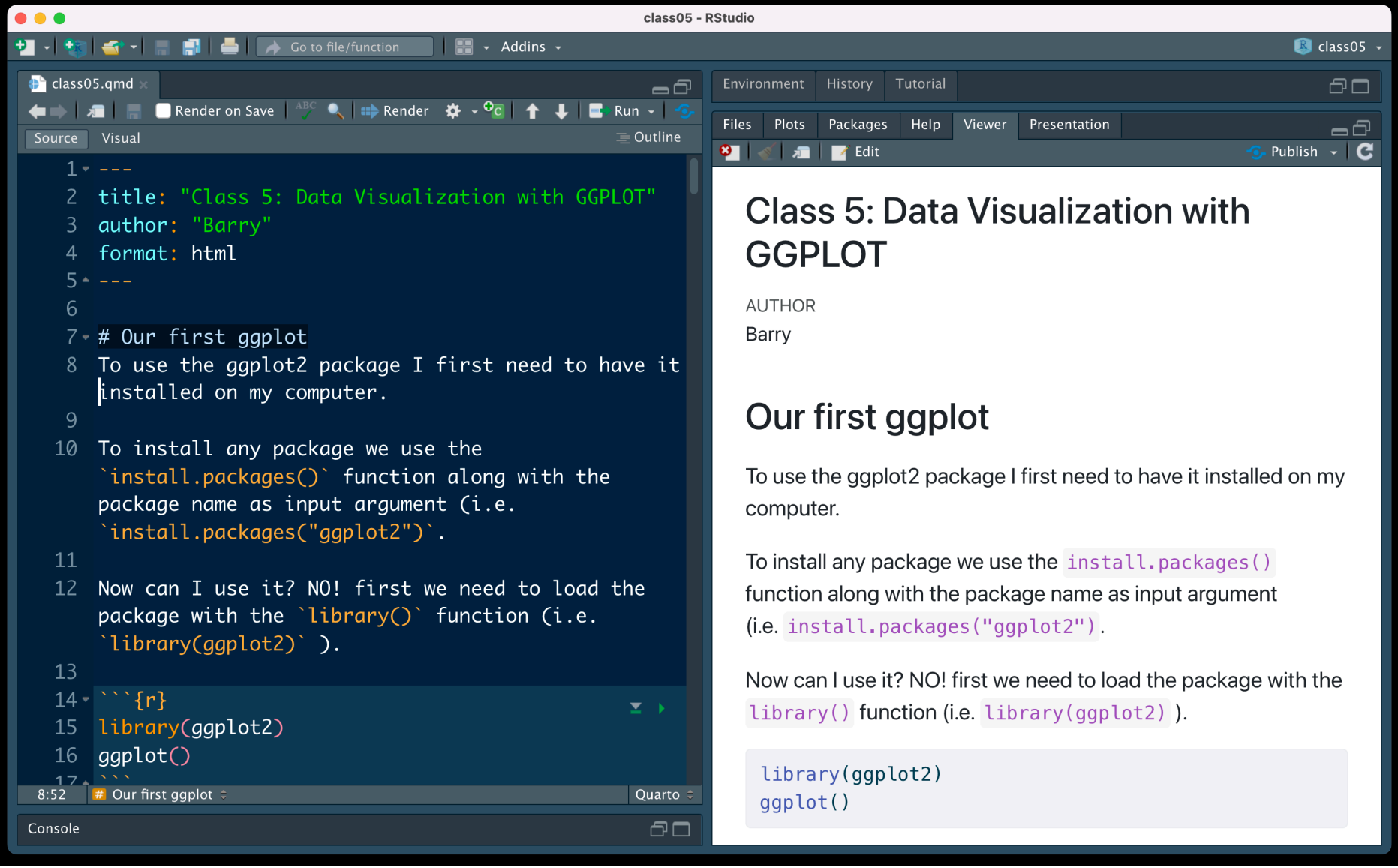
RStudio > File > New Project > New Directory

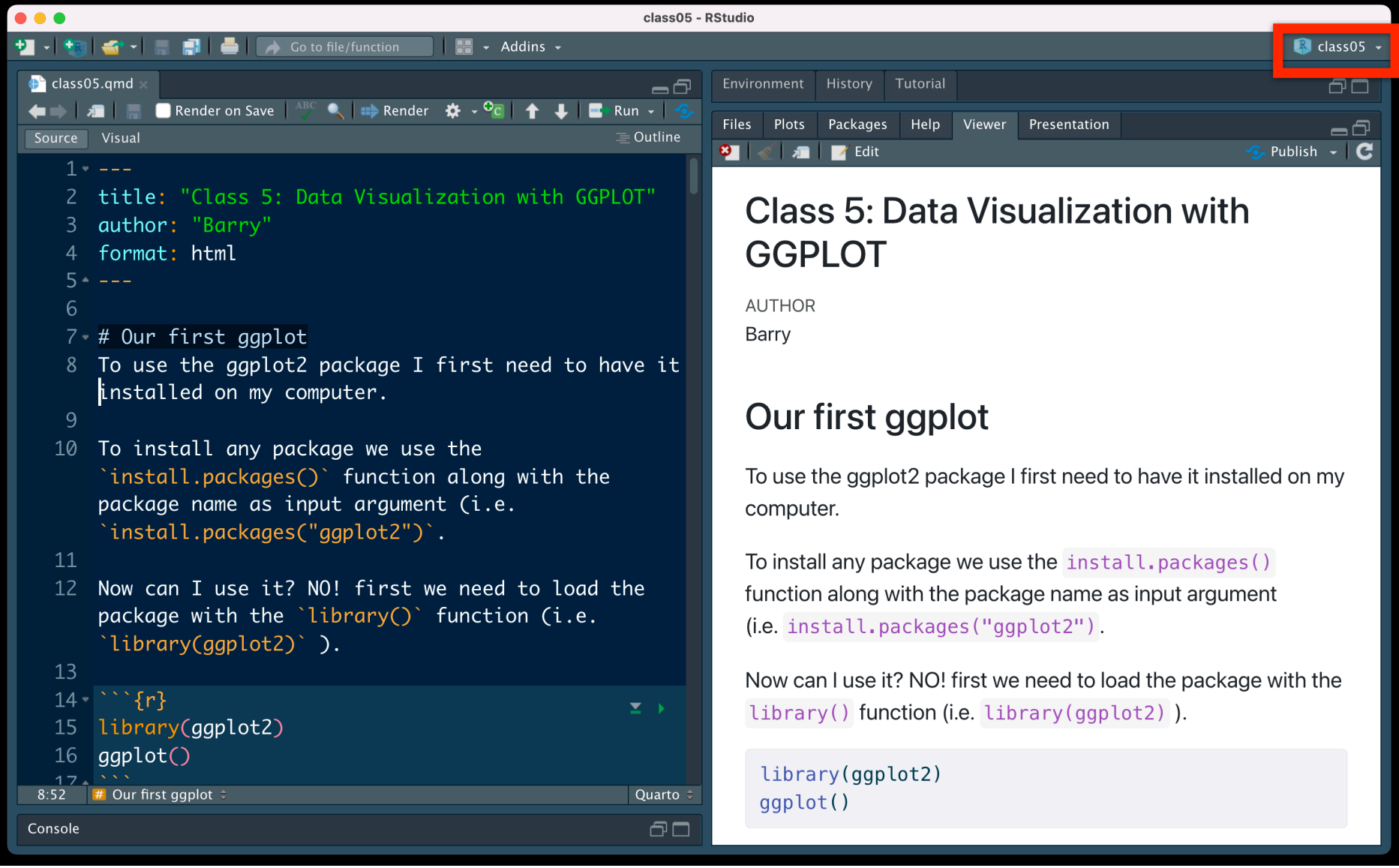


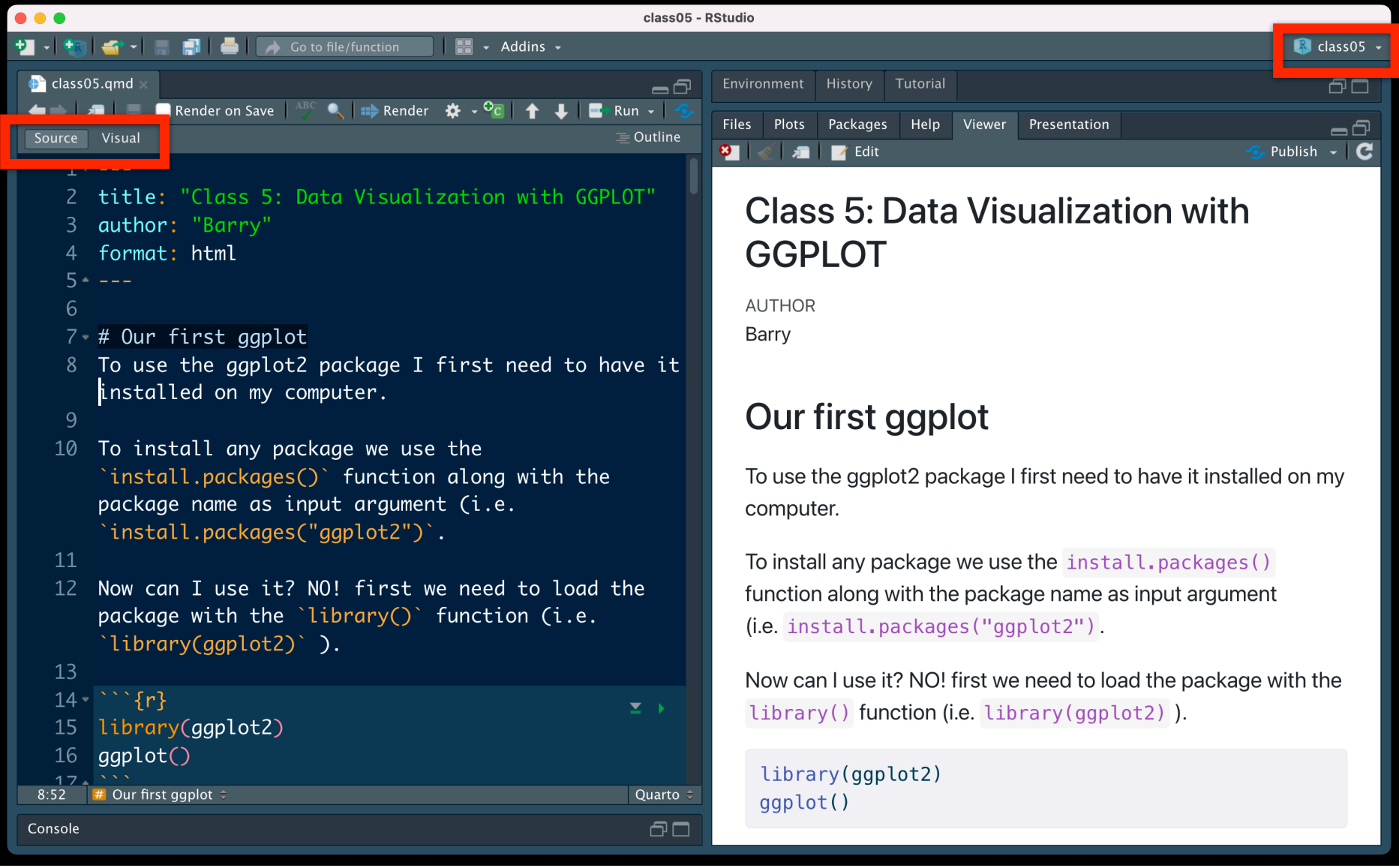
File > New File > **Quarto Document**

File > New File > **Quarto Document**

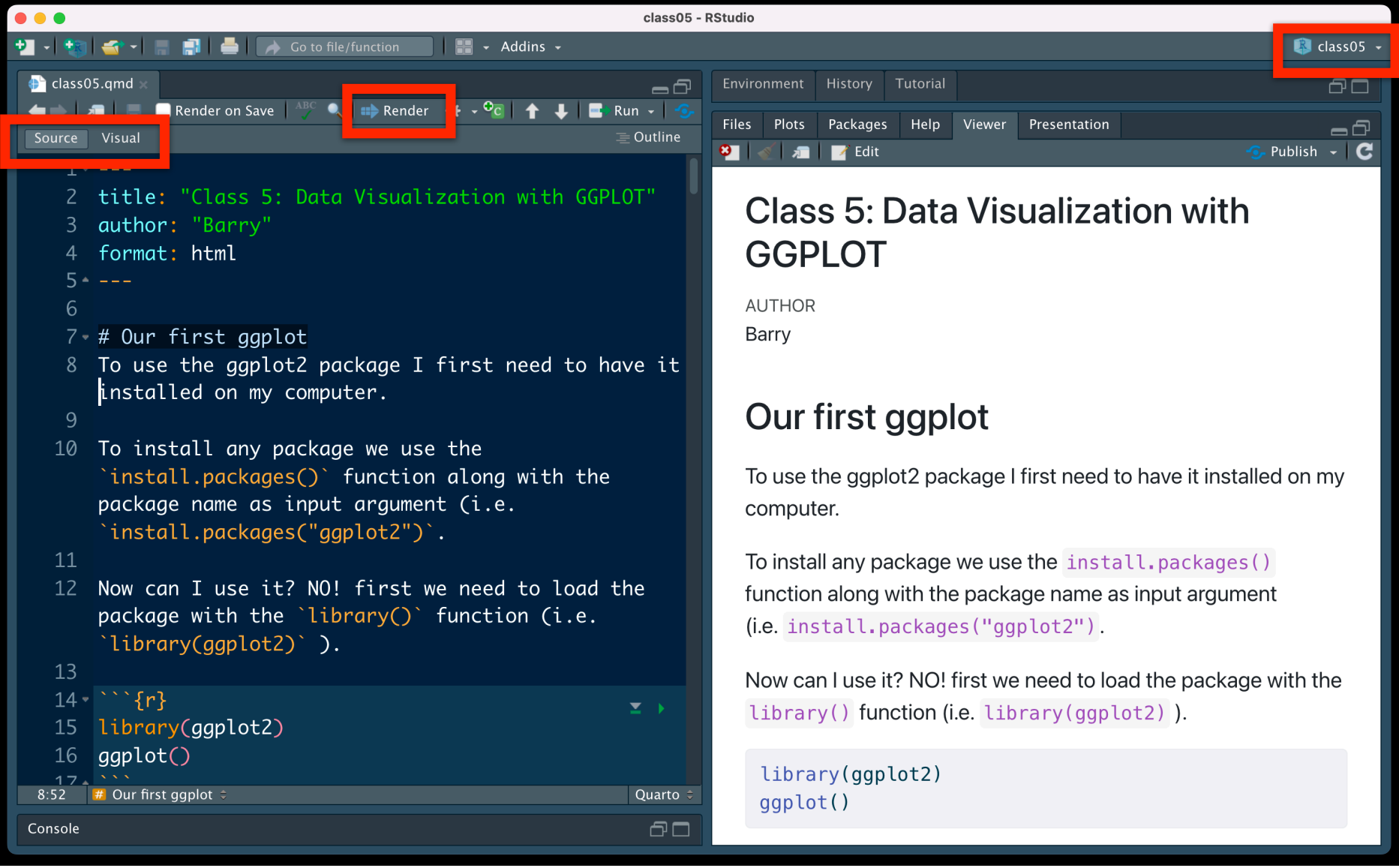
N.B. we will use this to save our work and make a **lab report**


class05 - RStudio
Go to file/function
Addins
class05
class05.qmd
Render on Save
Render
Run
Source
Visual
Outline
1 ---
2 title: "Class 5: Data Visualization with GGPLOT"
3 author: "Barry"
4 format: html
5 ---
6
7 # Our first ggplot
8 To use the ggplot2 package I first need to have it installed on my computer.
9
10 To install any package we use the `install.packages()` function along with the package name as input argument (i.e. `install.packages("ggplot2")`.
11
12 Now can I use it? NO! first we need to load the package with the `library()` function (i.e. `library(ggplot2)`).
13
14 ```{r}
15 library(ggplot2)
16 ggplot()
17 ```
8:52
Our first ggplot
Quarto
Console
Environment
History
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Edit
Publish
Class 5: Data Visualization with GGPLOT
AUTHOR
Barry
Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the install.packages() function along with the package name as input argument (i.e. install.packages("ggplot2").
Now can I use it? NO! first we need to load the package with the library() function (i.e. library(ggplot2)).
library(ggplot2)
ggplot()


class05 - RStudio
Go to file/function
Addins
class05
class05.qmd
Render on Save
Render
Run
Source
Visual
Outline
1 ---
2 title: "Class 5: Data Visualization with GGPLOT"
3 author: "Barry"
4 format: html
5 ---
6
7 # Our first ggplot
8 To use the ggplot2 package I first need to have it installed on my computer.
9
10 To install any package we use the `install.packages()` function along with the package name as input argument (i.e. `install.packages("ggplot2")`.
11
12 Now can I use it? NO! first we need to load the package with the `library()` function (i.e. `library(ggplot2)`).
13
14 ```{r}
15 library(ggplot2)
16 ggplot()
17 ```
8:52
Our first ggplot
Quarto
Console
Environment
History
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Edit
Publish
Class 5: Data Visualization with GGPLOT
AUTHOR
Barry
Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the install.packages() function along with the package name as input argument (i.e. install.packages("ggplot2").
Now can I use it? NO! first we need to load the package with the library() function (i.e. library(ggplot2)).
library(ggplot2)
ggplot()


class05 - RStudio
Go to file/function
Addins
class05
class05.qmd
Render on Save
Render
Run
Source
Visual
Outline
Environment
History
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Edit
Publish
---
title: "Class 5: Data Visualization with GGPLOT"
author: "Barry"
format: html

Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the `install.packages()` function along with the package name as input argument (i.e. `install.packages("ggplot2")`.
Now can I use it? NO! first we need to load the package with the `library()` function (i.e. `library(ggplot2)`).
```{r}
library(ggplot2)
ggplot()
```
8:52
Our first ggplot
Quarto
Console
Class 5: Data Visualization with GGPLOT
AUTHOR
Barry
Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the install.packages() function along with the package name as input argument (i.e. install.packages("ggplot2").
Now can I use it? NO! first we need to load the package with the library() function (i.e. library(ggplot2)).
library(ggplot2)
ggplot()


class05 - RStudio
Go to file/function
Addins
class05
class05.qmd
Render on Save
Render
Run
Source
Visual
Outline
Environment
History
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Edit
Publish
---
title: "Class 5: Data Visualization with GGPLOT"
author: "Barry"
format: html

Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the `install.packages()` function along with the package name as input argument (i.e. `install.packages("ggplot2")`.
Now can I use it? NO! first we need to load the package with the `library()` function (i.e. `library(ggplot2)`).
```{r}
library(ggplot2)
ggplot()
```
8:52
Our first ggplot
Quarto
Console
Class 5: Data Visualization with GGPLOT
AUTHOR
Barry
Our first ggplot
To use the ggplot2 package I first need to have it installed on my computer.
To install any package we use the install.packages() function along with the package name as input argument (i.e. install.packages("ggplot2").
Now can I use it? NO! first we need to load the package with the library() function (i.e. library(ggplot2)).
library(ggplot2)
ggplot()


Follow Along!
quarto® demo

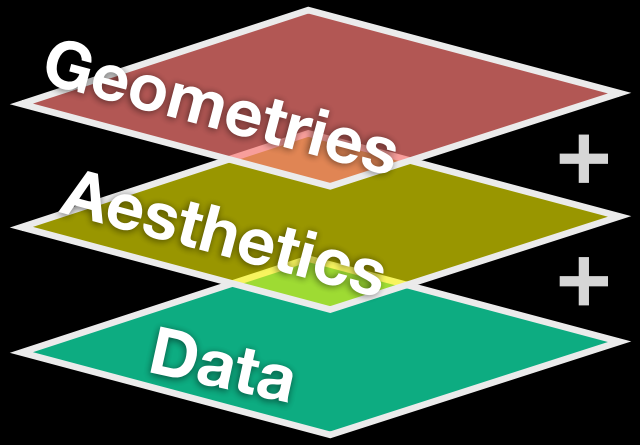
Summary!


ggplot2
Currently the premier plotting library on the planet!

data + aesthetics + geometrys

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. geom_point(), geom_line(), ...)

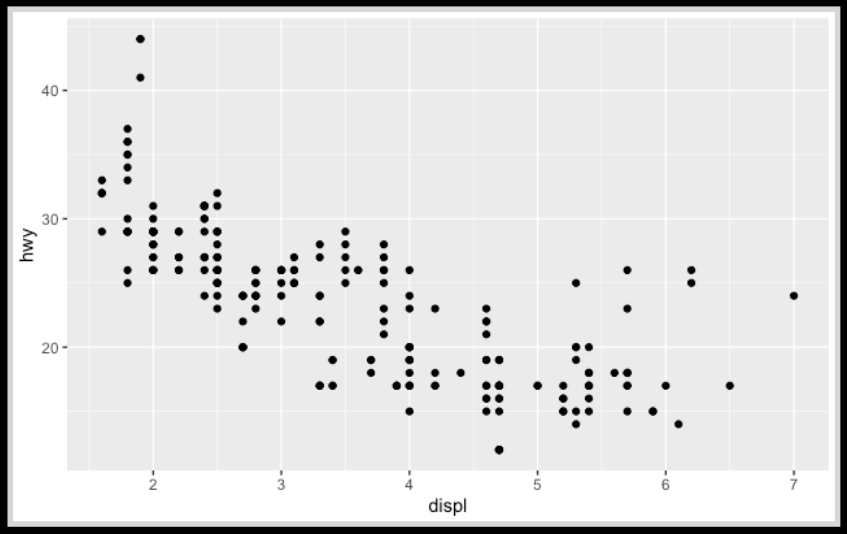
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy) +
  geom_point()
```



data + aesthetics + geometrys

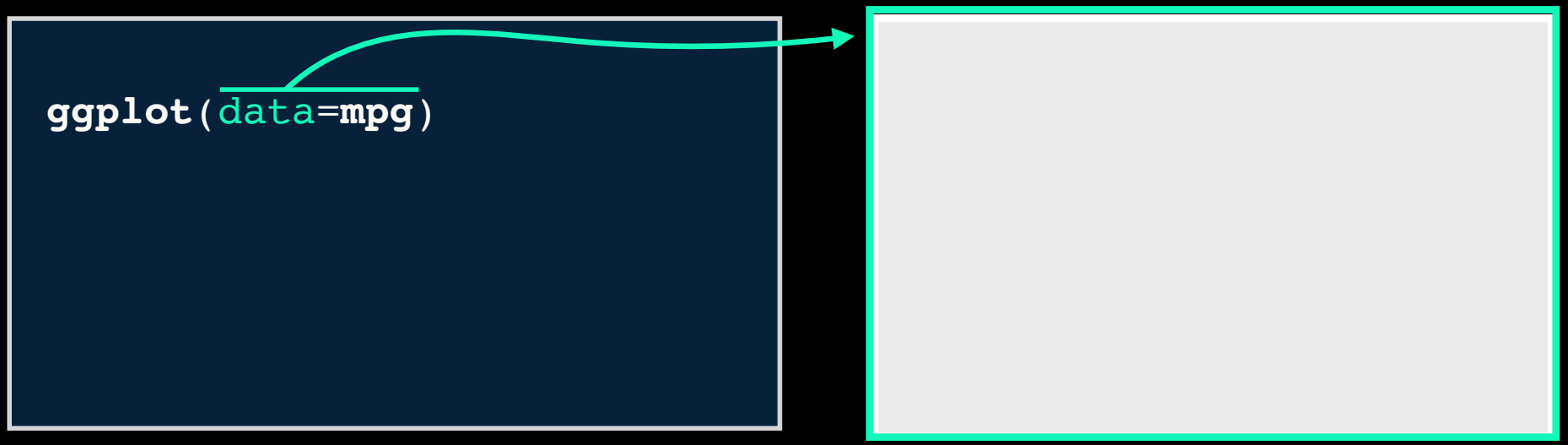
- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. geom_point(), geom_line(), ...)

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



data + aesthetics + geometrys

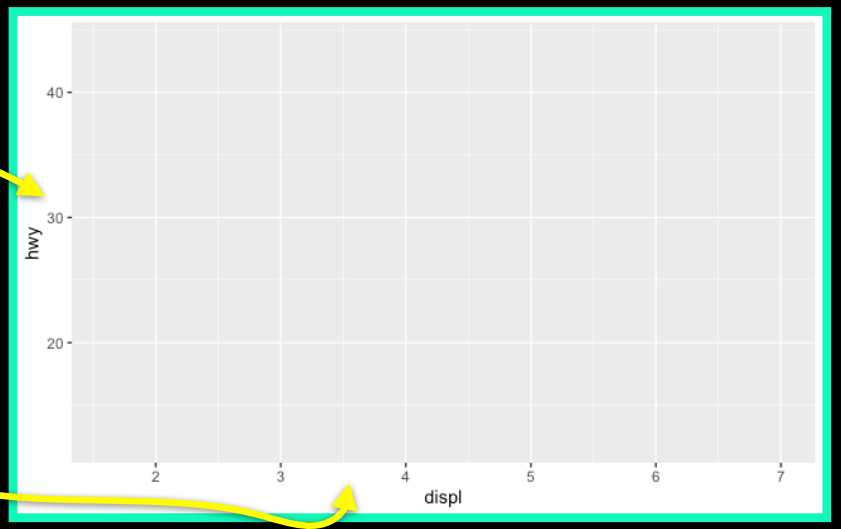
- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. geom_point(), geom_line(), ...)



data + aesthetics + geometrys

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. geom_point(), geom_line(), ...)

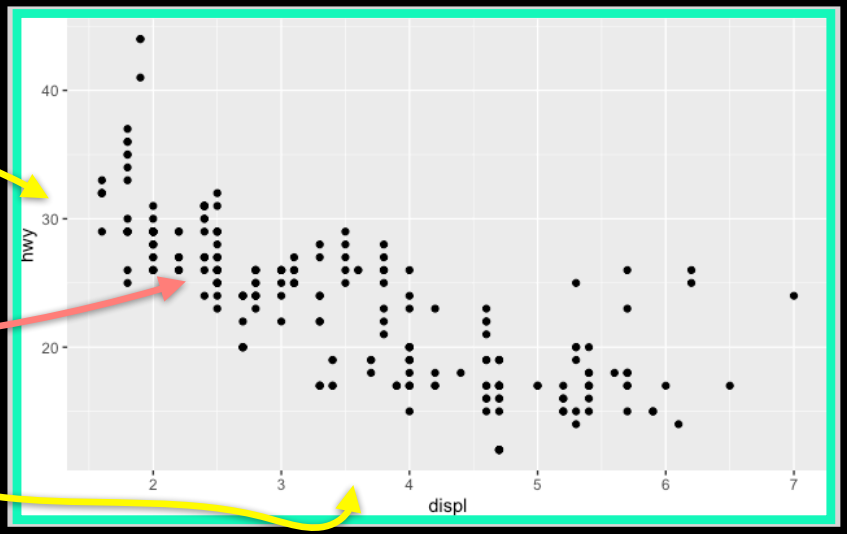
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy)
```



data + aesthetics + geometrys

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. geom_point(), geom_line(), ...)

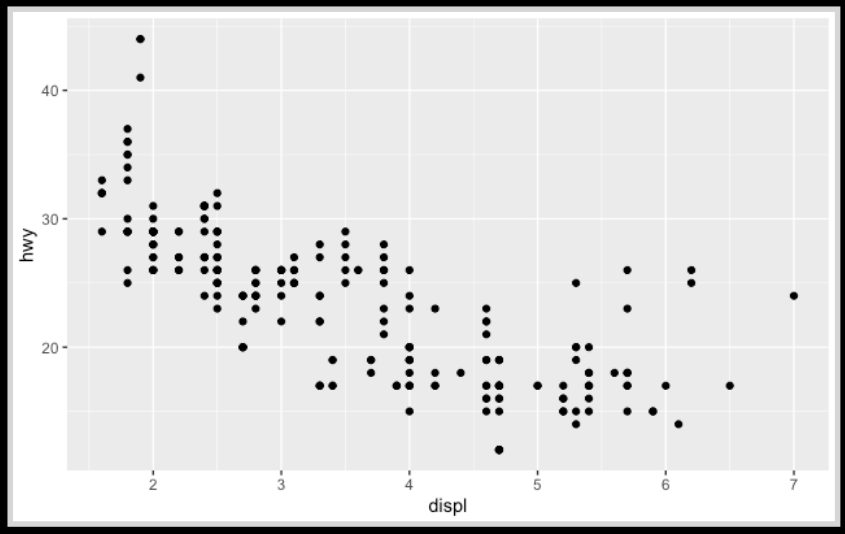
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy) +
  geom_point()
```



data + aesthetics + geometrys

- We can keep building more complicated plots by adding more **layers**

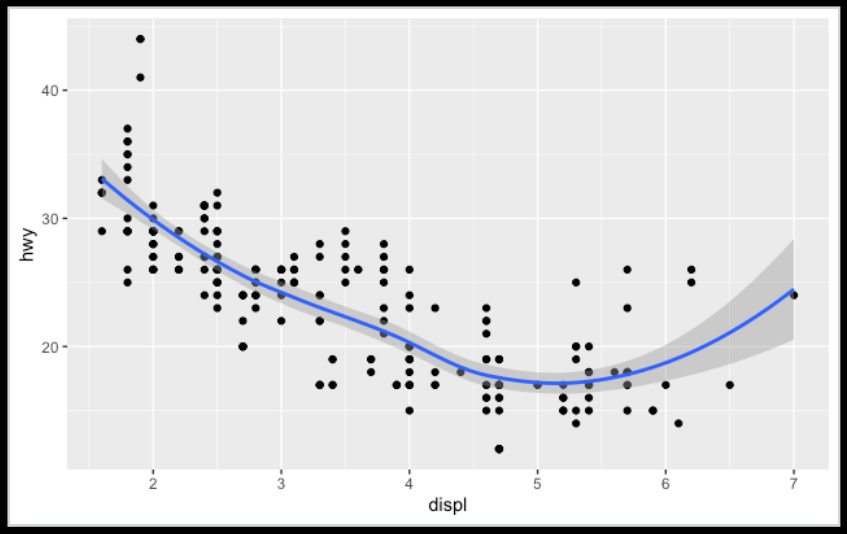
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy) +
  geom_point()
```



data + aesthetics + geometrys

- We can keep building more complicated plots by adding more **layers**
- For example lets add another ***geom***, in this case a smooth line fitted to the data...

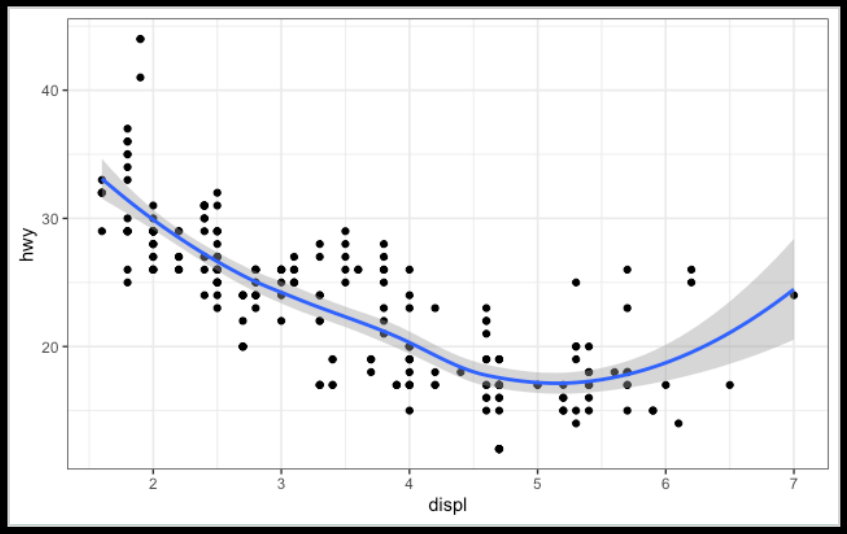
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy) +
  geom_point() +
  geom_smooth()
```



data + aesthetics + geometrys

- We can also add other customizations like themes...

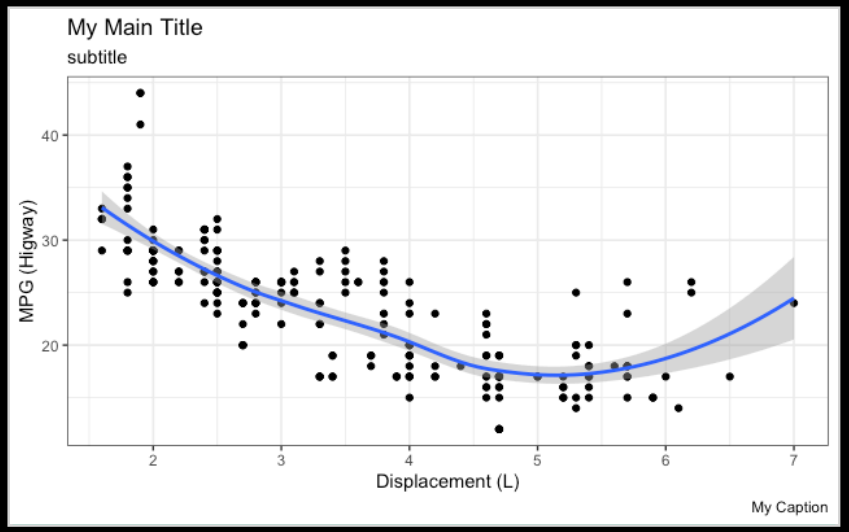
```
ggplot(data=mpg) +
  aes(x=displ, y=hwy) +
  geom_point() +
  geom_smooth() +
  theme_bw()
```



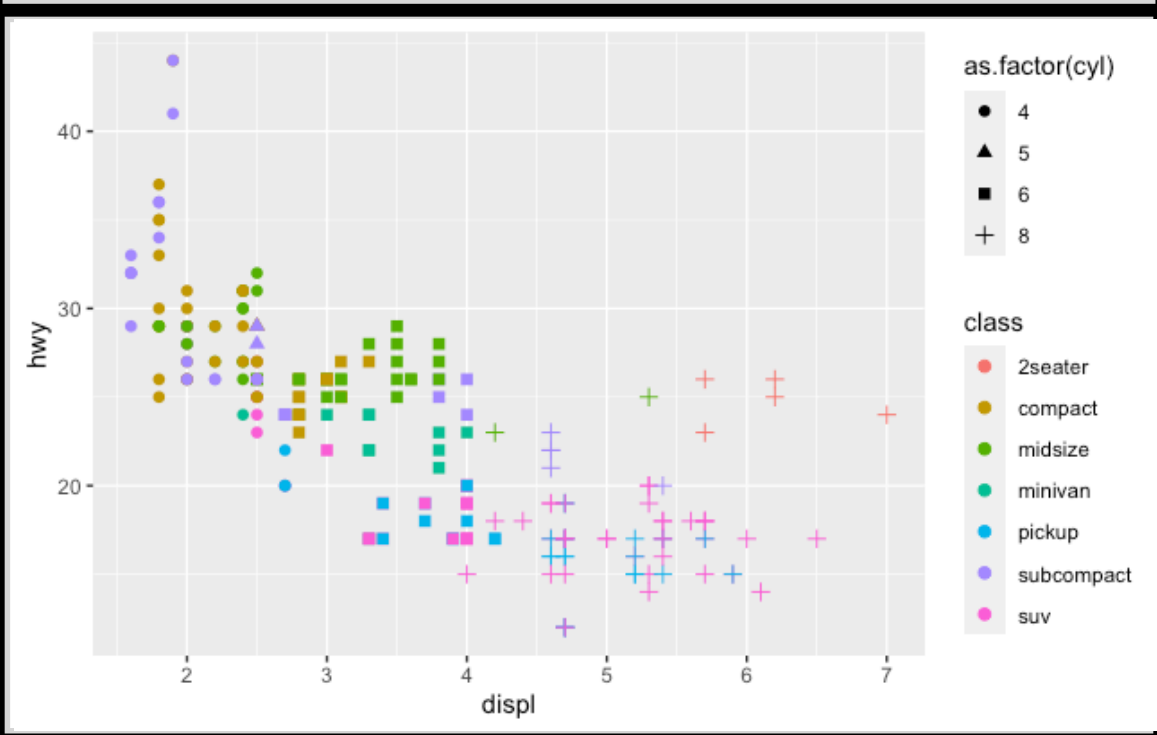
data + aesthetics + geometrys

- And various custom annotation labels...

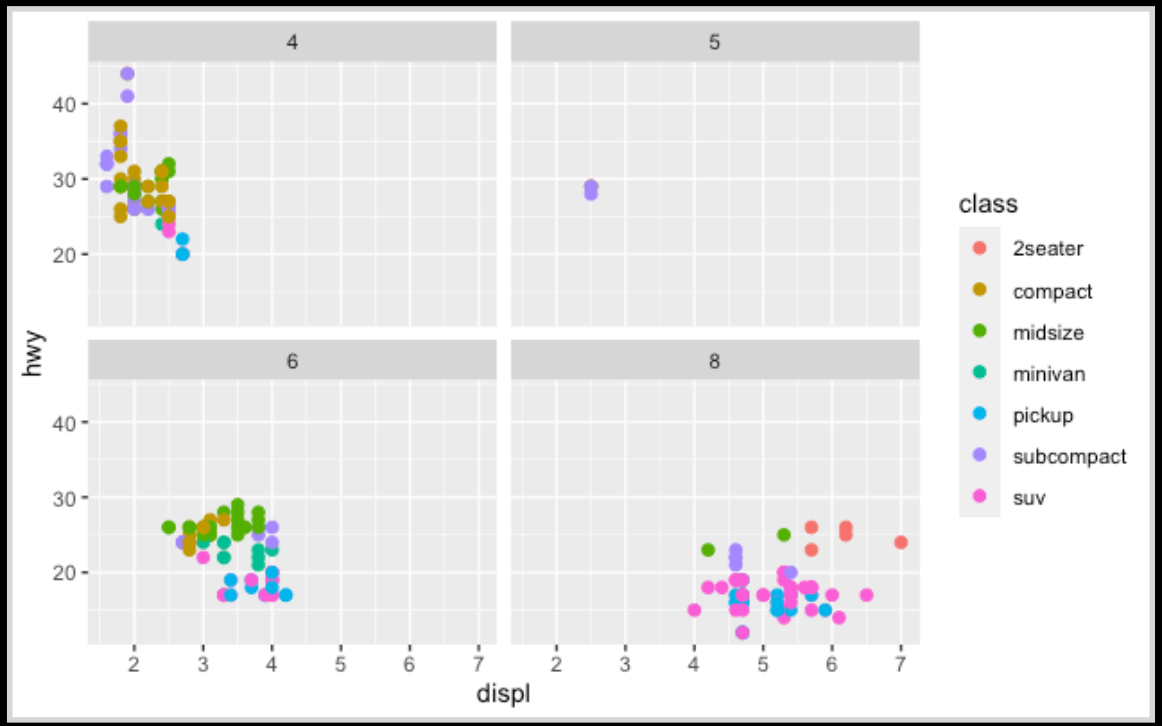
```
ggplot(data=mpg) +
 aes(x=displ, y=hwy) +
 geom_point() +
 geom_smooth() +
 theme_bw()+
 labs(title="My Main Title",
      subtitle = "subtitle",
      caption ="My Caption",
      x="Displacement (L)",
      y= "MPG (Higway)")
```



```
ggplot(data=mpg) +
  aes(x=displ, y=hwy, color=class,
      shape=factor(cyl)) +
  geom_point()
```



```
ggplot(data=mpg) +
  aes(x=displ, y=hwy, color=class) +
  geom_point() +
  facet_wrap(~cyl)
```



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 = <DATA>) +                          required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +         Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, 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' x 5' 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(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

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

a + geom_path(lineend="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, ymax, ymin, 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(yend=lat+1, xend=long+1**))**
b + geom_spoke(aes(angle = 1:1155, radius = 1**))**

ONE VARIABLE continuous

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, linetype, size, weight

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

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

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

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

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

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

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map**)**
+ expand_limits(x = map$long, y = map$lat**)**,
map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

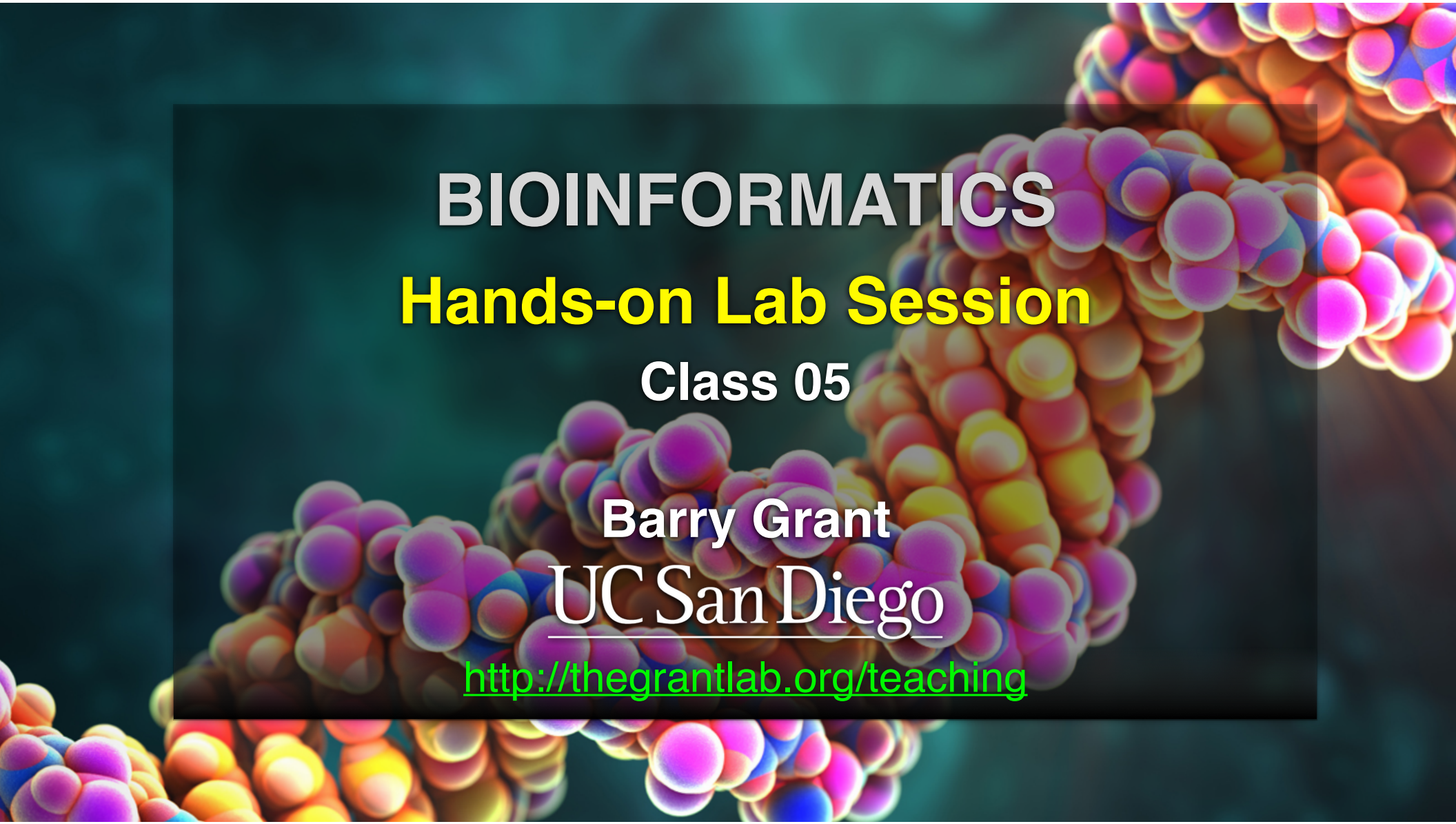
l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

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, width

Learn more about core `geom_FUNCTIONS()`

DataCamp course!



BIOINFORMATICS

Hands-on Lab Session

Class 05

Barry Grant

UC San Diego

http://thegrantlab.org/teaching

Required to make PDFs

In your UNIX “Terminal” (not the “R Console”)

```
# In your “Terminal” tab
quarto install tinytex
```

Problems visit:

https://yihui.org/tinytex