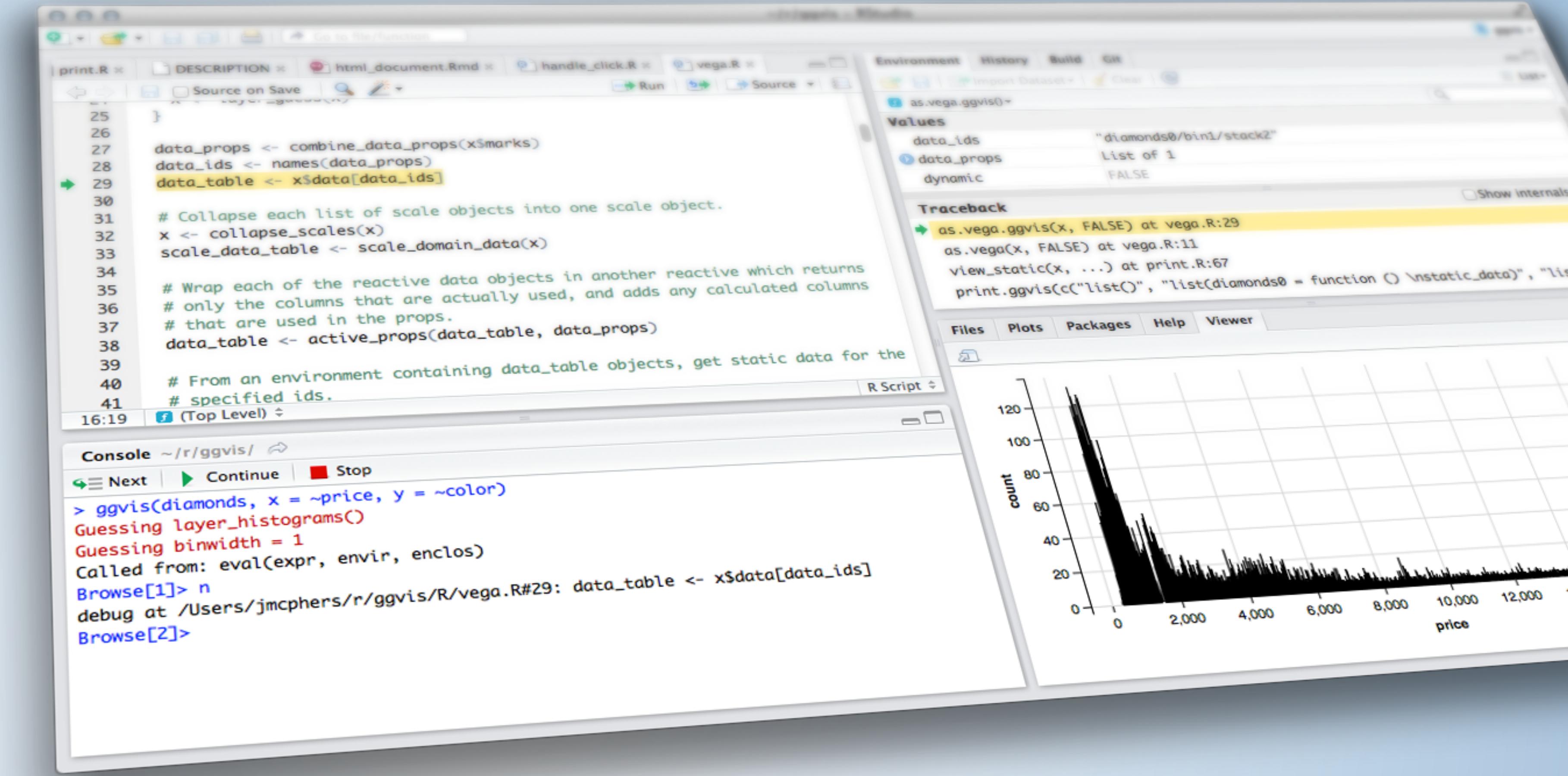


# INTERACTIVE DATA VISUALIZATIONS WITH PLOTLY



# OUTLINE

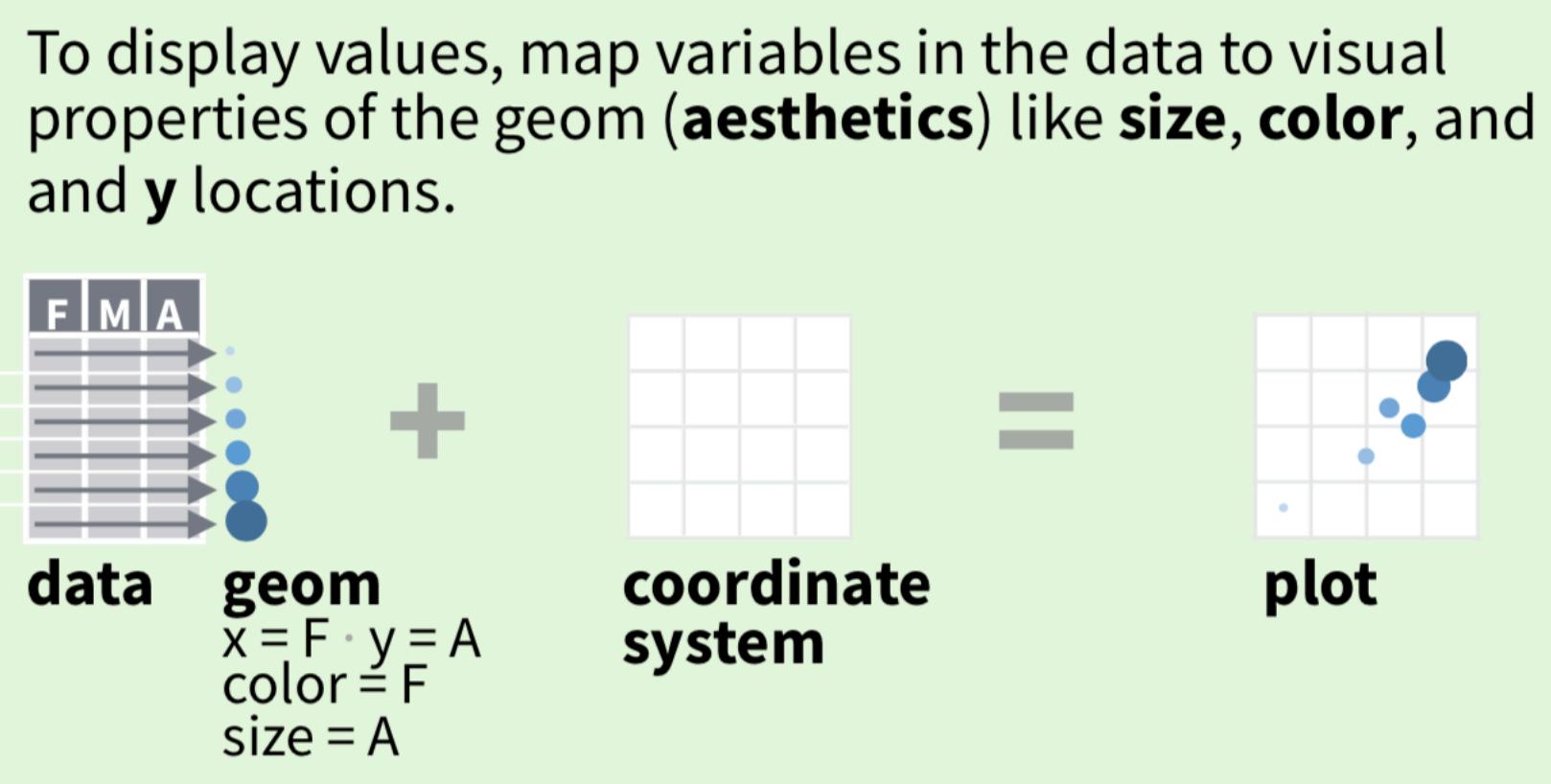
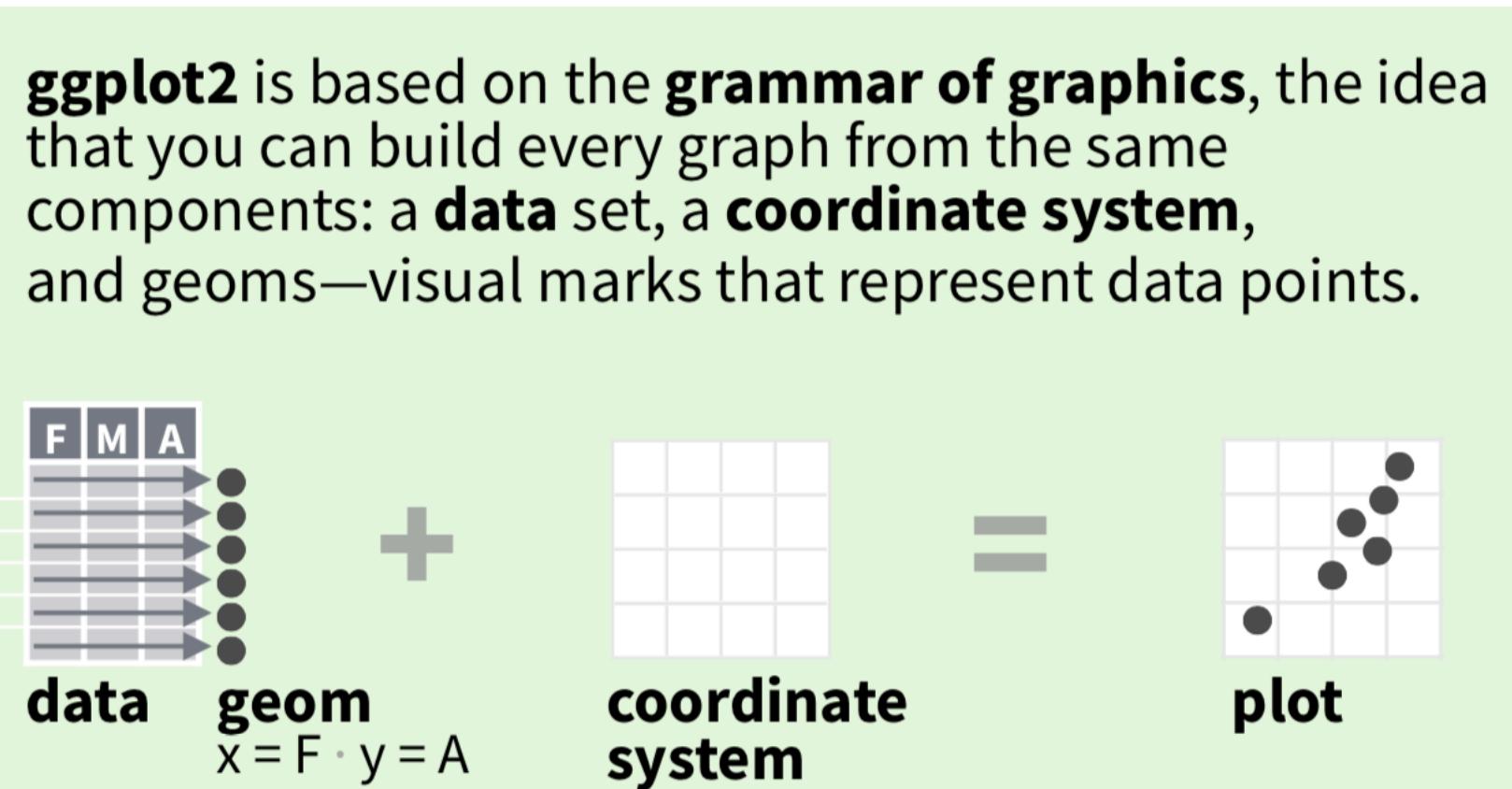
- ▶ ggplot2
  - ▶ Building a plot
  - ▶ Refresher
- ▶ Plotly
  - ▶ ggplotly
  - ▶ plotly standalone
  - ▶ plotly in shiny
- ▶ Advanced DT

“Most of us need to listen to the music to understand how beautiful it is. But often that’s how we present statistics: we just show the notes, we don’t play the music.”

-Hans Rosling

# ggplot2

# CREATING A GG PLOT



Complete the template below to build a graph.

```
ggplot (data = <DATA>) +  
<GEOM_FUNCTION>(mapping = aes(<Mappings>),  
stat = <STAT>, position = <POSITION>) +  
<COORDINATE_FUNCTION> +  
<FACET_FUNCTION> +  
<SCALE_FUNCTION> +  
<THEME_FUNCTION>
```

A green bracket on the right side of the template indicates which parts are required (the first one) and which are not required but have sensible defaults supplied (the others).

**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.

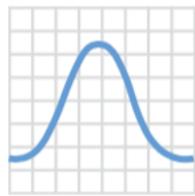
# TYPES OF PLOTS

## ONE VARIABLE continuous

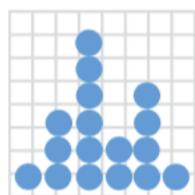
```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```



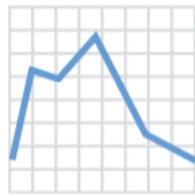
**c + geom\_area(stat = "bin")**  
x, y, alpha, color, fill, linetype, size



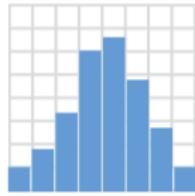
**c + geom\_density(kernel = "gaussian")**  
x, y, alpha, color, fill, group, linetype, size, weight



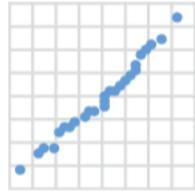
**c + geom\_dotplot()**  
x, y, alpha, color, fill



**c + geom\_freqpoly()** x, y, alpha, color, group,  
linetype, size



**c + geom\_histogram(binwidth = 5)** x, y, alpha,  
color, fill, linetype, size, weight

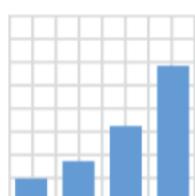


**c2 + geom\_qq(aes(sample = hwy))** x, y, alpha,  
color, fill, linetype, size, weight



## discrete

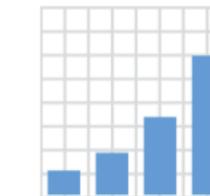
```
d <- ggplot(mpg, aes(fl))
```



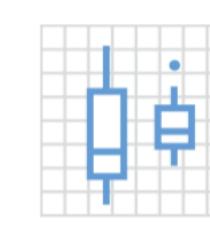
**d + geom\_bar()**  
x, alpha, color, fill, linetype, size, weight

## discrete x , continuous y

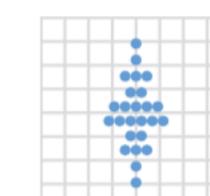
```
f <- ggplot(mpg, aes(class, hwy))
```



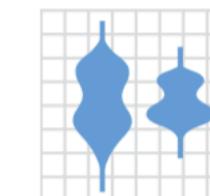
**f + geom\_col()**, x, y, alpha, color, fill, group,  
linetype, size



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



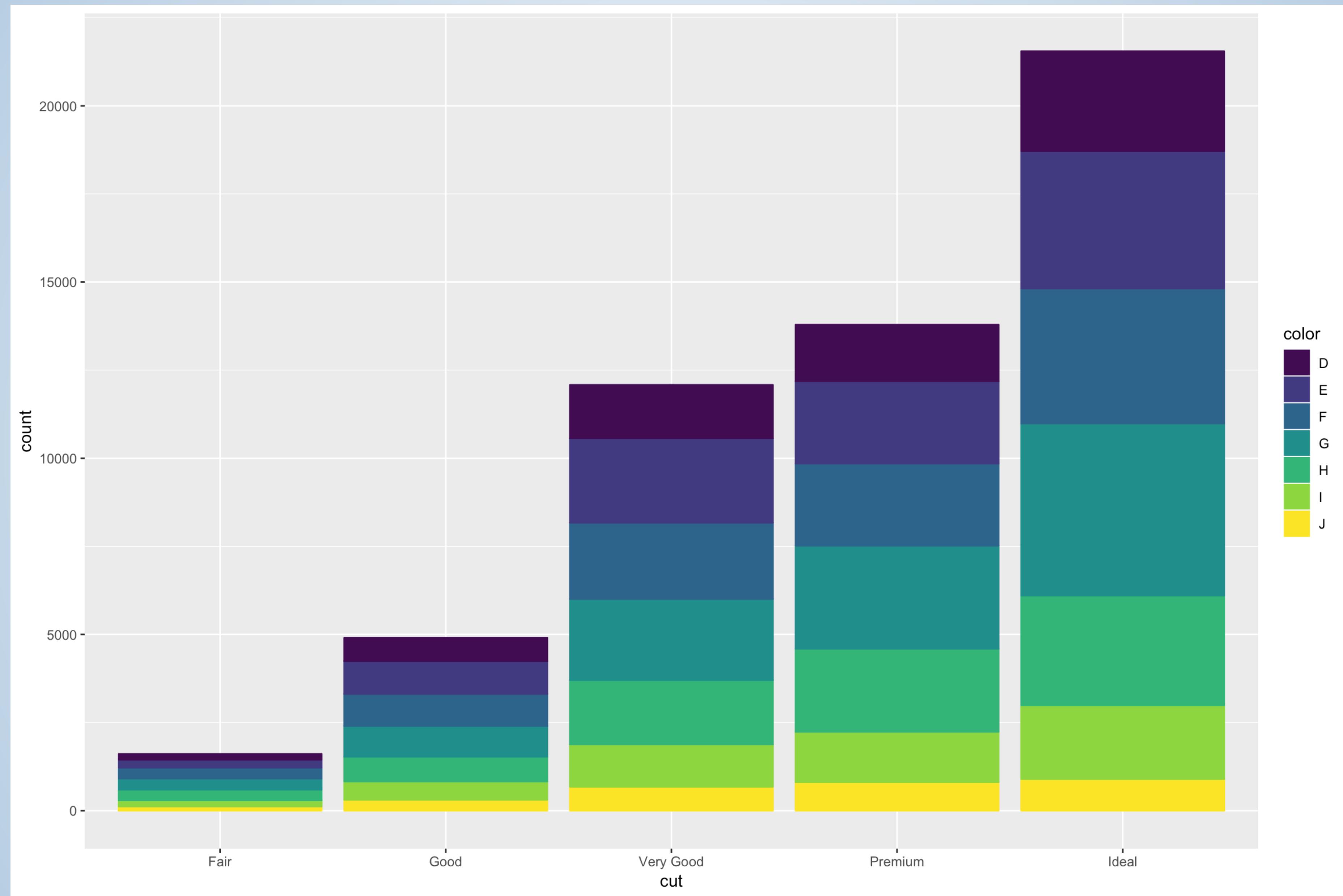
**f + geom\_dotplot(binaxis = "y", stackdir =  
"center")**, x, y, alpha, color, fill, group



**f + geom\_violin(scale = "area")**, x, y, alpha, color,  
fill, group, linetype, size, weight

# DEMO

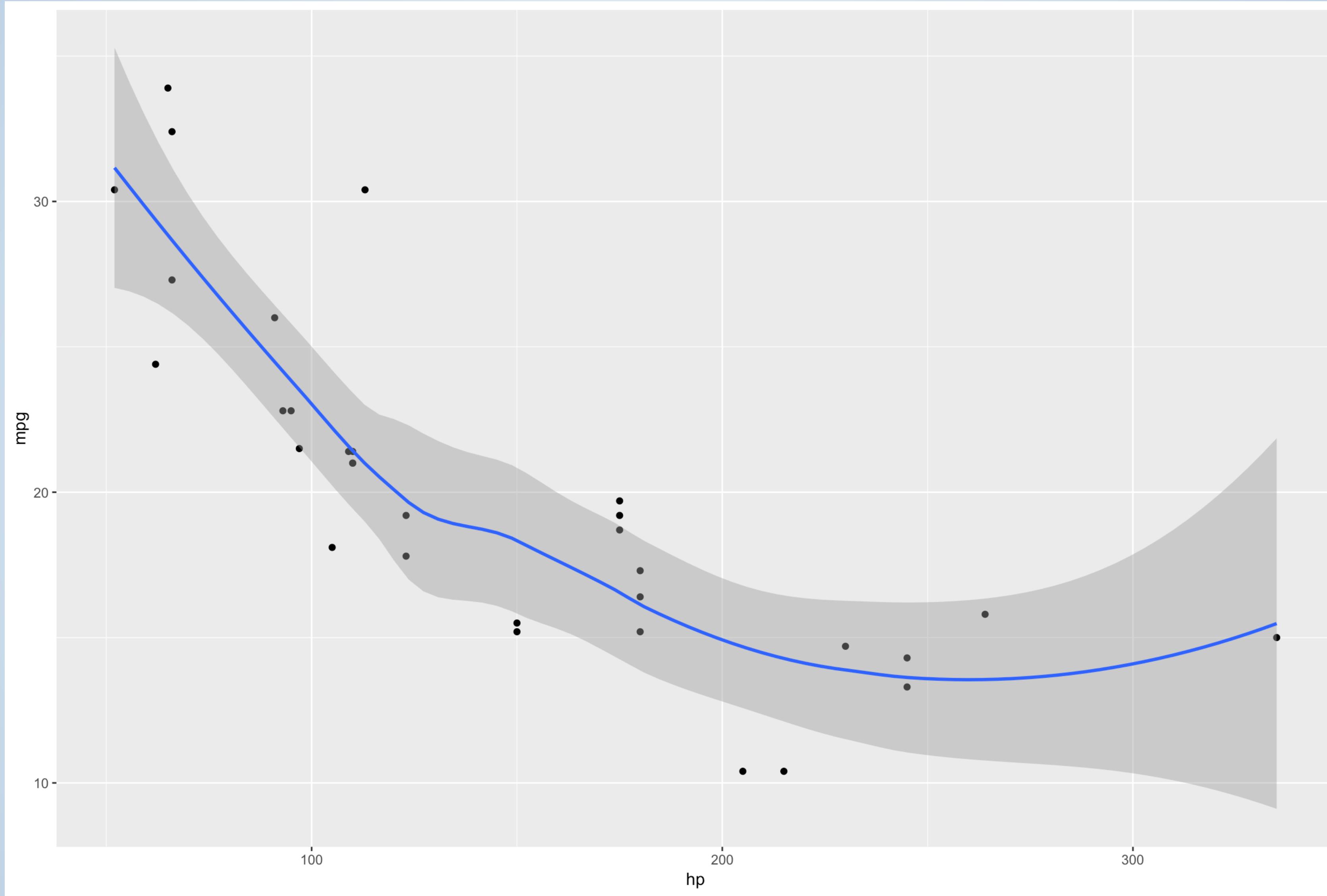
```
<><>  
ggplot(diamonds, aes(x=cut)) +  
  geom_bar()
```





# DEMO

```
ggplot(mtcars, aes(x = hp, y = mpg)) +  
  geom_point() +  
  geom_smooth()
```





# EXERCISE

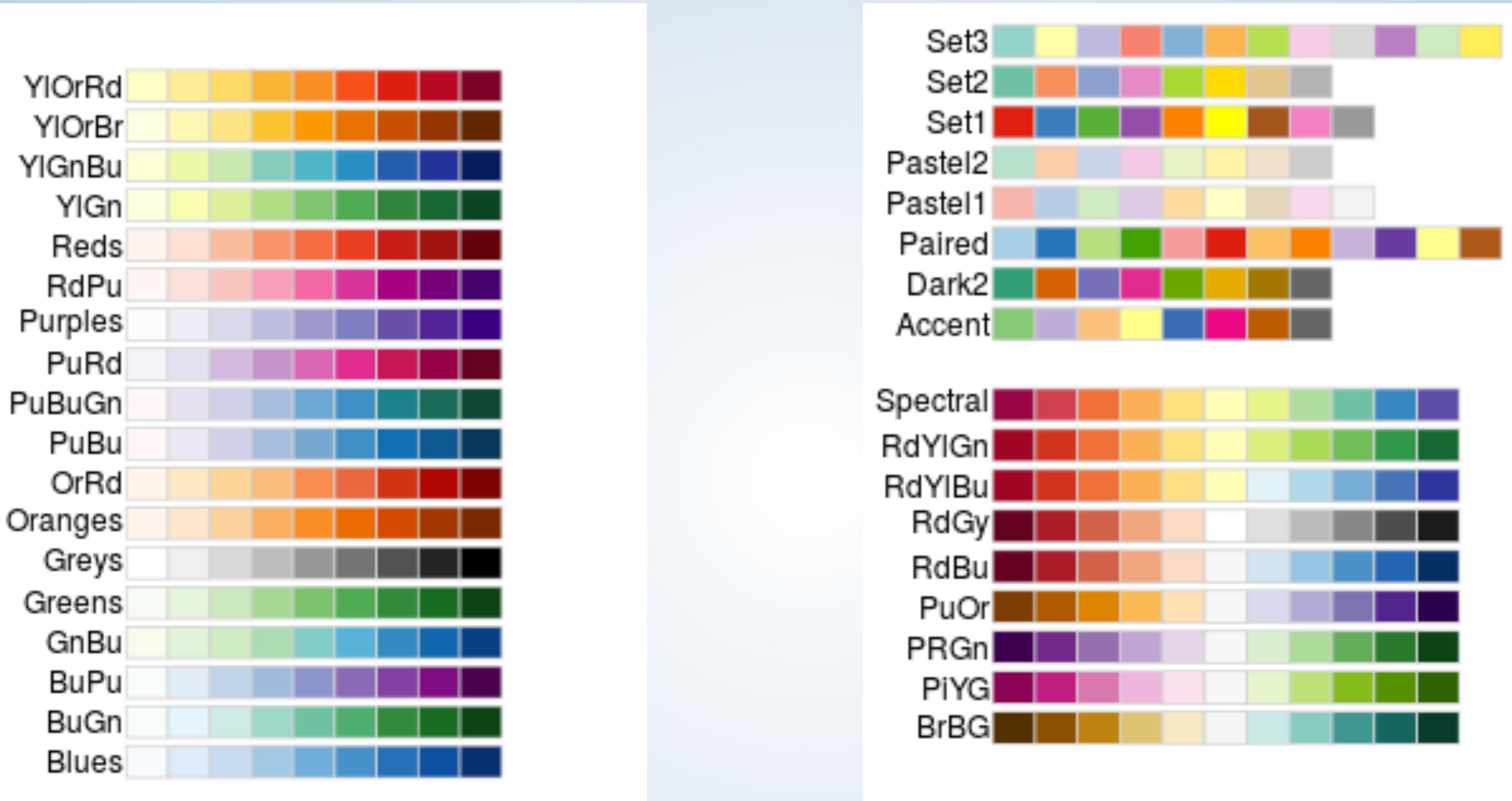
- ▶ Build any plot
  - ▶ Use any dataset

3m 00s



# DEMO

Are there any other plots from the markdown file you would like to see?



<https://colorbrewer2.org/>

plotly

ggplotly



# DEMO

```
ggplotly(  
  ggplot(mtcars, aes(x = hp, y = mpg)) +  
    geom_point() +  
    geom_smooth()  
)
```



# DEMO

```
ggplotly(  
  ggplot(mtcars, aes(x = hp, y = mpg) +  
    geom_point() +  
    ggtitle("Car Miles per Gallon by Horse Power") +  
    xlab("Horse Power") +  
    ylab("MPG"),  
  tooltip = c("x", "y"))  
)
```



# DEMO

```
ggplotly(  
  ggplot(mtcars, aes(x = hp, y = mpg, text = paste("<b>", rowname, "</b><br>",  
    "MPG:", mpg, "<br>Horse Power:", hp))) +  
  geom_point() +  
  ggtitle("Car Miles per Gallon by Horse Power") +  
  xlab("Horse Power") +  
  ylab("MPG"),  
  tooltip = "text"  
)
```



# EXERCISE

- ▶ Warp your previous ggplot
  - ▶ Create a tooltip
  - ▶ Run the code and mess around with the user options in plotly

5m 00s

# OTHER FEATURES IN PLOTLY

- Users can save their output by clicking the camera
- You can set tooltip popup type ahead of time
  - ie: `%>% layout(hovermode = 'compare')`
- You can even combine two plots
  - ie: `subplot(plot1, plot2, nrows = 2, shareX = TRUE, heights = c(.7, .3))`
- Plotly will recognize most ggplot arguments, but some don't always come over easily.

plotly standalone

### Line Plots

```
plot_ly(  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter' ,  
  mode = 'lines' )
```

### Bubble Charts

```
plot_ly(  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter' ,  
  mode = 'markers' ,  
  size = c( 1, 5, 10 ),  
  marker = list(  
    color = c( 'red', 'blue' ,  
    'green' )))
```

### Scatter Plots

```
plot_ly(  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter' ,  
  mode = 'markers' )
```

### Heatmaps

```
plot_ly(  
  z = volcano ,  
  type = 'heatmap' )
```

### Histograms

```
x <- rchisq ( 100, 5, 0 )  
plot_ly(  
  x = x ,  
  type = 'histogram' )
```

### 3D Surface Plots

```
# Using a dataframe:  
plot_ly(  
  type = 'surface' ,  
  z = ~volcano )
```

### Box Plots

```
plot_ly(  
  y = rnorm( 50 ),  
  type = 'box' ) %>%  
  
add_trace(y = rnorm( 50, 1 ))
```

### 3D Line Plots

```
plot_ly(  
  type = 'scatter3d' ,  
  x = c( 9, 8, 5, 1 ) ,  
  y = c( 1, 2, 4, 8 ) ,  
  z = c( 11, 8, 15, 3 ) ,  
  mode = 'lines' )
```

### Bar Charts

```
plot_ly(  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'bar' ,  
  mode = 'markers' )
```

### Area Plots

```
plot_ly(  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter' ,  
  mode = 'lines' ,  
  fill = 'tozerooy' )
```

### 2D Histogram

```
plot_ly(  
  x = rnorm( 1000, sd = 10 ) ,  
  y = rnorm( 1000, sd = 5 ) ,  
  type = 'histogram2d' )
```

### 3D Scatter Plots

```
plot_ly(  
  type = 'scatter3d' ,  
  x = c( 9, 8, 5, 1 ) ,  
  y = c( 1, 2, 4, 8 ) ,  
  z = c( 11, 8, 15, 3 ) ,  
  mode = 'markers' )
```

plotly  
in an app

DEMO



Go to old\_faithful.R



# EXERCISE

- ▶ Go to movies\_20.R
  - ▶ Edit the UI and Server functions so that scatter plot uses plotly
  - ▶ Create a useful tooltip

5m 00s



# SOLUTION

- ▶ Check your app against movies\_21.R
- ▶ Add `library(plotly)`
  - ▶ Change `plotOutput` to `plotlyOutput`
  - ▶ Change `renderPlot` to `renderPlotly`
  - ▶ Wrap the plot in `ggplotly()`
    - ▶ Place tooltip as a paste function in `aes()` and  
`set tooltip = "text"`
    - ▶ Or set `tooltip = c("x", "y")`

DT

Advanced

# DT OPTIONS AND FUNCTIONS

- ▶ <https://rstudio.github.io/DT/options.html>
  - ▶ Determine which elements of the table to display (dom)
  - ▶ Scrolling
  - ▶ Selection
  - ▶ Pages
  - ▶ etc.
- ▶ <https://rstudio.github.io/DT/functions.html>
  - ▶ Format Rows based off values
  - ▶ Set row format
  - ▶ etc.

# DT EXTENSIONS

- ▶ <https://rstudio.github.io/DT/extensions.html>
- ▶ Buttons
  - ▶ Download data (only works on data displayed)
- ▶ Fixed column/header
  - ▶ Keeps certain columns or header from scrolling
- ▶ Scroller

# INTERACTIVE DATA VISUALIZATIONS WITH PLOTLY

