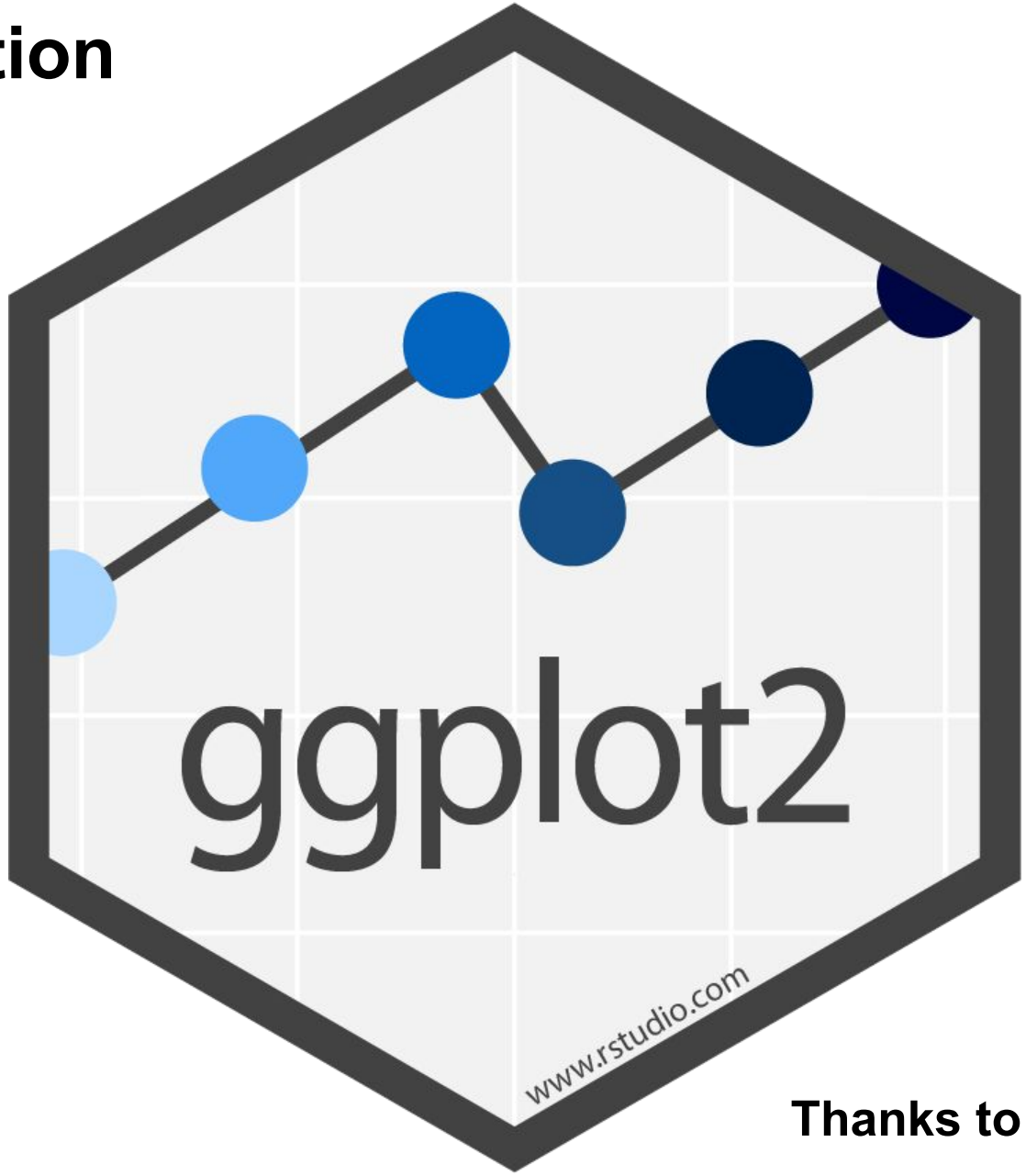
Introduction to….



Thanks to R-Ladies East Lansing for the slides!

First thing's first….

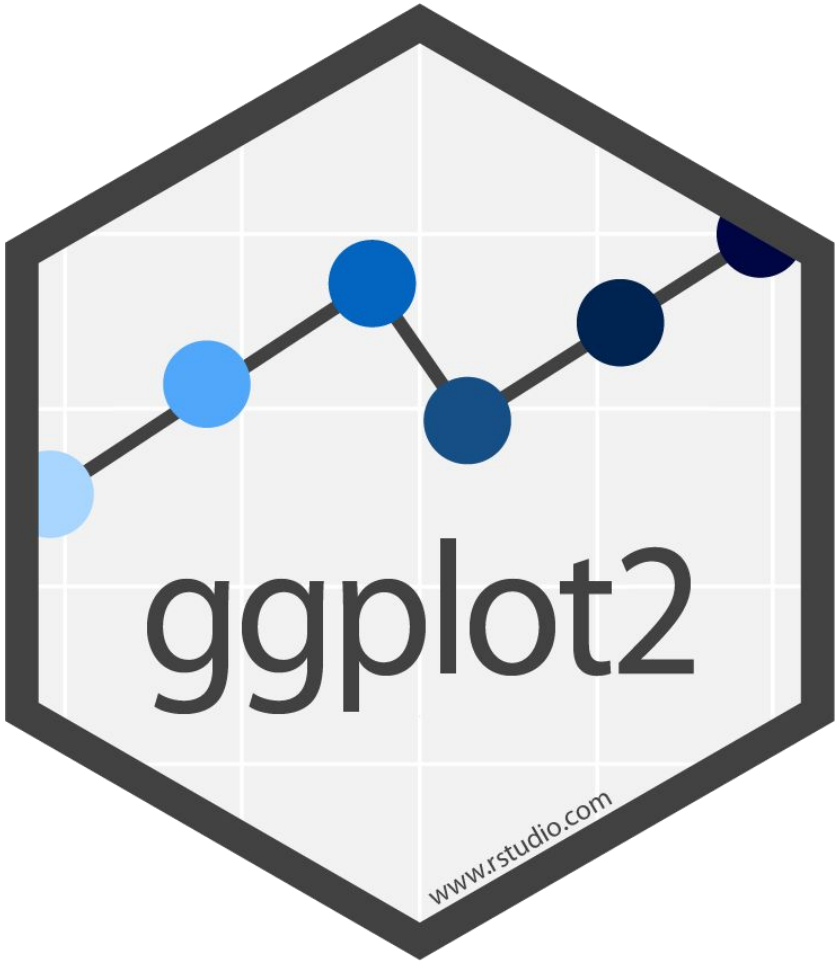
Have you already installed Rstudio? If not, do it now!

https://www.rstudio.com/products/rstudio/download/#download

What is ggplot?

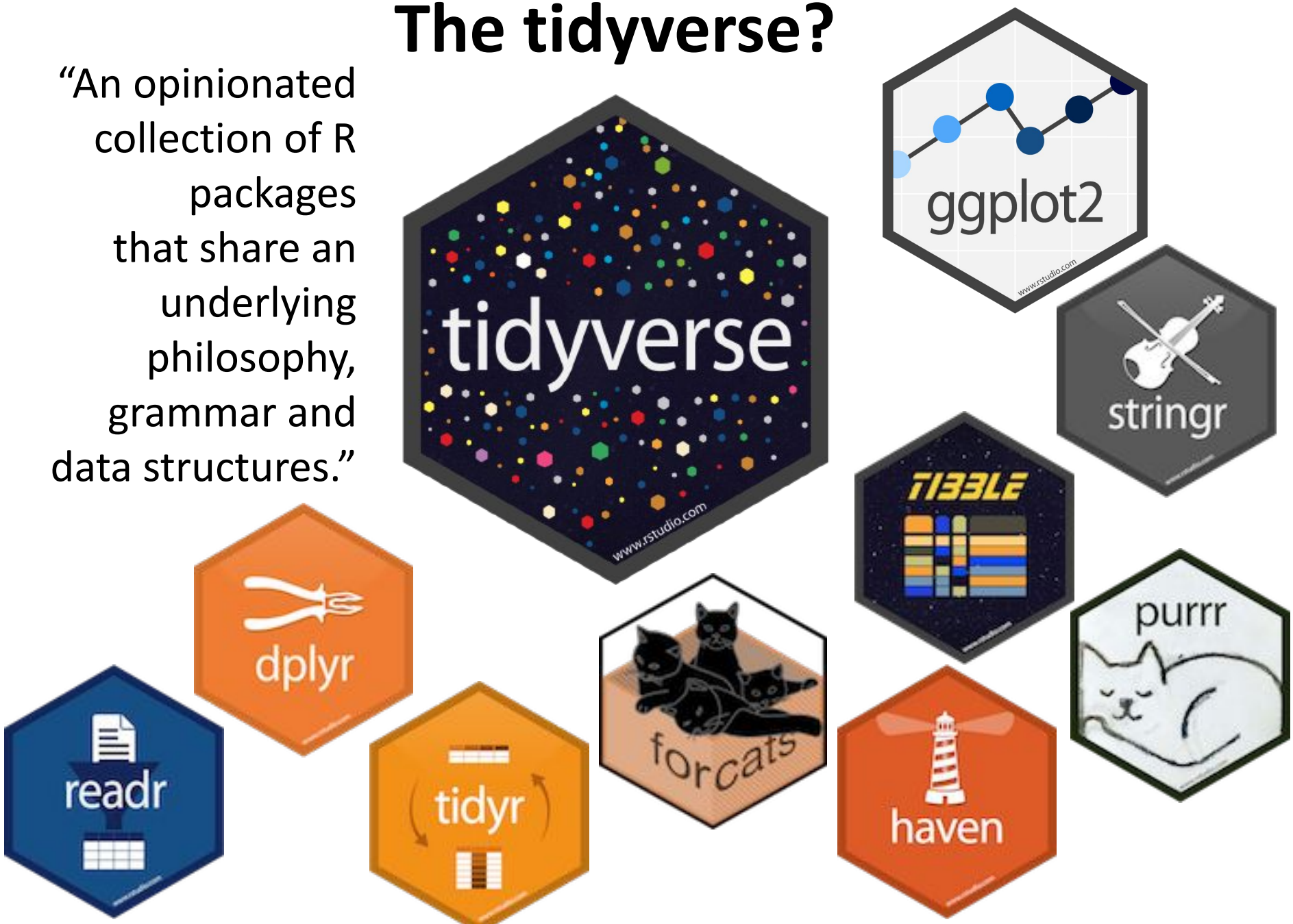
- ggplot is a package (ggplot2) for ‘rapid exploration of data’ graphically in R

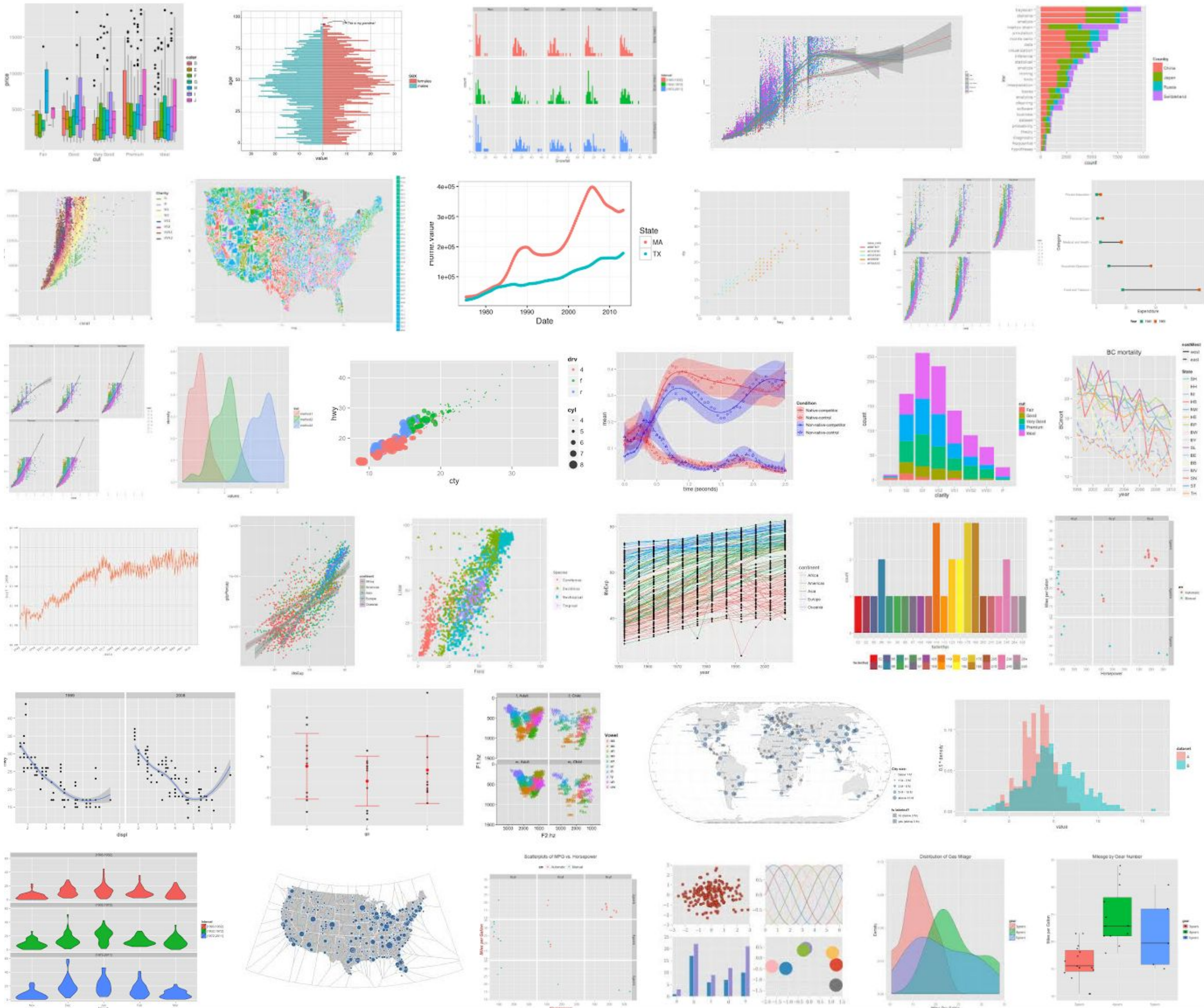
- It is part of the ‘tidyverse’



The tidyverse?

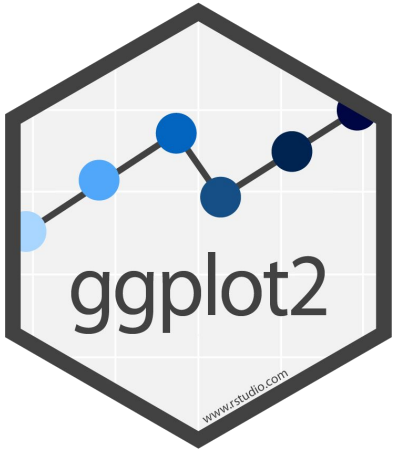
"An opinionated collection of R packages that share an underlying philosophy, grammar and data structures."





Home.value
Date
State
MA
TX
hwy
cty
drv
cyl
BC mortality
year
clarity
displ
F2.hz

Why “gg”?



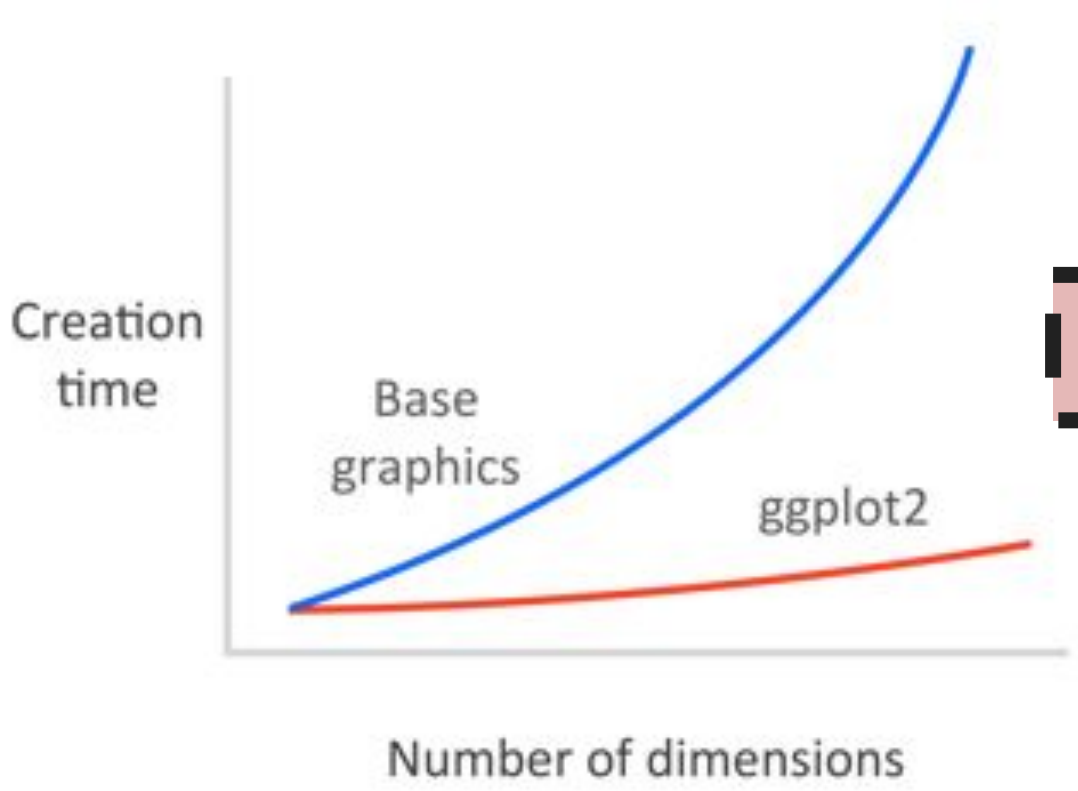
- **gg** is for “grammar of graphics”

- Uses a set of terms that defines the basic components of a plot

- Produces figures using coherent, consistent syntax

ggplot vs. base graphics

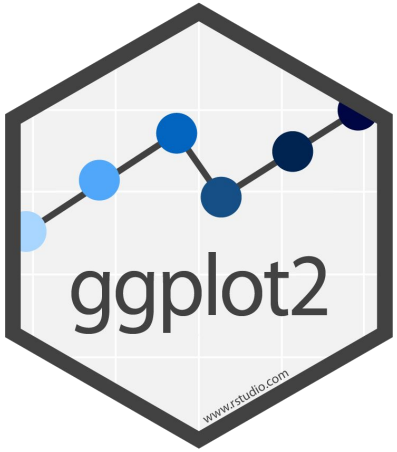
- **ggplot**: rapid data exploration
- **Base graphics**: customization / complexity



Saves time!

*From http://seananderson.ca/

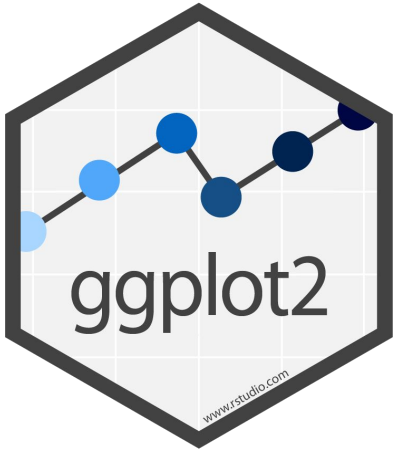
ggplot uses specific grammar



- A basic **ggplot2** plot consists of:
  - **data**: Must be a data.frame
  - **aes**thetics: How your data are represented visually
    - x, y, color, size, shape, etc.

      +
  - **geom**etry: Geometries of plotted objects
    - points, lines, polygons, etc.
  - and more…

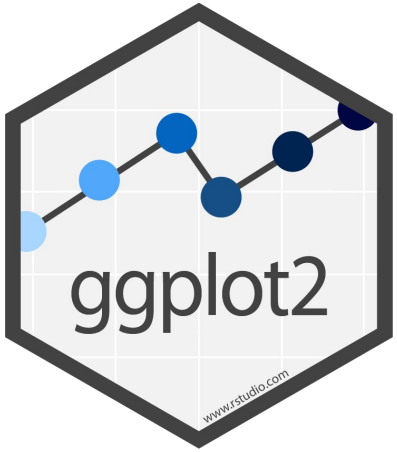
ggplot uses specific grammar



- A basic **ggplot2** plot consists of:
  - **data**: Must be a data.frame
    - **aes**thetics: How your data are represented visually
      - x, y, color, size, shape, etc.

      +
    - **geom**etry: Geometries of plotted objects
      - points, lines, polygons, etc.
    - and more…

ggplot uses specific grammar

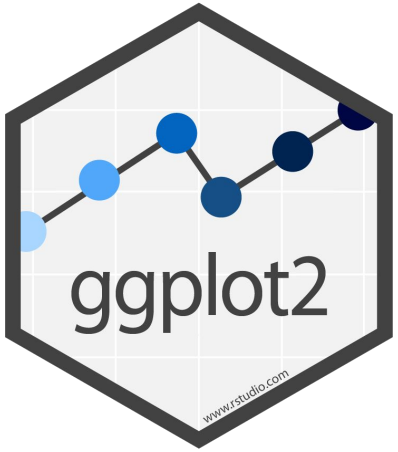


- A basic **ggplot2** plot consists of:
  - **data**: Must be a data.frame
  - **aes**thetics: How your data are represented visually
    - x, y, color, size, shape, etc.

+

  - **geom**etry: Geometries of plotted objects
    - points, lines, polygons, etc.
  - and more…

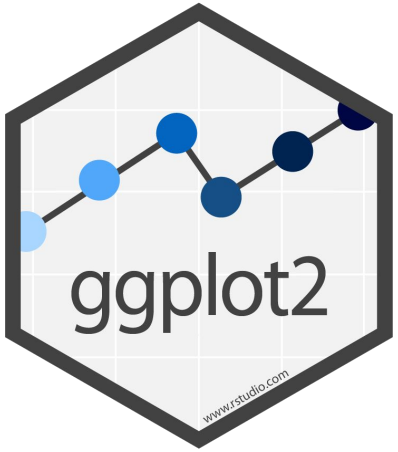
ggplot uses specific grammar



- A basic **ggplot2** plot consists of:
  - **data**: Must be a data.frame
  - **aes**thetics: How your data are represented visually
    - x, y, color, size, shape, etc.

    +
  - **geom**etry: Geometries of plotted objects
    - points, lines, polygons, etc.
  - and more…

ggplot uses specific grammar



- A basic **ggplot2** plot consists of:

– **data**: Must be a data.frame

– **aes**thetics: How your data are represented visually

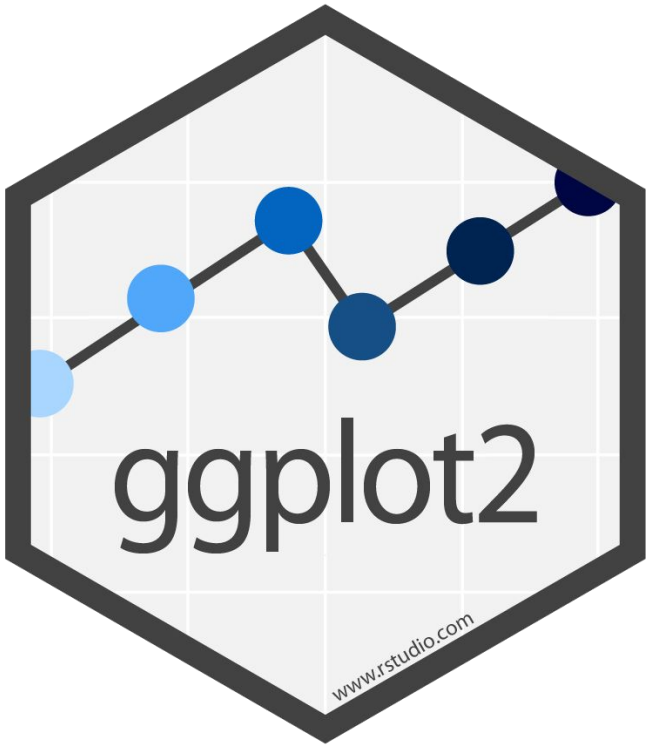
  - x, y, color, size, shape, etc.

+

– **geom**etry: Geometries of plotted objects

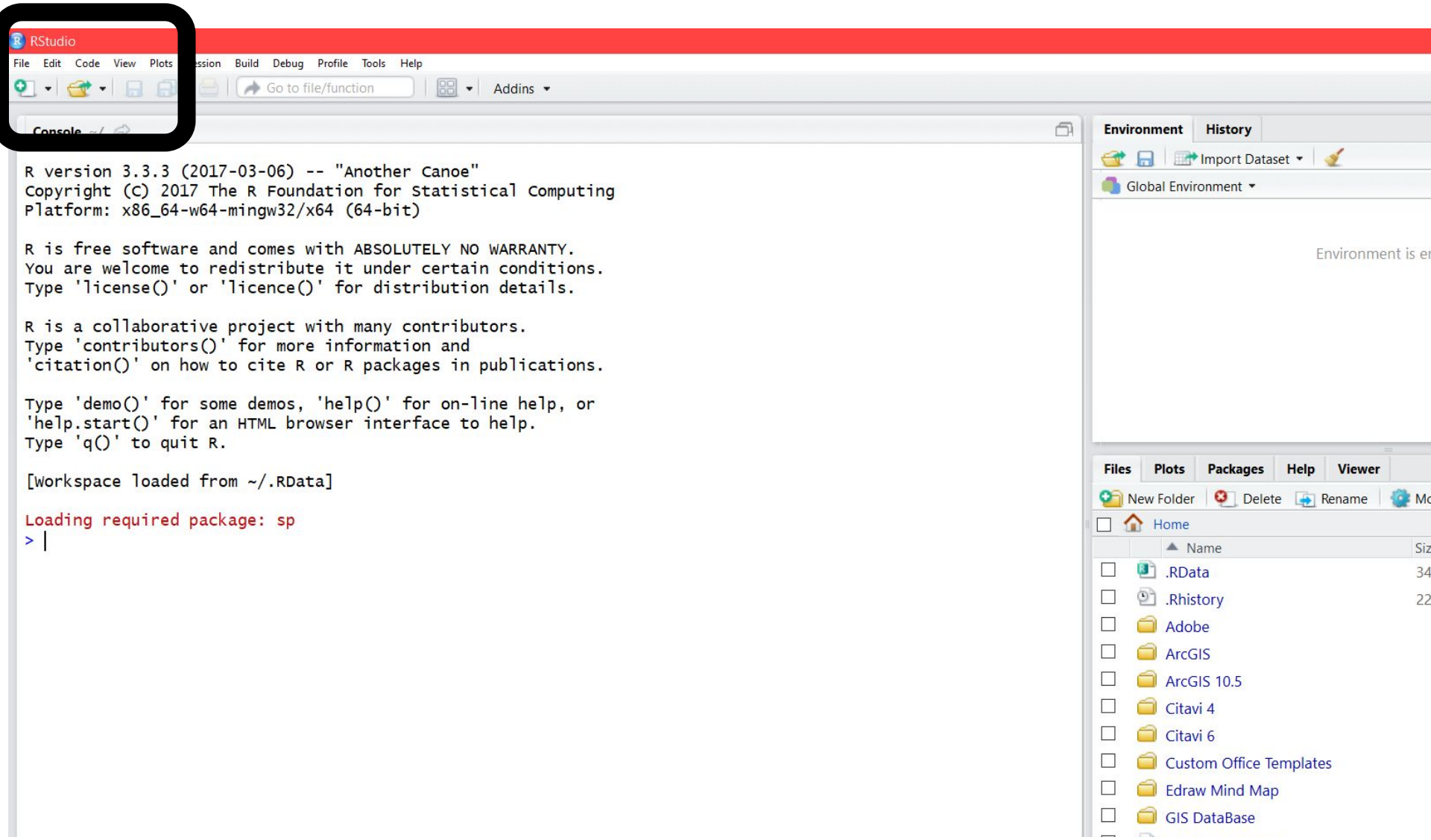
  - points, lines, polygons, etc.

– and more…

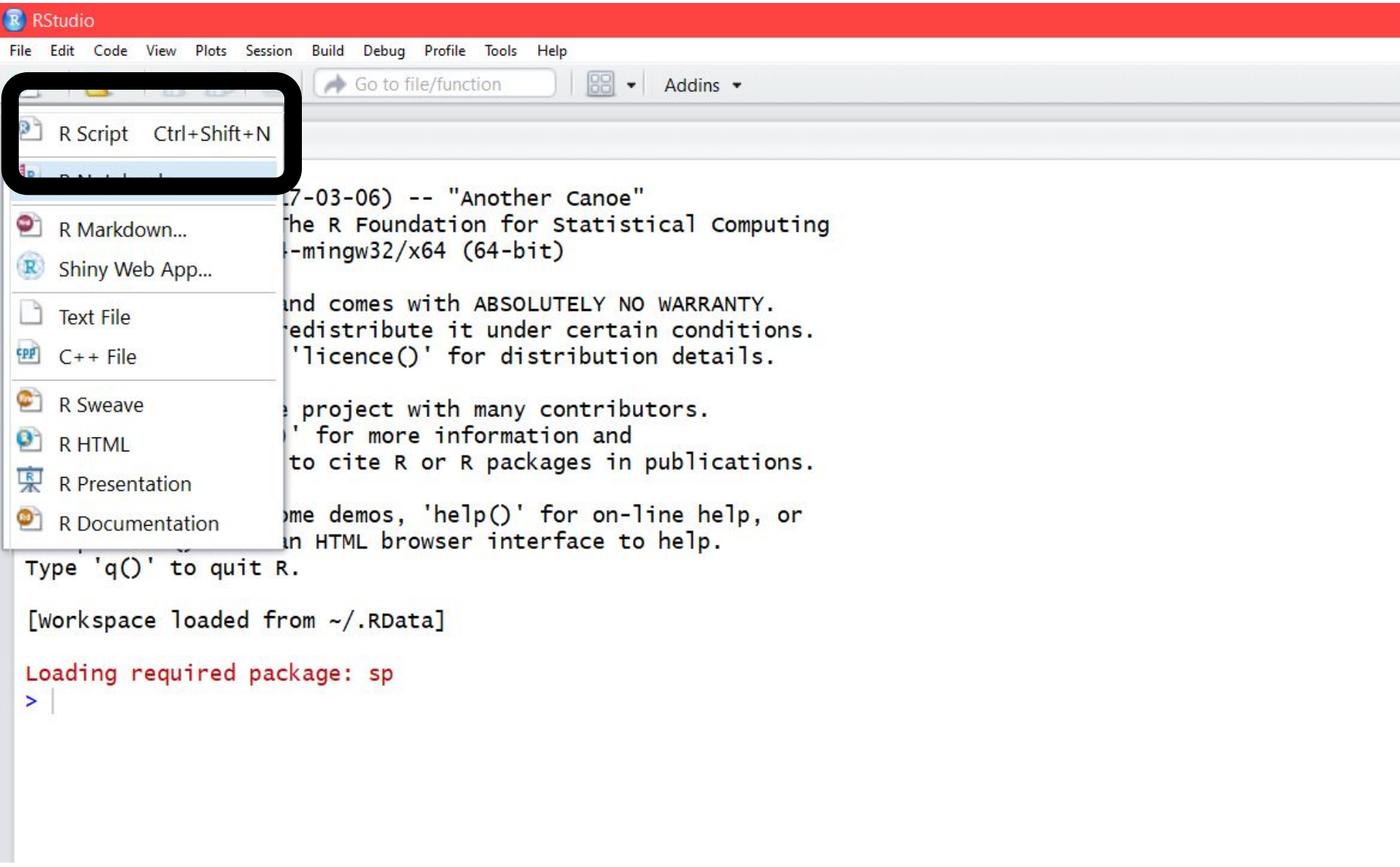


ggplot: Let’s get started!

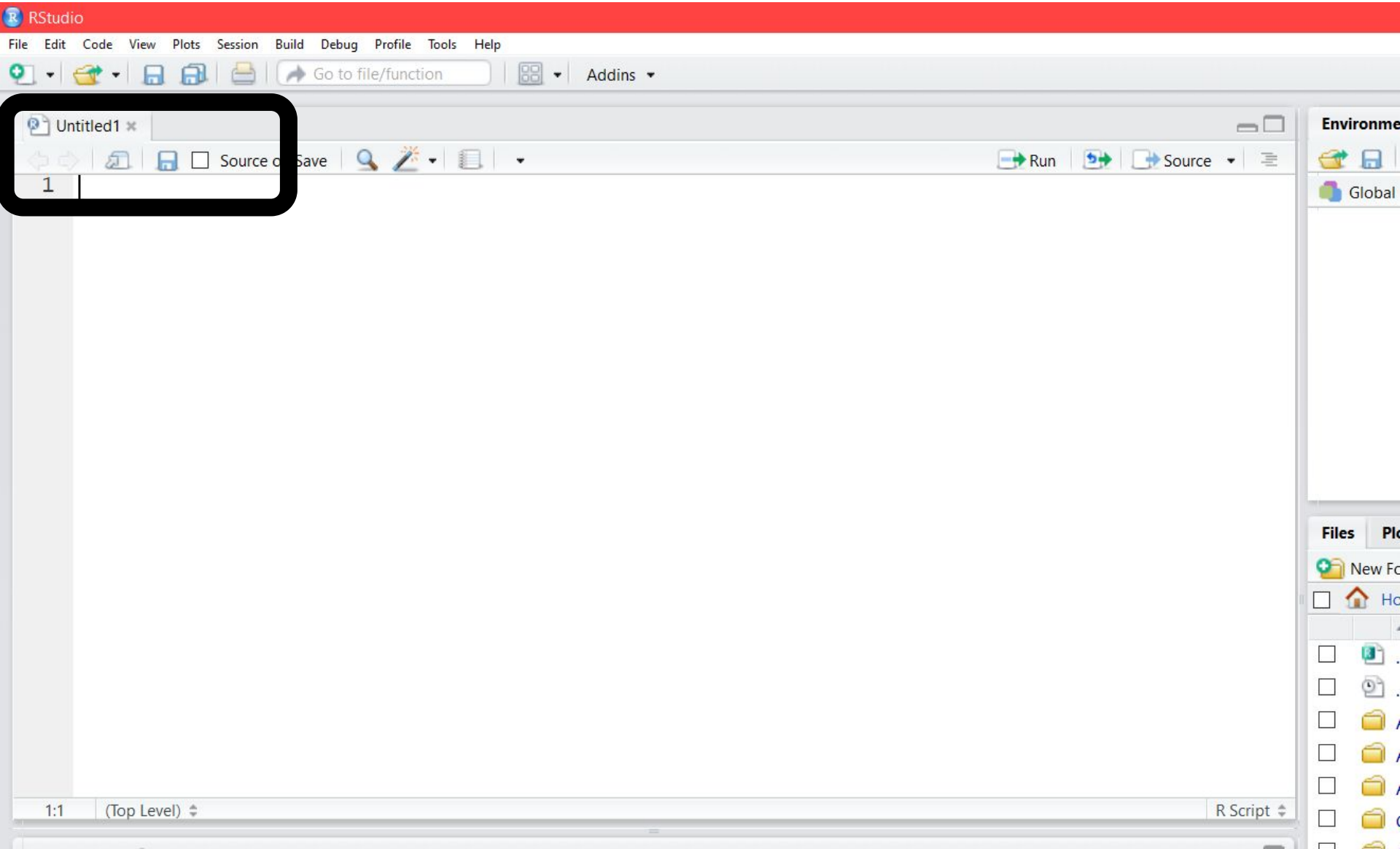
Getting started: opening RStudio



Getting started: opening RStudio



Getting started: opening RStudio



Getting started: using packages

1

```
install.packages("ggplot2")
```

Downloads files to computer

1 x per computer

Getting started: using packages

1

```
install.packages("ggplot2")
```

Downloads files to computer

1 x per computer

2

```
library("ggplot2")
```

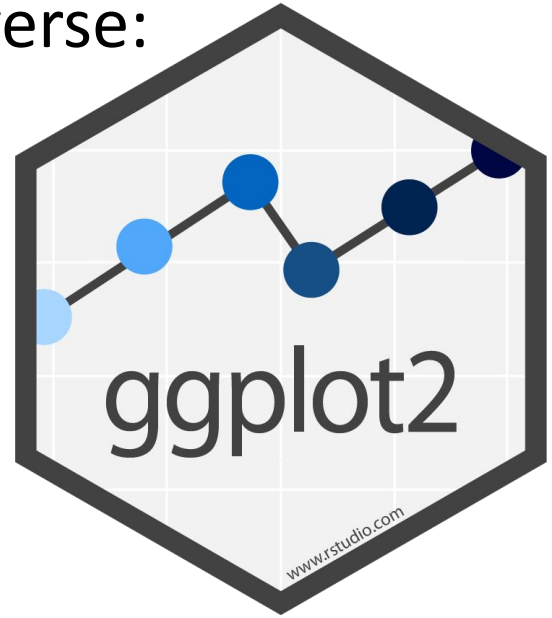
Loads package

1 x per R Session

Getting started: using packages

- To **install** ggplot and the rest of the tidyverse:

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

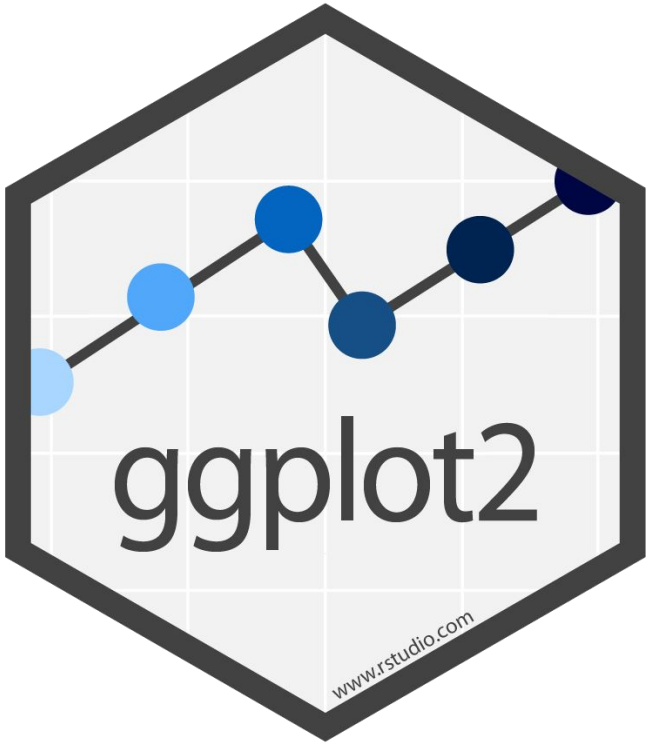


- To **load** tidyverse into workspace:

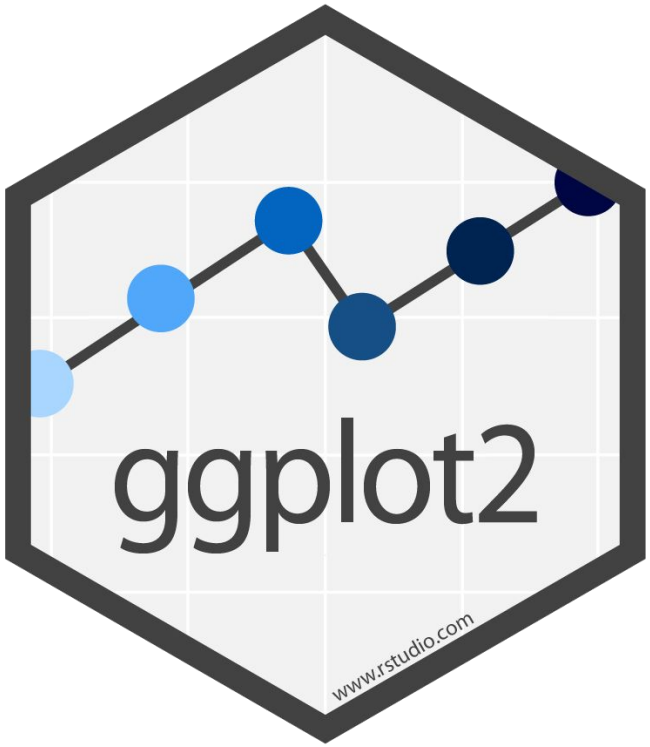
```
library("tidyverse")
```

Importing data

```
iris <- read.csv(file=”Iris.csv”)
```



Part 1: ggplot grammar!



ggplot grammar: data

ggplot grammar: data

```
head(iris)
```

ggplot grammar: data

head(iris)

```
> head(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
```

Best Practices: start with tidy data!

Psst. Attend our Nov 5 Tidy Data Workshop!

ggplot grammar: data

```
str(iris)
```

ggplot grammar: data

str(iris)

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1
 1 1 1 1 1 1 ...
```

ggplot grammar: data

ggplot()

ggplot grammar: data

ggplot()

ggplot grammar: data

```
basic_plot <- ggplot(data=iris)
```

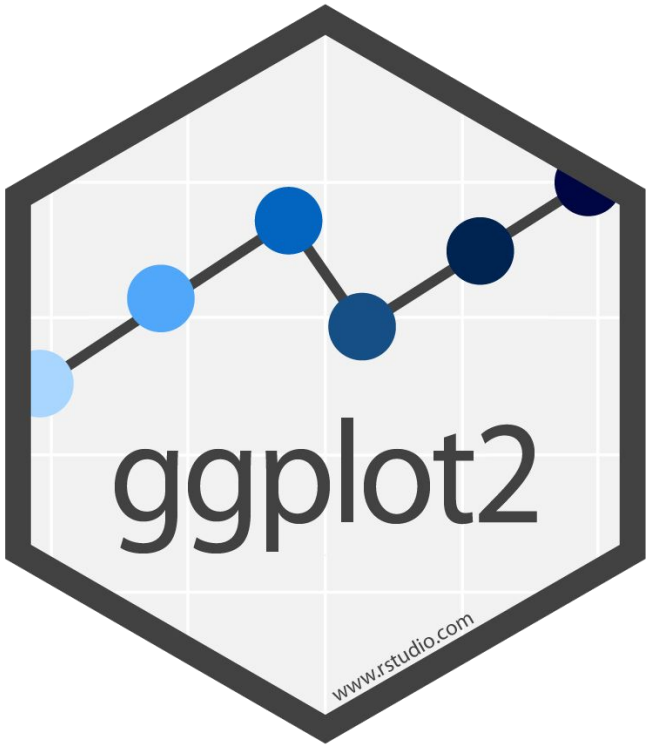
```
basic_plot
```

Best Practices: use informative names for your variables!

ggplot grammar: data

```
basic_plot <- ggplot(data=iris)
```

```
basic_plot
```



ggplot grammar: aesthetics

ggplot grammar: aesthetics

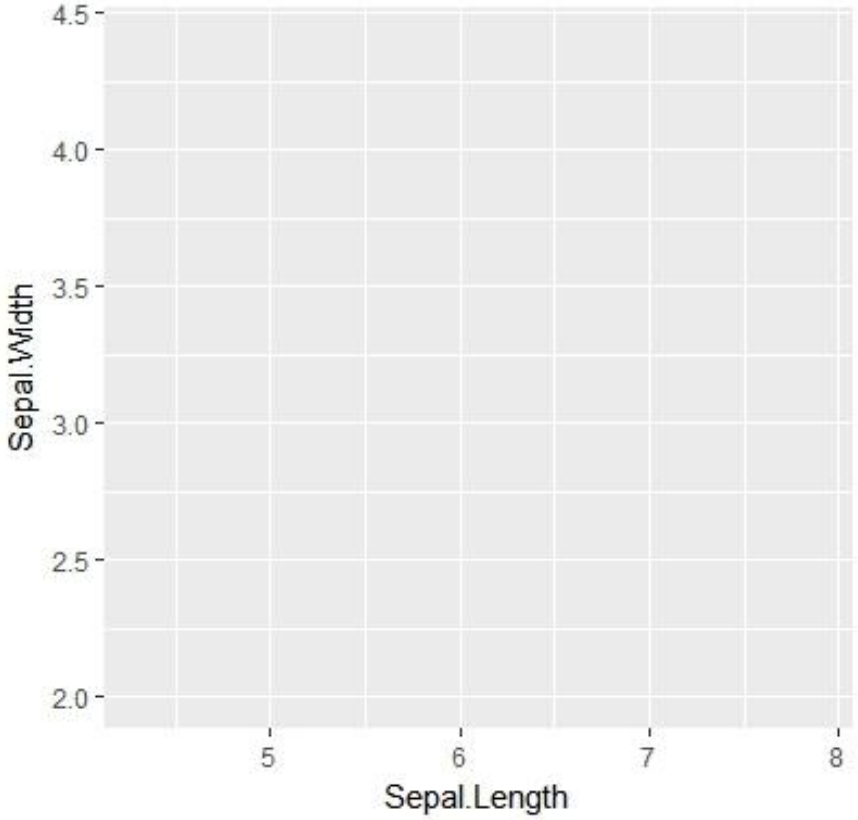
```
basic_plot +
      aes(x=Sepal.Length, y=Sepal.Width)
```

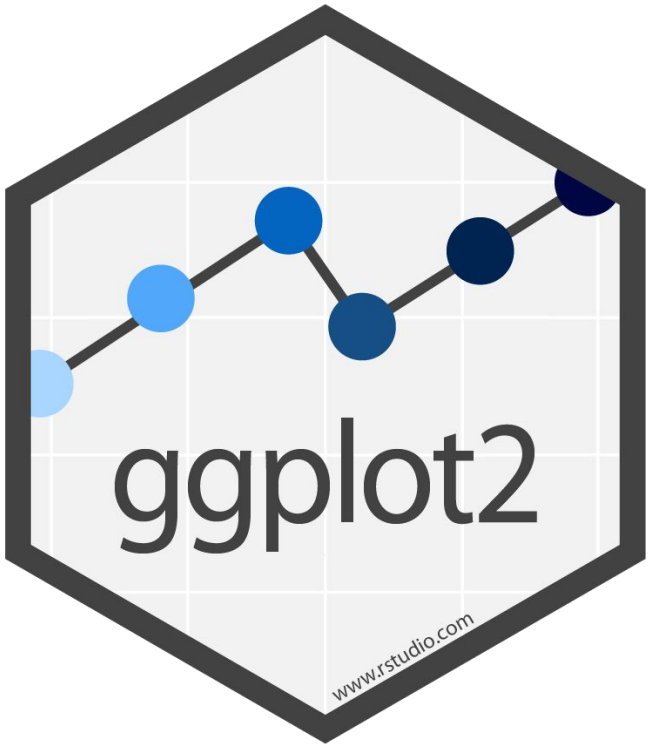
Reminder: basic_plot <- ggplot(data=iris)

ggplot grammar: aesthetics

basic_plot +

aes(x=Sepal.Length, y=Sepal.Width)





ggplot grammar: geometry

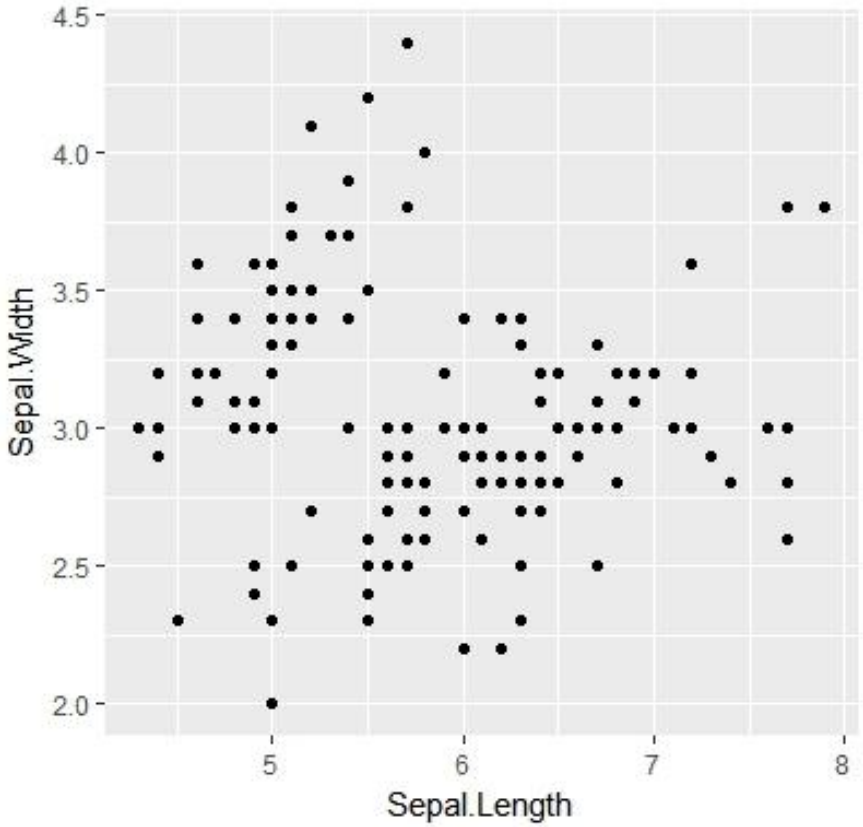
ggplot grammar: geometry

```
basic_plot +
    aes(x=Sepal.Length, y=Sepal.Width) +
    geom_point()
```

Reminder: basic_plot <- ggplot(data=iris)

ggplot grammar: geometry

```
basic_plot +
    aes(x=Sepal.Length, y=Sepal.Width) +
    geom_point()
```



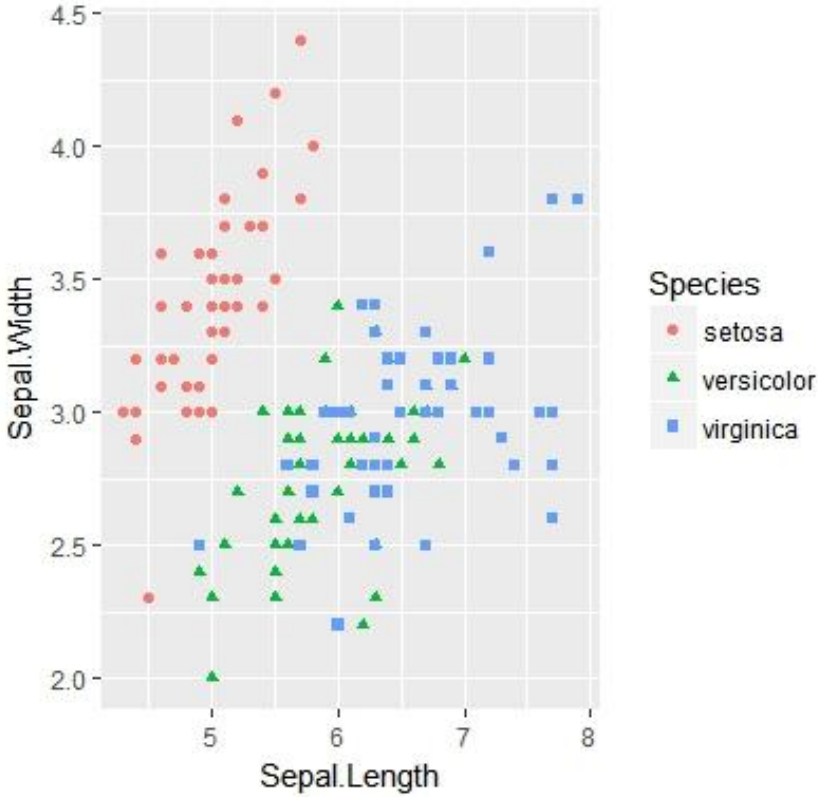
ggplot grammar: geometry & aesthetics

```
basic_plot +
        aes(x=Sepal.Length, y=Sepal.Width,
        color=Species, shape=Species) +
        geom_point()
```

Reminder: basic_plot <- ggplot(data=iris)

ggplot grammar: geometry & aesthetics

```
basic_plot +
        aes(x=Sepal.Length, y=Sepal.Width,
        color=Species, shape=Species) +
        geom_point()
```



ggplot grammar: geometry & aesthetics

```
basic_plot +
        aes(x=Sepal.Length, y=Sepal.Width,
        color=Species, shape=Species) +
        geom_point() +
        geom_smooth(method='lm')
```

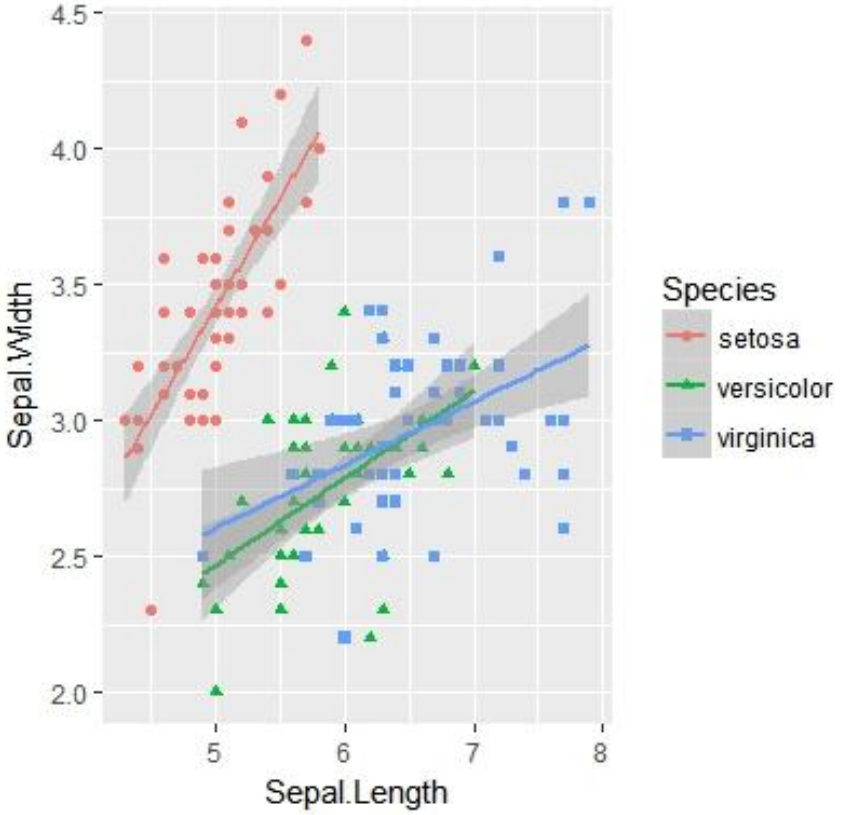
Reminder: basic_plot <- ggplot(data=iris)

ggplot grammar: geometry & aesthetics

```
basic_plot +
        aes(x=Sepal.Length, y=Sepal.Width,
        color=Species, shape=Species) +
        geom_point() +
        geom_smooth(method='lm')
```

Other **geom**'s also exist.

-coming soon!-



ggplot grammar: review

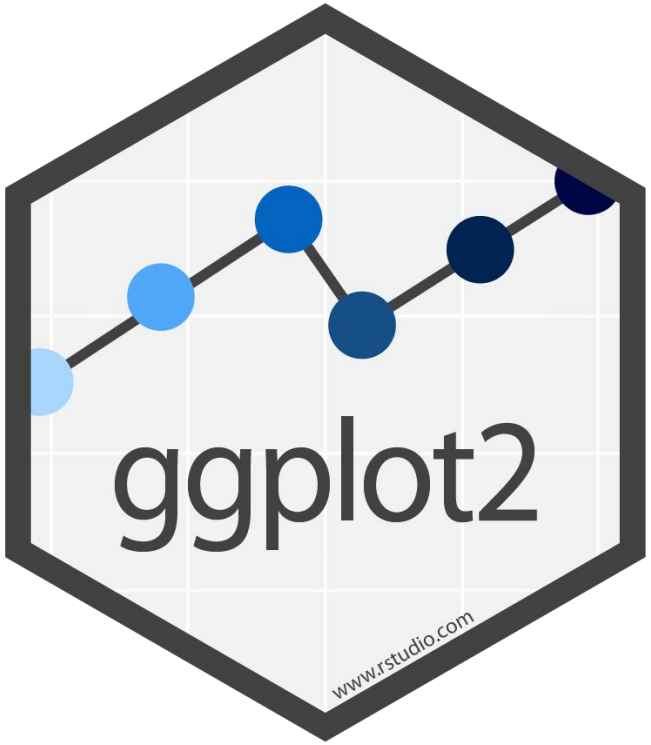
- Specify the data using **data=**

  ggplot(data = _____) + aes(x=___, y=___) + …

  or

  ggplot(data = _____, aes(x=___, y=___)) + …

  …use '+' sign to add layers:



ggplot grammar: It’s your turn!

ggplot grammar: practice

- Make a plot called “quiz_1”
- Use the data set: mpg
- Make a box plot using “geom_boxplot()”
- Examine the relationship between “class” and “cty”

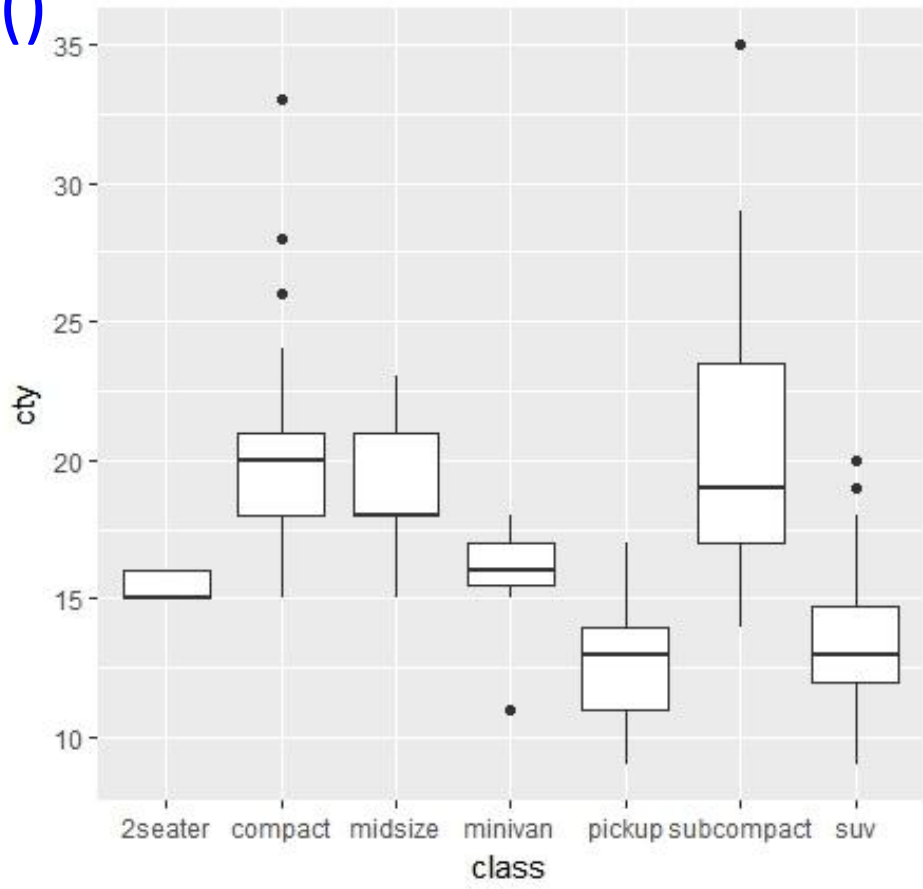
Put up a red post-it if you’re stuck!

ggplot grammar: practice

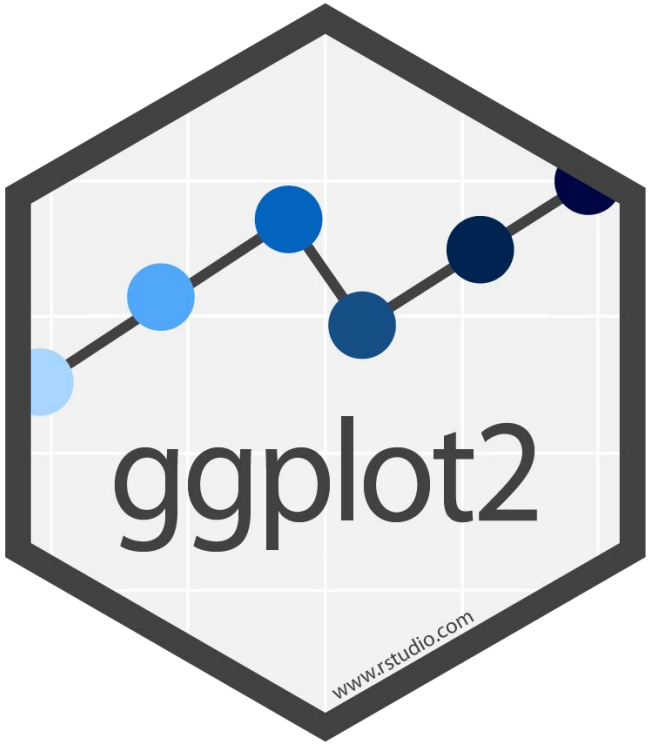
```
head(mpg)
str(mpg)
quiz_1 <- ggplot(data = mpg) +
            aes(x = class, y = cty) +
            geom_boxplot()
quiz_1
```

ggplot grammar: practice

```
quiz_1 <- ggplot(data = mpg) +
              aes(x = class, y = cty) +
              geom_boxplot()
quiz_1
```



Inspired yet?



ggplot part 1: COMPLETE!

Dataviz 2.0 | publication-ready plots & tables

Hosted By
Janani R. and 3 others

Details
Moving ahead with ggplot2!

WHO is this for?

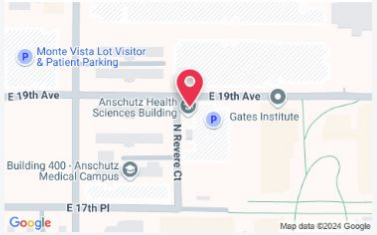
- If you are newish to R but have dabbled a bit!
- If you have heard about ggplot2 and want to learn more.

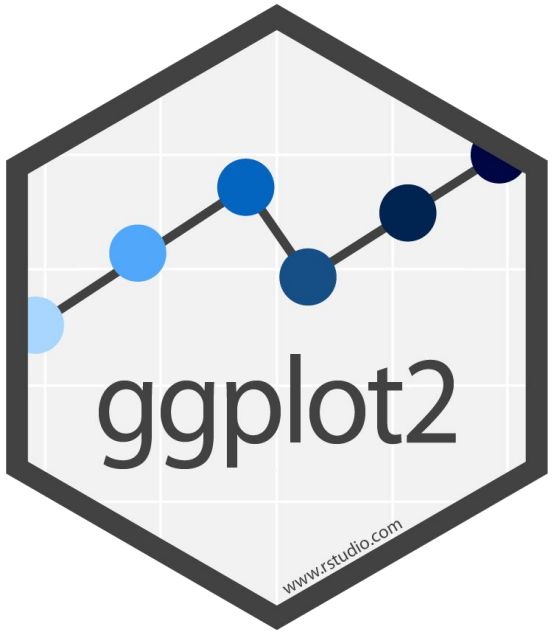
Organizer tools

R-Ladies Aurora
Public group

Tuesday, October 15, 2024
5:00 PM to 6:30 PM MDT
Add to calendar

TBD
1890 N Revere Ct, Aurora, CO 80045
· Aurora, CO





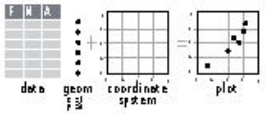
ggplot part 2: Making publication ready plots and tables

Data Visualization with ggplot2

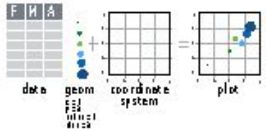
Cheat Sheet

Basics

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



To display data values, map variables in the data set to aesthetic properties of the geom like size, color, and x and y locations.



Build a graph with ggplot() or qplot()

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

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method = "lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a geom_*() or stat_*() function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

aesthetic mappings | data | geom

```
qplot(x = cty, y = hwy, color = cyl, 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 to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

```
a <- ggplot(seals, aes(x = long, y = lat))
b <- ggplot(economics, aes(date, unemploy))
```

```
a + geom_blank()
```
(Useful for expanding limits)

```
a + geom_curve(aes(yend = lat + delta_lat, xend = long + delta_long, curvature = z))
```
x, xend, y, yend, alpha, angle, color, curvature, linetype, size

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

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

```
a + geom_rect(aes(xmin = long, ymin = lat, xmax = long + delta_long, ymax = lat + delta_lat))
```
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

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

```
a + geom_segment(aes(yend = lat + delta_lat, xend = long + delta_long))
```
x, xend, y, yend, alpha, color, linetype, size

```
a + geom_spoke(aes(yend = lat + delta_lat, xend = long + delta_long))
```
x, y, angle, radius, alpha, color, linetype, size

One Variable

Continuous

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

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

```
a + geom_area(aes(y = ..density..), stat = "bin")
```

```
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

```
a + geom_freqpoly(aes(y = ..density..))
```

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

```
a + geom_histogram(aes(y = ..density..))
```

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_bar(stat = "identity")
```
x, y, alpha, color, fill, linetype, size, weight

```
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

More practice, more fun!

[ggplot2_practice_rmarkdown.Rmd]