Data Scientifique

GradProSkills

Introduction to R: Day 3
Data Visualization

Instructor: Yara Abu Awad

Workshop Schedule

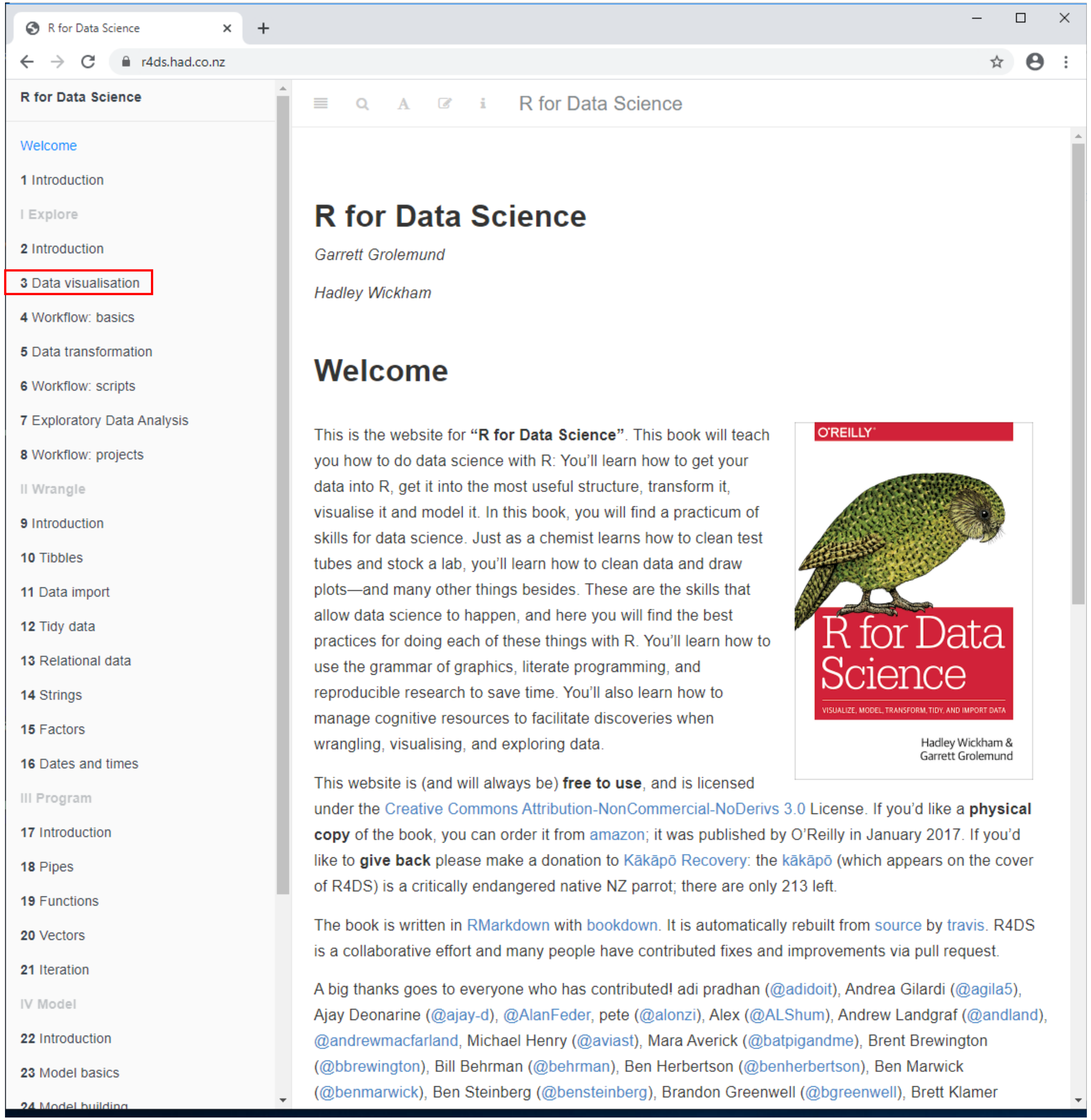
- In this workshop, the following topics will be covered:
  - Mastering R basics: will include an introduction to the R environment, packages and data types
  - Describing data: will demonstrate how to generate descriptive statistics, table outputs and simple statistical tests (i.e. t tests)
  - Visualizing data: will show participants how to generate multiple types of plots and charts

Plan for today

- First 15 minutes will consist of a quick review
- Go to https://github.com/YaraRAA/DataSci_IntrotoR
- Scroll down to: **README.md**
- You will see some poll links with numbers
- Let’s begin!

Plan for today ctd…

- Then….
- I will talk for an hour (if you're lucky, maybe less) about how to draw plots
    - using base R
    - using ggplot2
- You will get a chance to practice afterwards with the exercises provided. Again, I recommend that you work in pairs.


R for Data Science
r4ds.had.co.nz
R for Data Science
O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund

R for Data Science

Garrett Grolemund

Hadley Wickham

Welcome

This is the website for **"R for Data Science"**. This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you'll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You'll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You'll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualising, and exploring data.

This website is (and will always be) **free to use**, and is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License. If you'd like a **physical copy** of the book, you can order it from amazon; it was published by O'Reilly in January 2017. If you'd like to **give back** please make a donation to Kākāpō Recovery: the kākāpō (which appears on the cover of R4DS) is a critically endangered native NZ parrot; there are only 213 left.

The book is written in RMarkdown with bookdown. It is automatically rebuilt from source by travis. R4DS is a collaborative effort and many people have contributed fixes and improvements via pull request.

A big thanks goes to everyone who has contributed! adi pradhan (@adidoit), Andrea Gilardi (@agila5), Ajay Deonarine (@ajay-d), @AlanFeder, pete (@alonzi), Alex (@ALShum), Andrew Landgraf (@andland), @andrewmacfarland, Michael Henry (@aviast), Mara Averick (@batpigandme), Brent Brewington (@bbrewington), Bill Behrman (@behrman), Ben Herbertson (@benherbertson), Ben Marwick (@benmarwick), Ben Steinberg (@bensteinberg), Brandon Greenwell (@bgreenwell), Brett Klamer

Textbook for ggplot2

- https://r4ds.had.co.nz/
- Chapter 3 of R for Data Science
- Why ggplot2?
  - Very popular package
  - A lot of examples online
  - Part of the tidyverse collection of packages https://www.tidyverse.org/

me →

Some words about the tidyverse

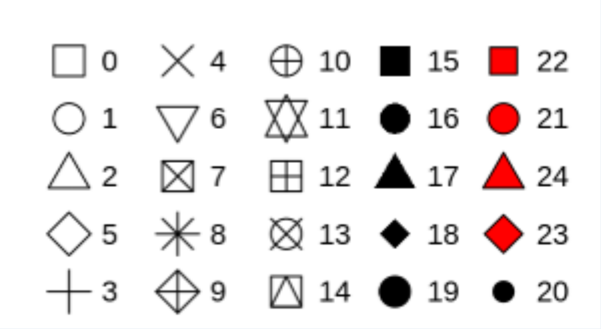
- I personally do not use tidyverse packages, so bear with me
- I also disagree with some of the suggestions in this textbook but feel free to follow it if it works for you!
- FYI, I like the following packages:
  - base R for basic plotting (if I need to quickly check a distribution)
  - mgcv plot of a gam model for the non-linear shape of a dose-response (gam stands for generalized additive model)
  - plotly for fancy plots for publication (they have a lot of easy tutorials online)
  - data.table for data management - data.table is excellent for very large (100 million + rows) datasets

Image source: https://www.amazon.co.uk/3dRose-Inches-Grumposaurus-Dinosaur-mp_104180_1/dp/B00BMCQSQU

Plots using base R

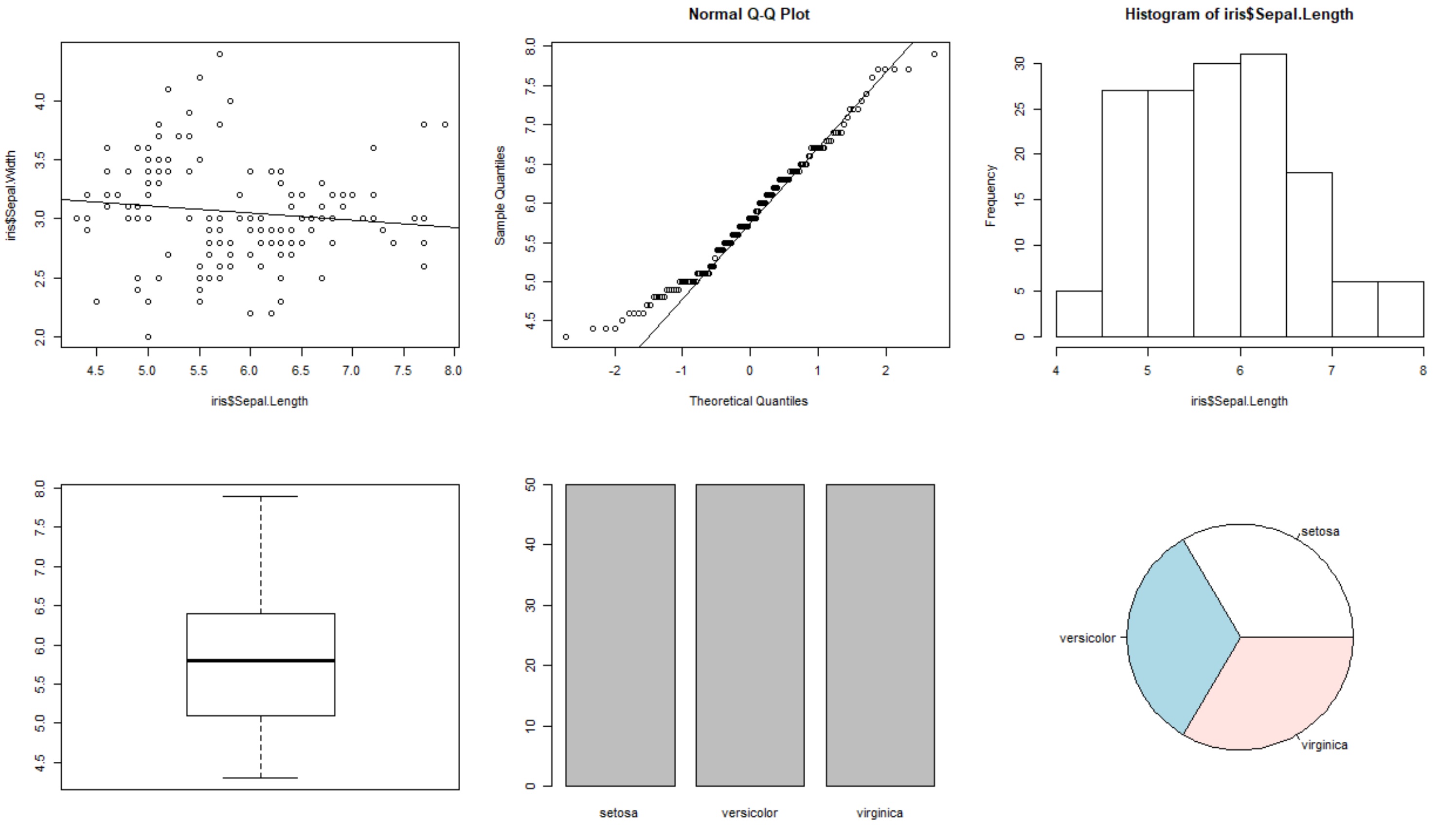
Function	Input	Output
Plot abline	(x , y) 2 numeric vectors i.e. 2 columns (a = constant, b = slope)	Scatter plot Adds one or more straight lines through the current plot
hist	One numeric vector i.e. one column	Histogram of a continuous variable
qqnorm qqline	One numeric vector i.e. one column	Normal qqplot of a continuous variable Adds a line to a “theoretical”, by default normal, quantile-quantile plot
boxplot	One numeric vector i.e. one column	Box plot
pie	A table object	Pie chart
barplot	A table object	Bar plot

Useful arguments in base R plot functions

Argument	Input	What it does
main = , xlab = , ylab =	‘title in quotes’	Adds main title, x axis and y axis title to plot respectively
add =	T	Add this new plot to the previous
xlim =, ylim =	c(min, max)	Sets beginning and end of x axis & y axis respectively
col =	a color in quotations i.e. ‘red’ OR rgb(red,green,blue,transparency)	Determines the colour of the plot
cex =	number	Relative size of text
pch =	number 	Determines shape of points Same numbers are also used in ggplot2 syntax but instead of pch = , the syntax is shape =

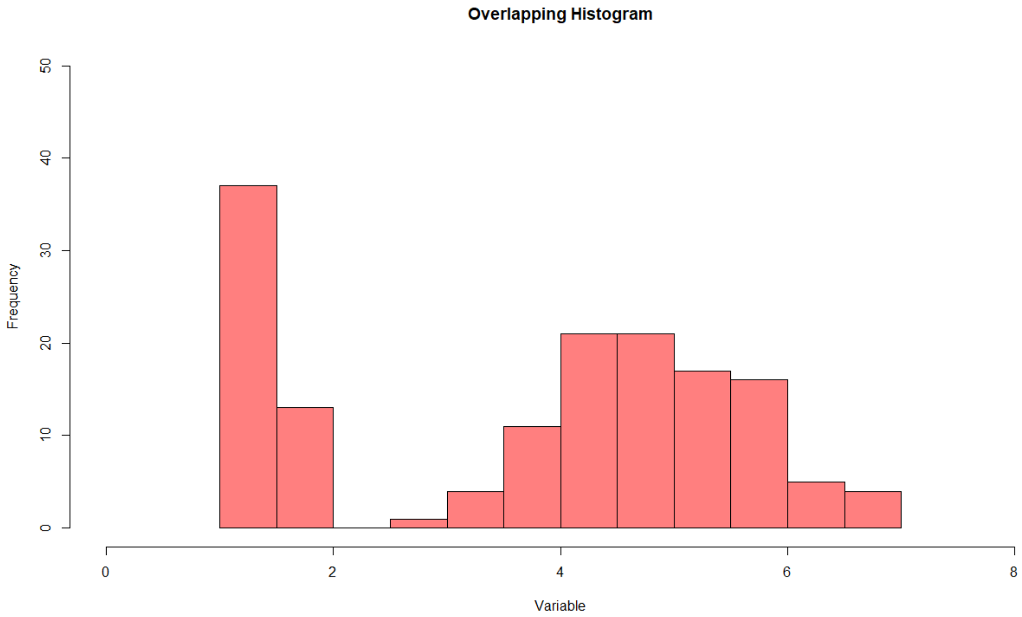
Example: multiple plots in one

```
par(mfrow =c(2,3))
plot(iris$Sepal.Length, iris$Sepal.Width)
abline(lm(Sepal.Width ~ Sepal.Length, data = iris ))
qqnorm(iris$Sepal.Length)
qqline(iris$Sepal.Length)
hist(iris$Sepal.Length)
boxplot(iris$Sepal.Length)
barplot(table(iris$Species))
pie(table(iris$Species))
```

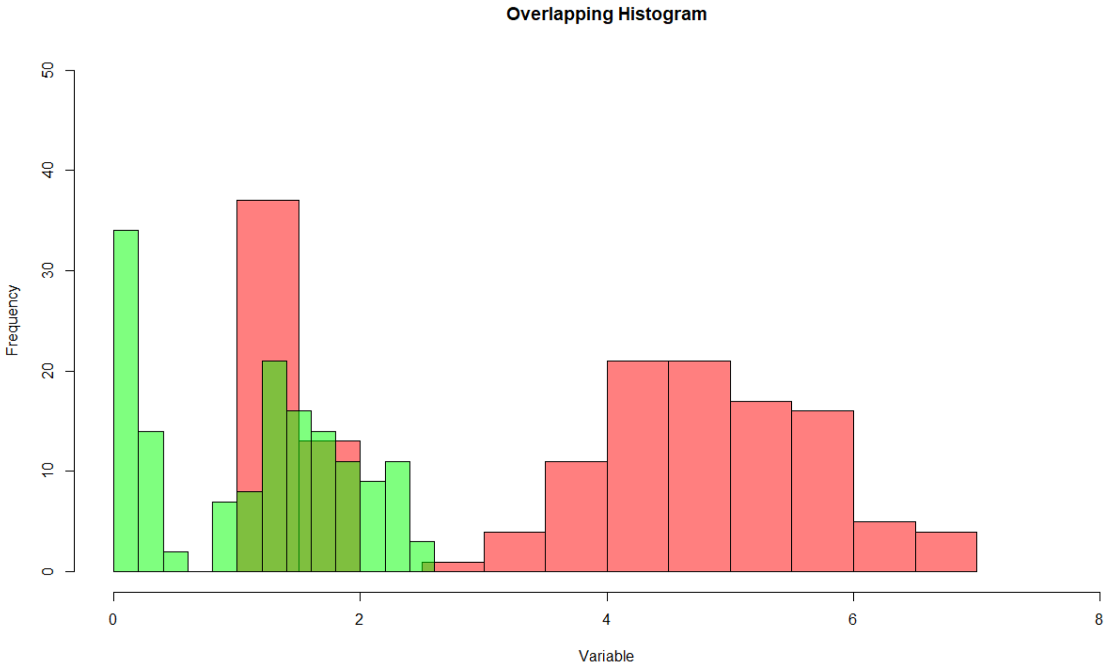


Example: overlaying plots

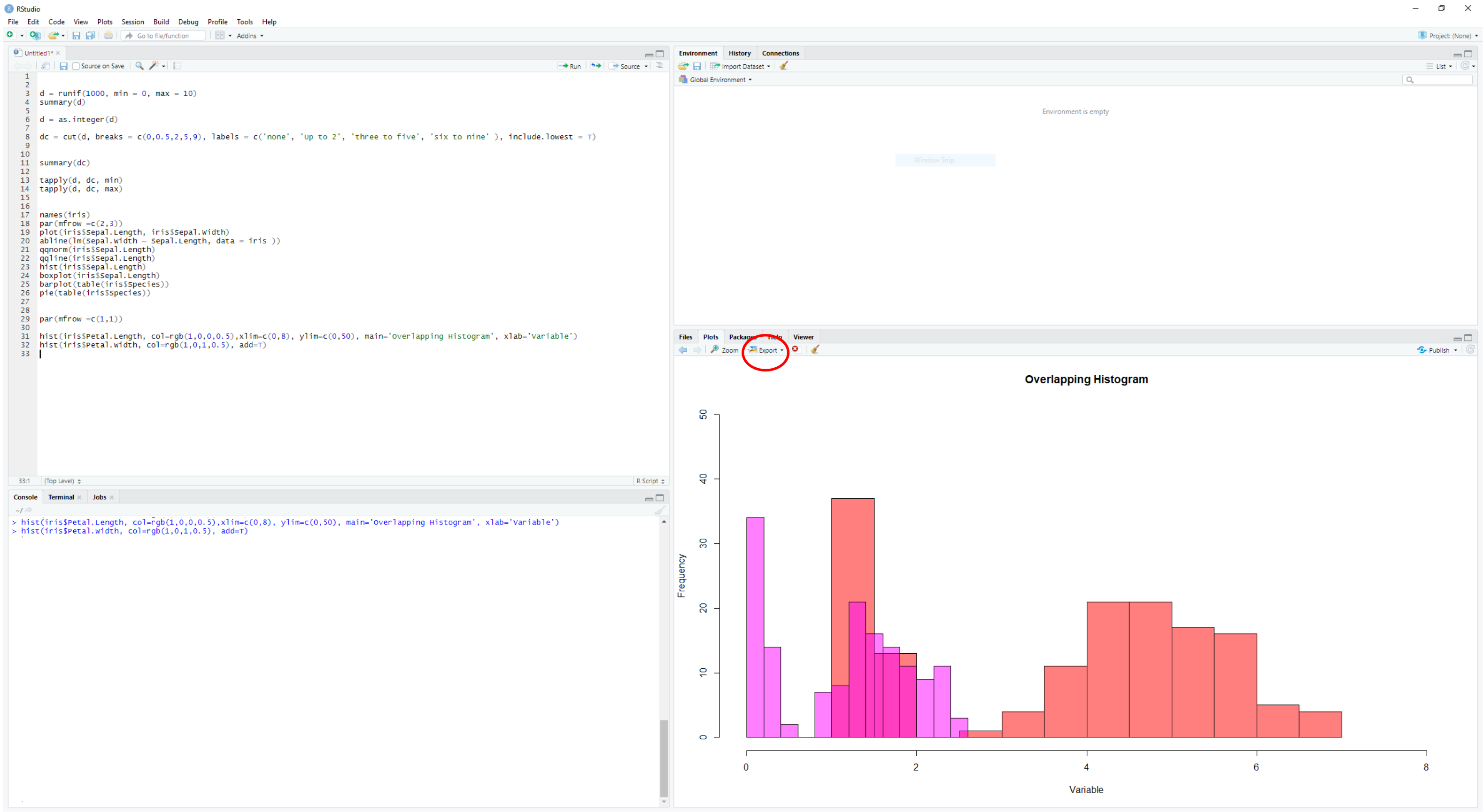
```
hist(iris$Petal.Length, col=rgb(1,0,0,0.5),xlim=c(0,8),
ylim=c(0,50), main='Overlapping Histogram', xlab='Variable')
```



```
hist(iris$Petal.Width, col=rgb(0,1,0,0.5), add=T)
```



Saving your plot



```


d = runif(1000, min = 0, max = 10)
summary(d)

d = as.integer(d)

dc = cut(d, breaks = c(0,0.5,2,5,9), labels = c('none', 'Up to 2', 'three to five', 'six to nine' ), include.lowest = T)


summary(dc)

tapply(d, dc, min)
tapply(d, dc, max)


names(iris)
par(mfrow =c(2,3))
plot(iris$Sepal.Length, iris$Sepal.Width)
abline(lm(Sepal.Width ~ Sepal.Length, data = iris ))
qqnorm(iris$Sepal.Length)
qqline(iris$Sepal.Length)
hist(iris$Sepal.Length)
boxplot(iris$Sepal.Length)
barplot(table(iris$Species))
pie(table(iris$Species))


par(mfrow =c(1,1))

hist(iris$Petal.Length, col=rgb(1,0,0,0.5),xlim=c(0,8), ylim=c(0,50), main='Overlapping Histogram', xlab='Variable')
hist(iris$Petal.Width, col=rgb(1,0,1,0.5), add=T)
```

```
> hist(iris$Petal.Length, col=rgb(1,0,0,0.5),xlim=c(0,8), ylim=c(0,50), main='Overlapping Histogram', xlab='Variable')
> hist(iris$Petal.Width, col=rgb(1,0,1,0.5), add=T)
```

Alternatively, before plotting run code:

pdf('nameoffile.pdf')

plot(x, y)

dev.off()

Saves your plot to a pdf file named 'nameoffile.pdf' in your working directory.

Now to ggplot2

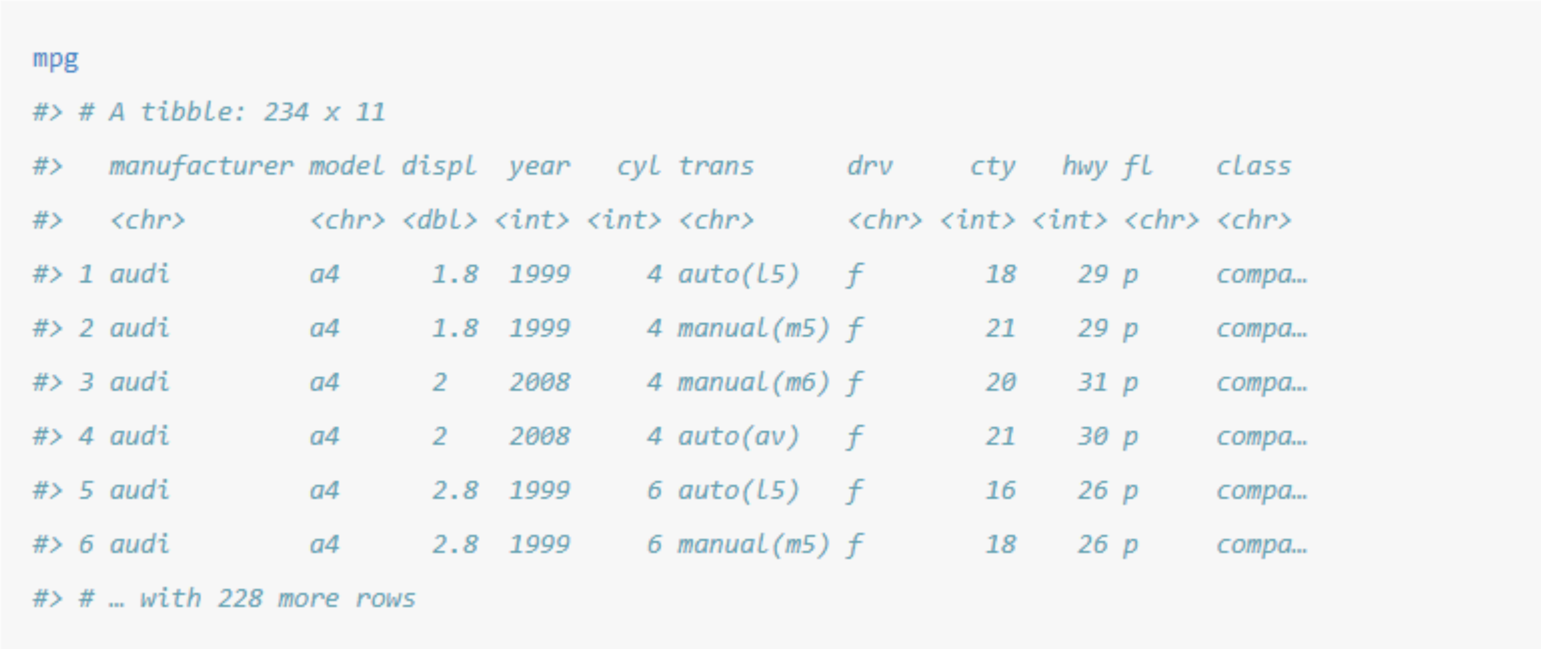
- First install the tidyverse packages and then load them:

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

- Alternatively:

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

We will be using the mpg data frame which is found in ggplot2



```
mpg
#> # A tibble: 234 x 11
#>   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa…
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
#> # … with 228 more rows
```

Among the variables in mpg are:

1. displ, a car’s engine size, in litres.

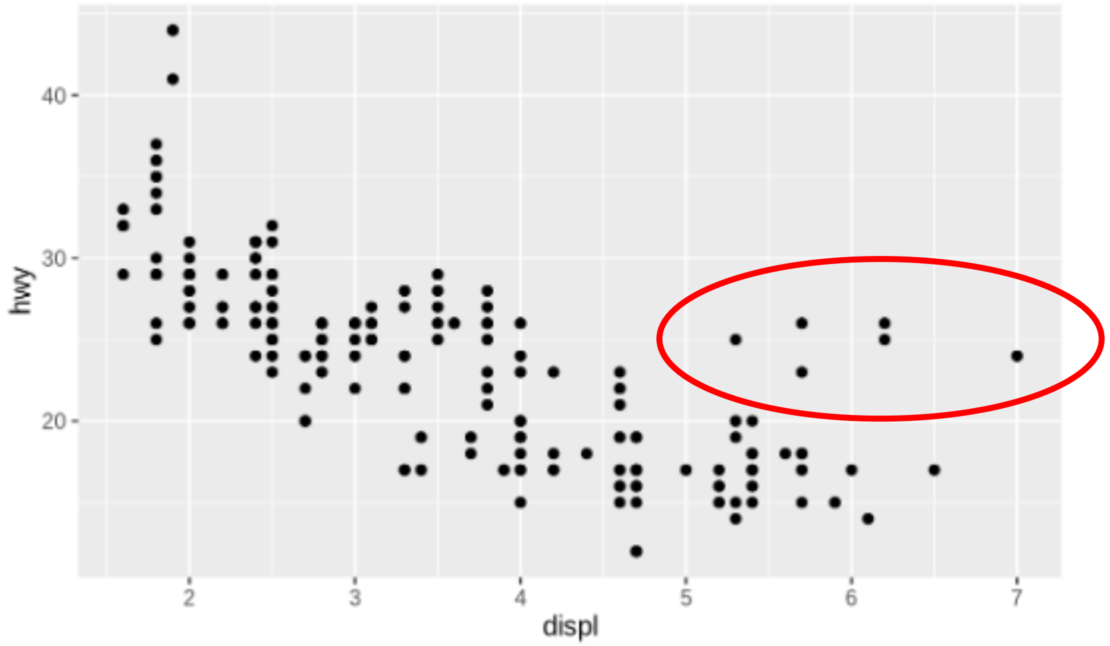
2. hwy, a car’s fuel efficiency on the highway, in miles per gallon (mpg). A car with a low fuel efficiency consumes more fuel than a car with a high fuel efficiency when they travel the same distance.

Creating a ggplot

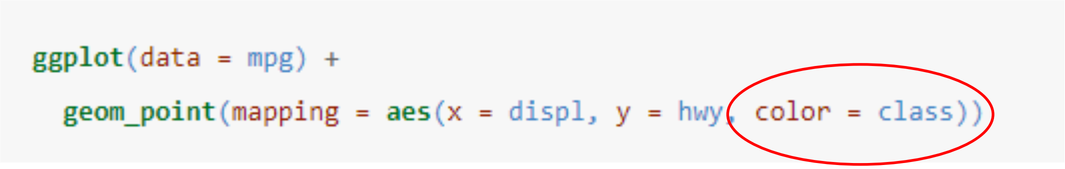
- To plot mpg, run this code to put displ on the x-axis and hwy on the y-axis:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

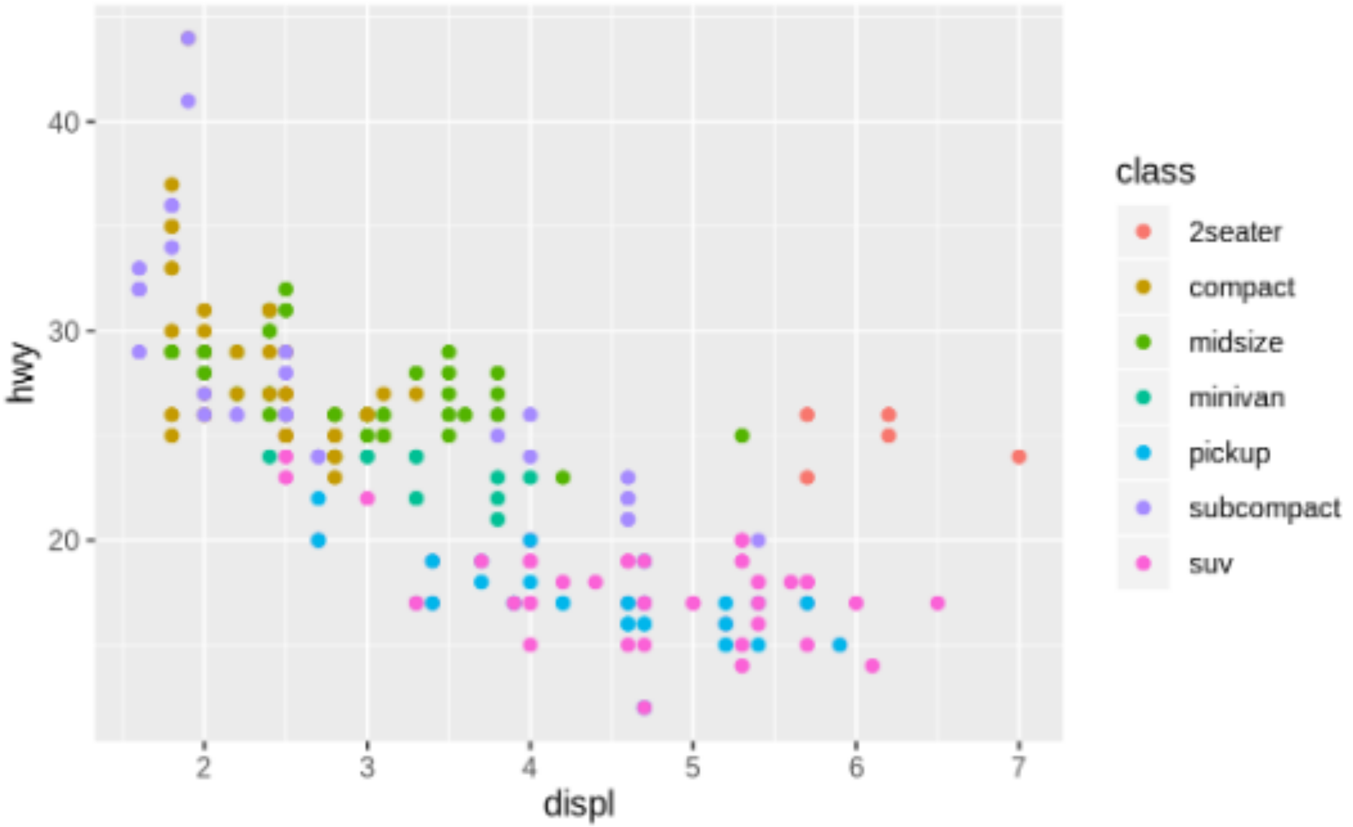
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy))
```



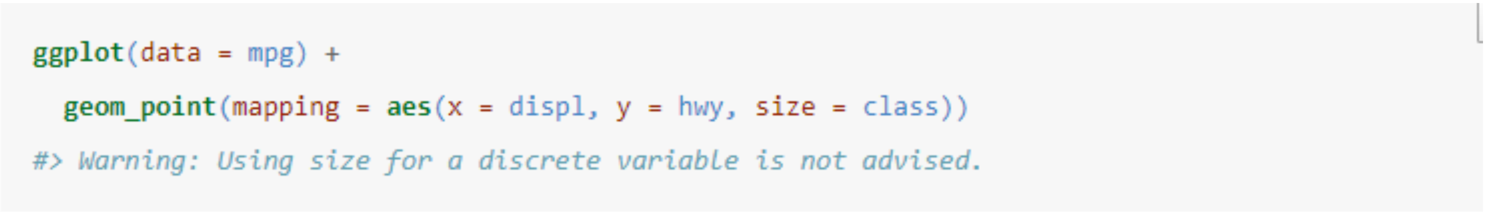
Colour coded point data



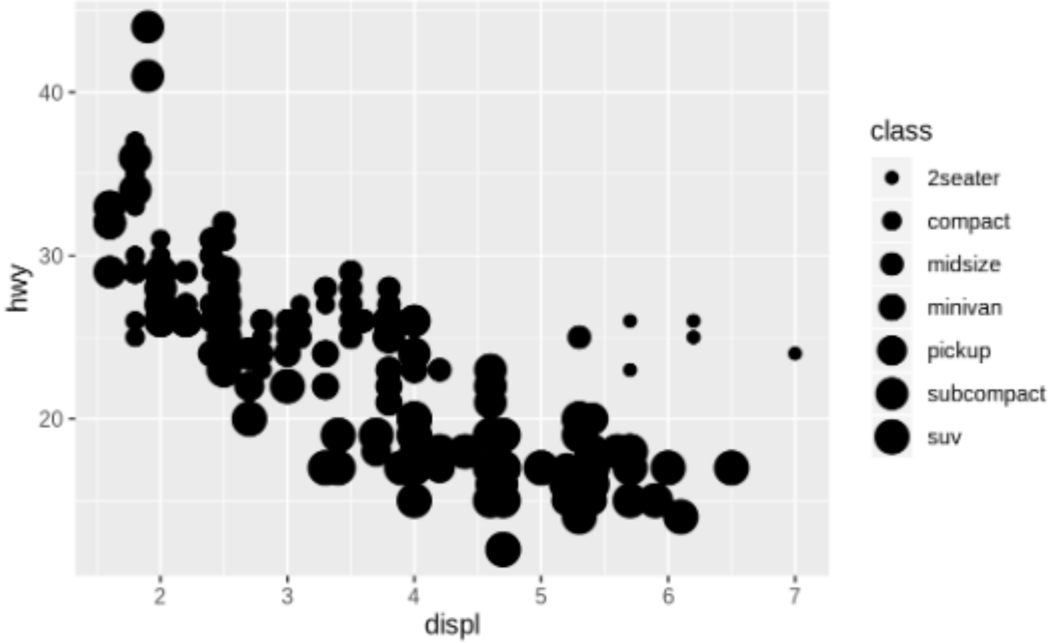
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



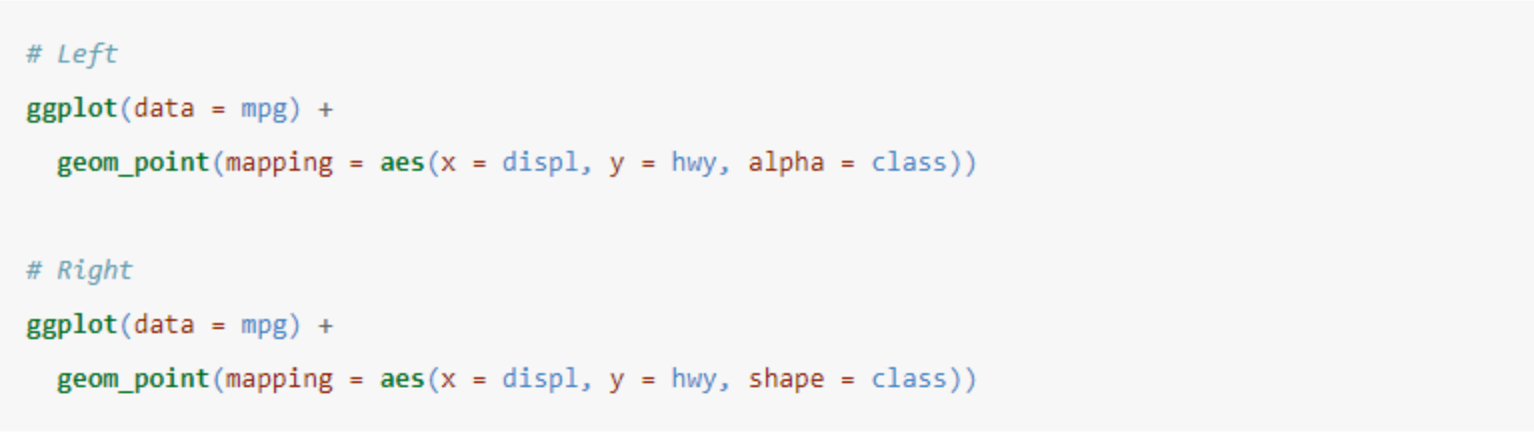
Indicating class by point size



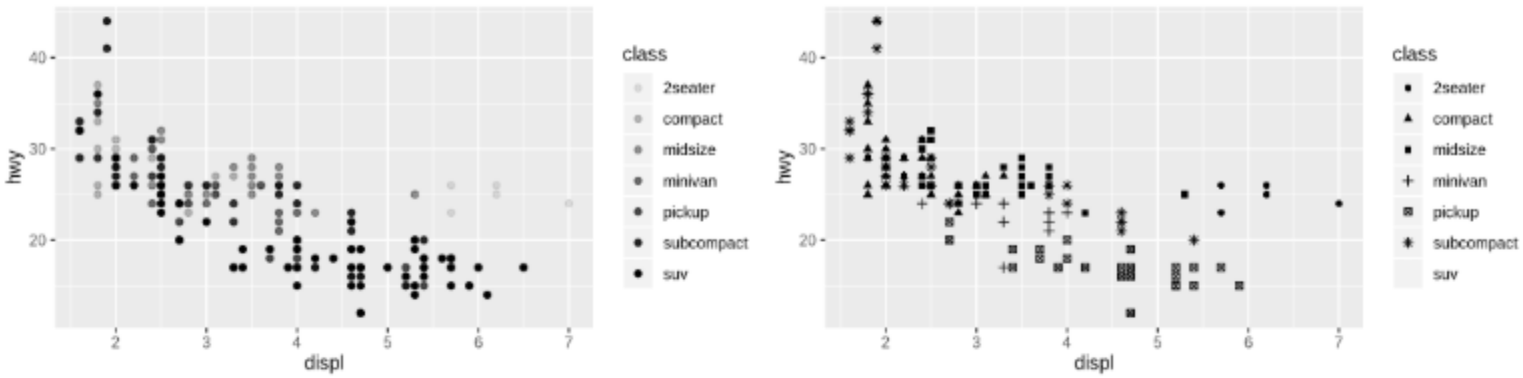
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
#> Warning: Using size for a discrete variable is not advised.
```



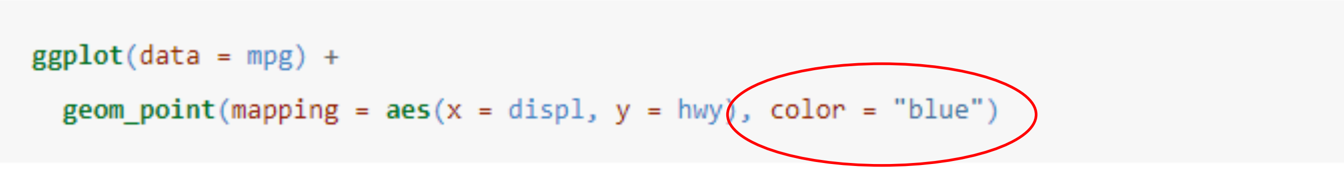
Indicating class by point shape / transparency



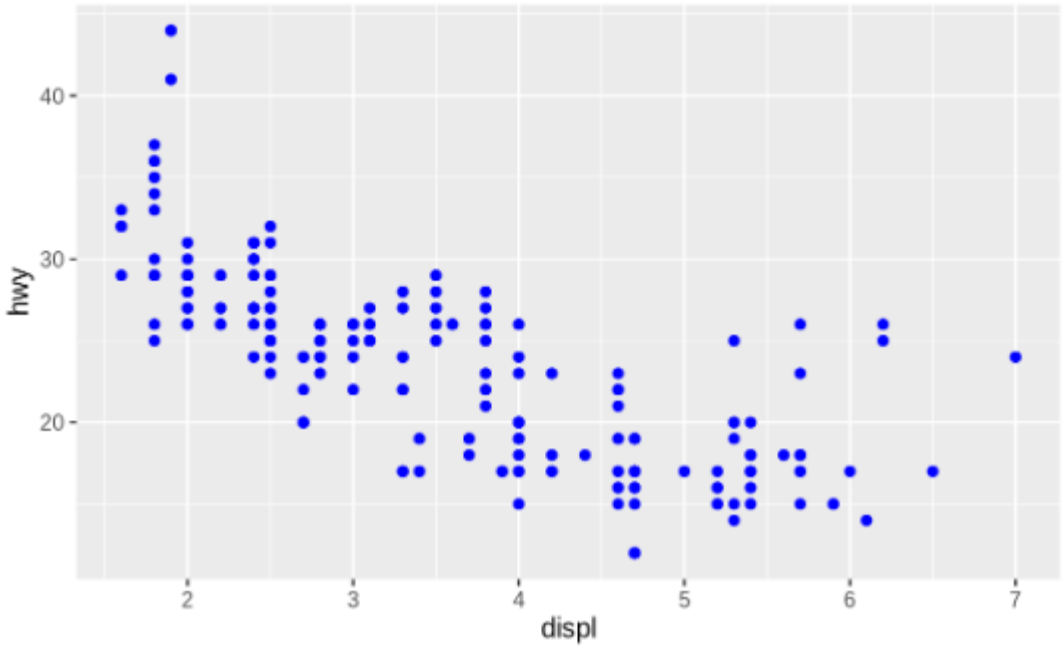
```
# Left
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))

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



Selecting the color of your points



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

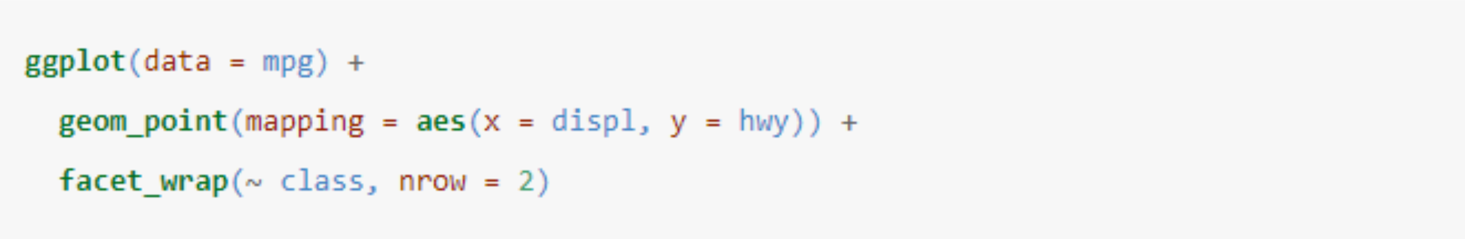


Avoiding errors in your code

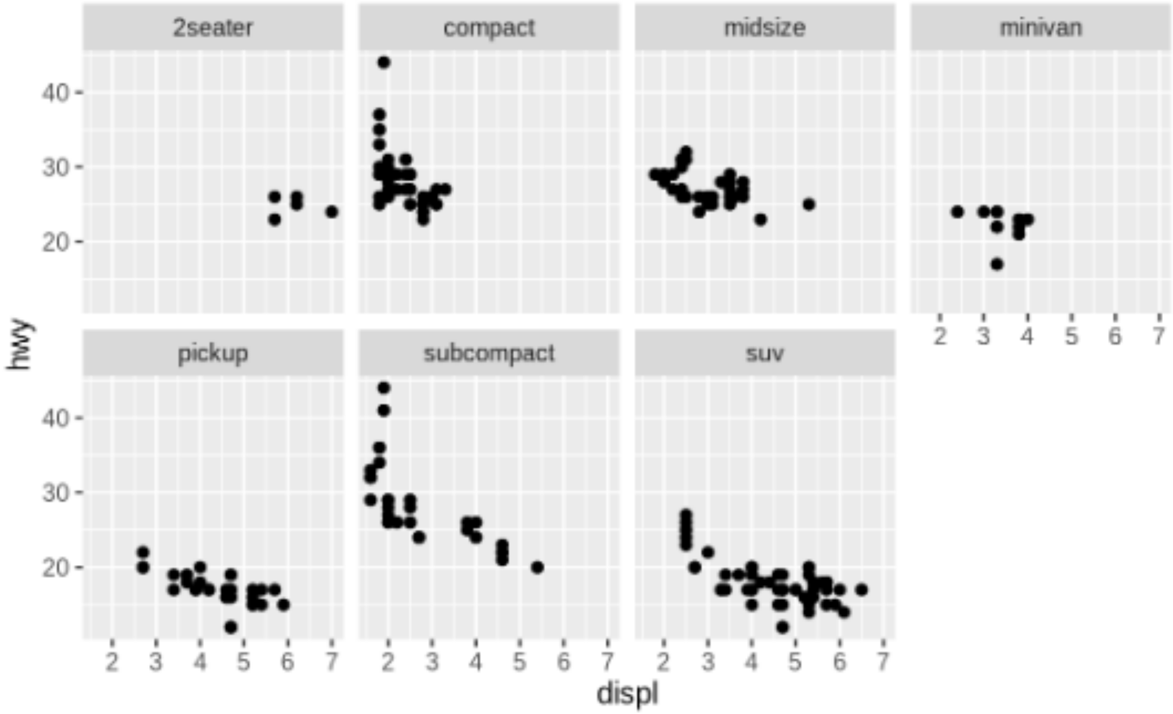
- Make sure:
    - you have entered the code exactly as it is presented
    - commas are in the right place
    - every opening quote " has a closing quote "
    - every open bracket (has a closing bracket)
    - specifically in ggplot2, the + has to come at the end of the line

- If you run code and nothing happens, you will see a + sign in your console. This means that R is waiting for you to close a function that you started (conversely, > means that R is ready for a new command):
    - when this happens, hit the escape button to abort the current command and check your code

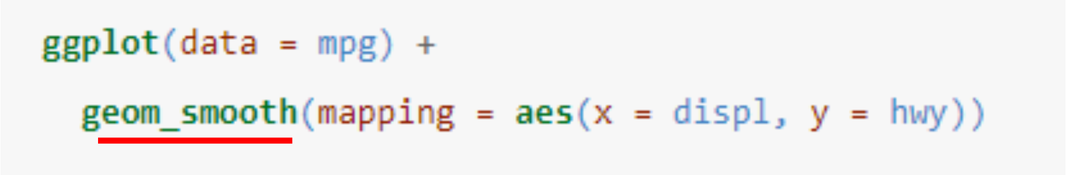
Plotting subsets of data



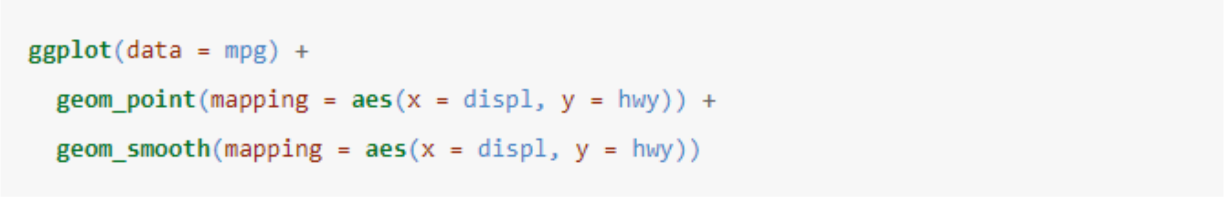
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



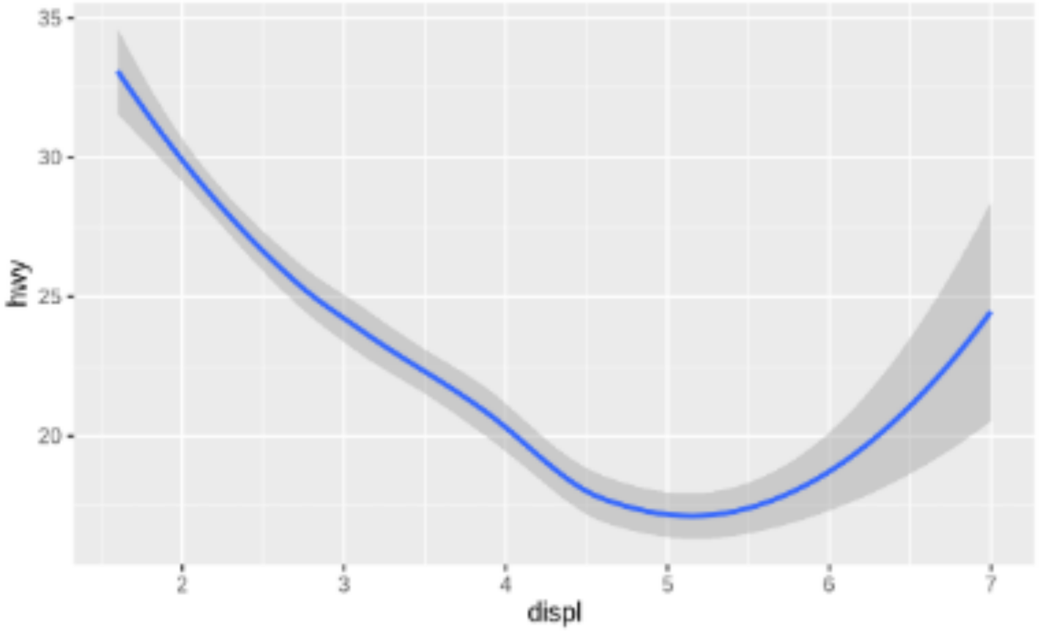
Applying a smooth function to your scatter plot

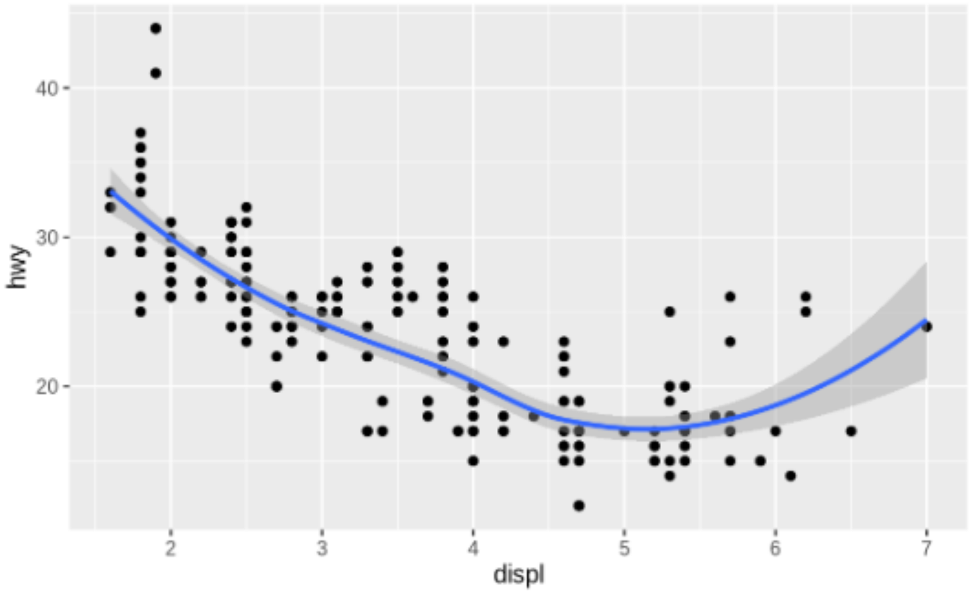


```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



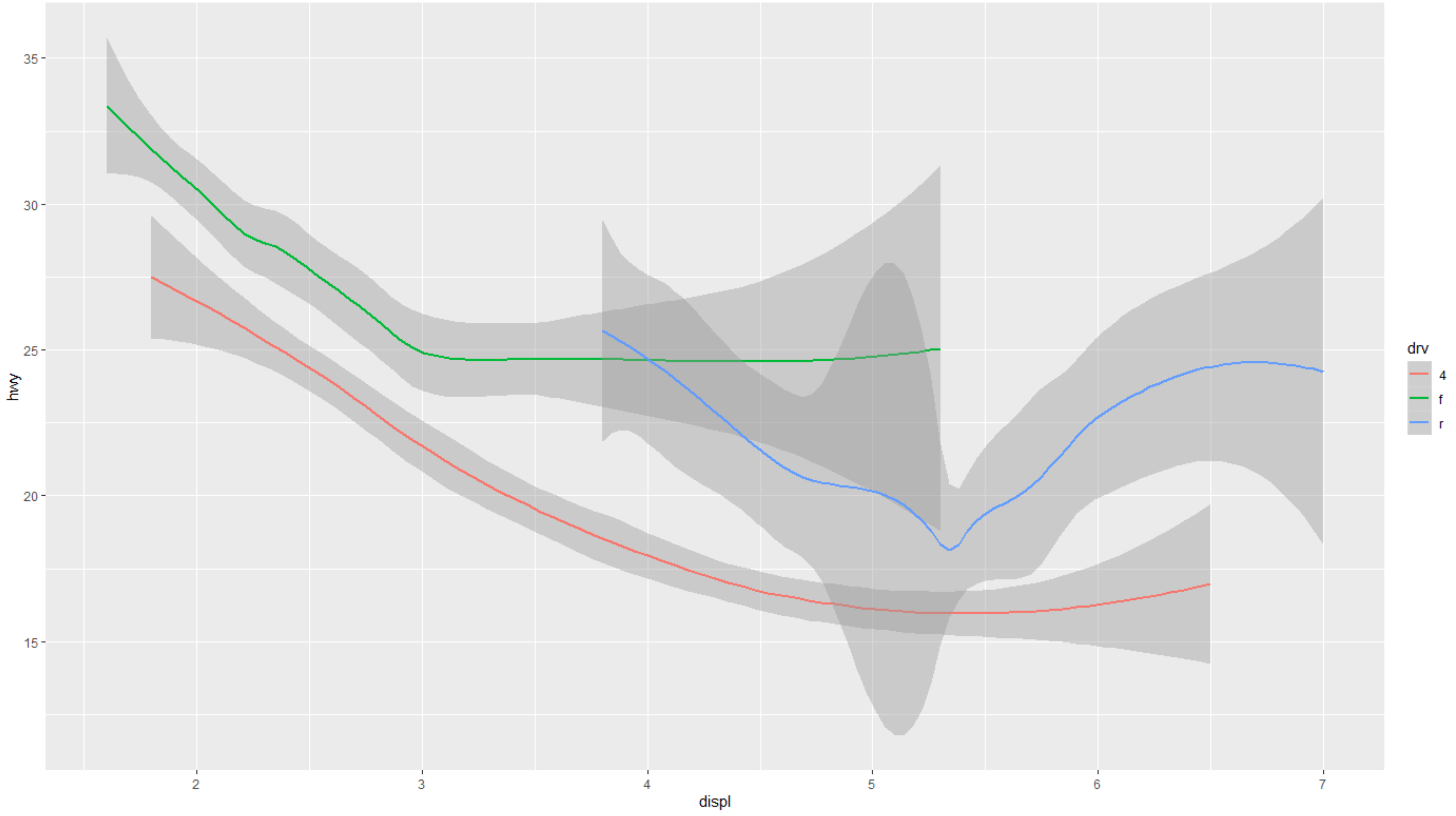
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```





Apply smooth functions to different subsets

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

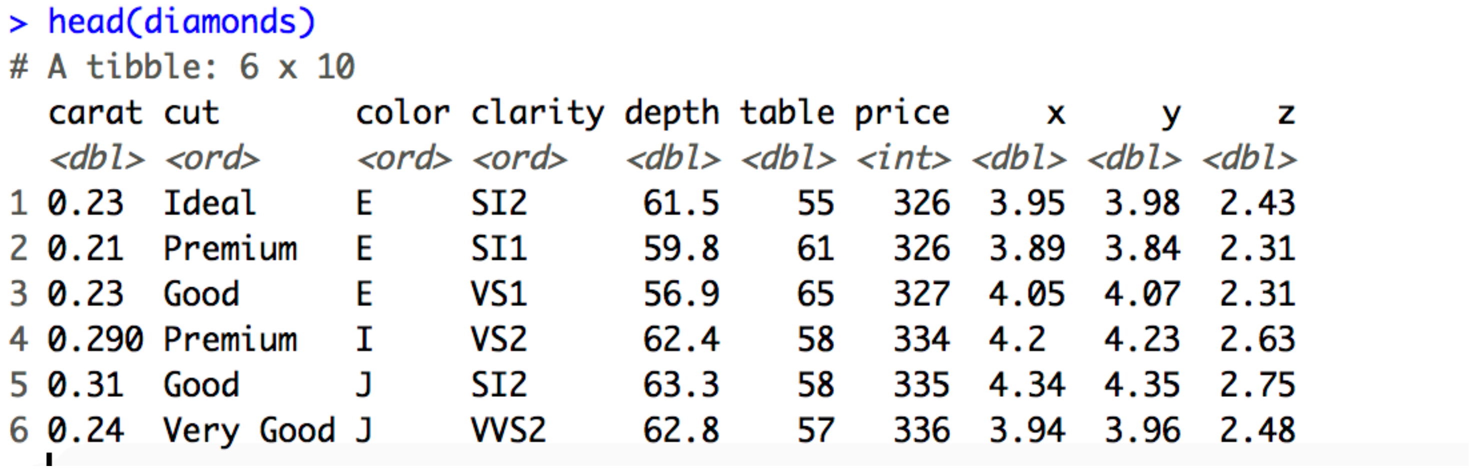


drv
f = front-wheel drive,
r = rear wheel drive,
4 = 4wd

Note: by default, geom_smooth Uses mgcv gam when there are > 1000 observations

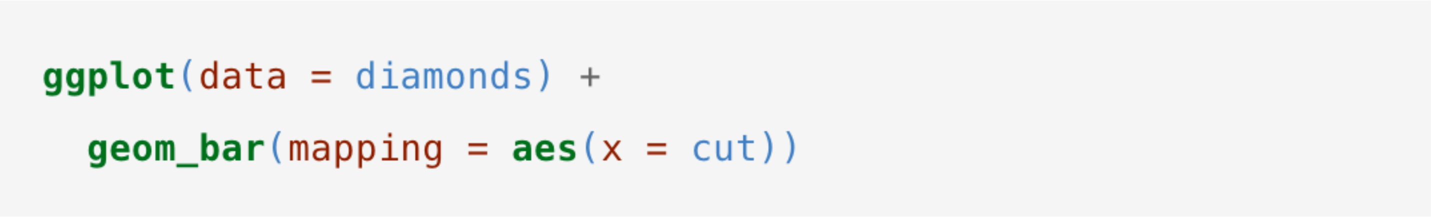
What about categorical data?

- Data frame diamonds in ggplot2

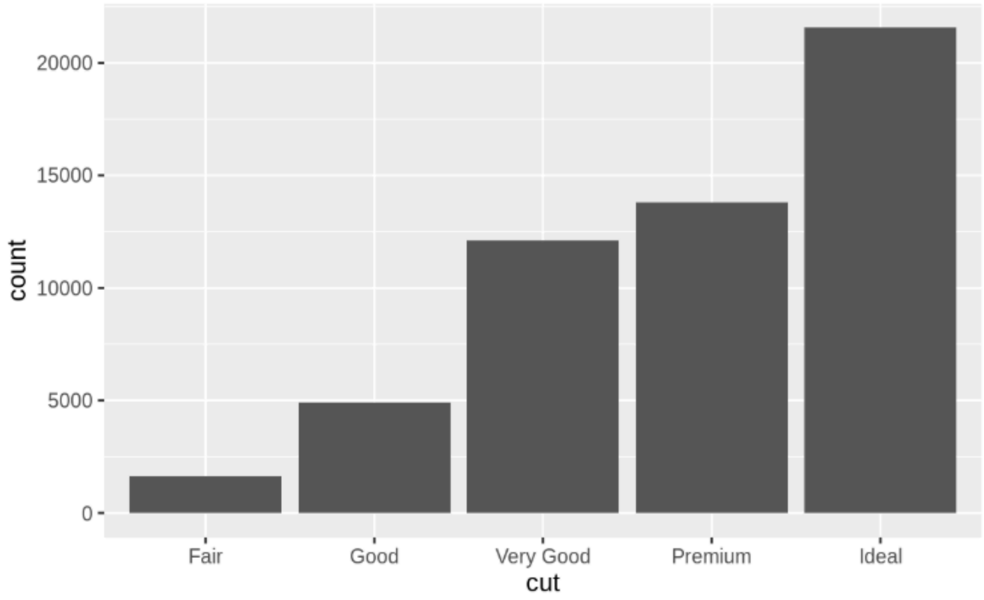


```
> head(diamonds)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23  Ideal     E     SI2      61.5    55   326 3.95  3.98  2.43
2 0.21  Premium   E     SI1      59.8    61   326 3.89  3.84  2.31
3 0.23  Good      E     VS1      56.9    65   327 4.05  4.07  2.31
4 0.290 Premium   I     VS2      62.4    58   334 4.2   4.23  2.63
5 0.31  Good      J     SI2      63.3    58   335 4.34  4.35  2.75
6 0.24  Very Good J     VVS2     62.8    57   336 3.94  3.96  2.48
```

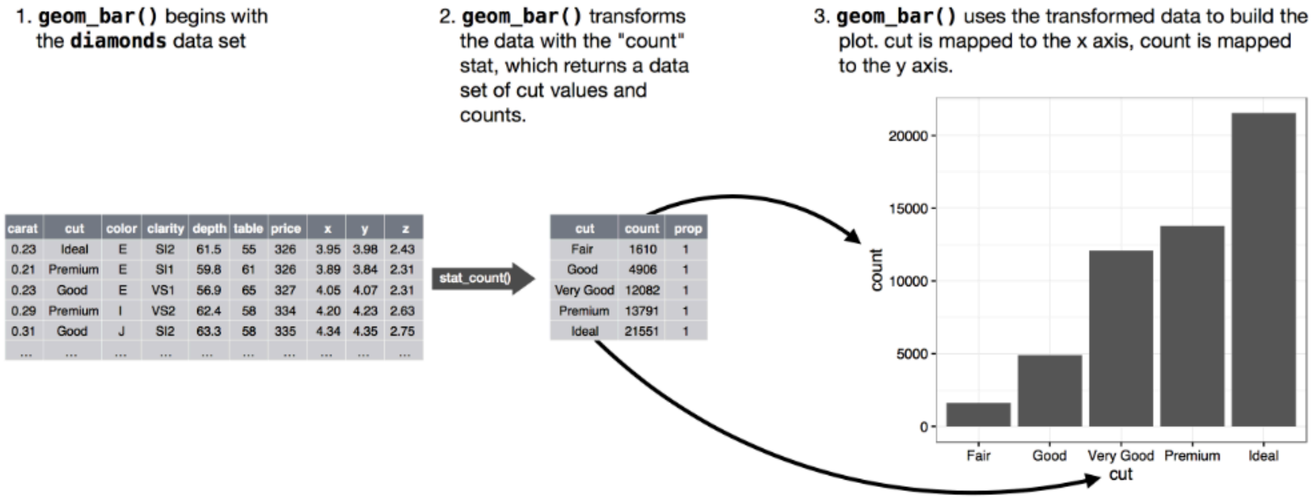
Bar plot: Frequency of diamonds by quality of cut



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



Bar plot: A count variable is created!



1. **geom_bar()** begins with the **diamonds** data set

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

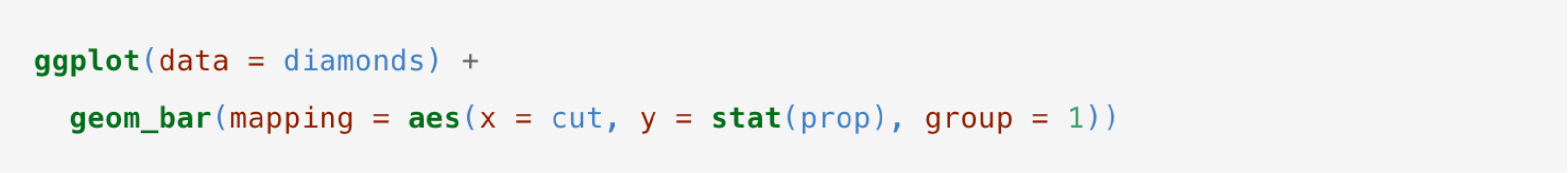
stat_count()

2. **geom_bar()** transforms the data with the "count" stat, which returns a data set of cut values and counts.

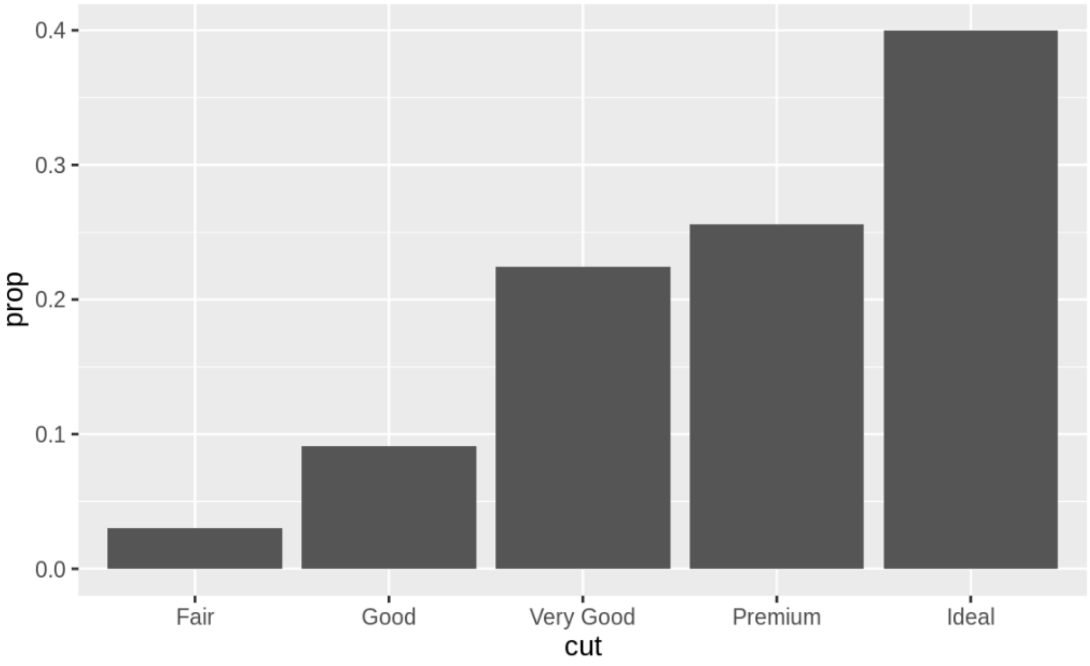
cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

3. **geom_bar()** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.

Bar plot: what if I want to plot proportion instead?

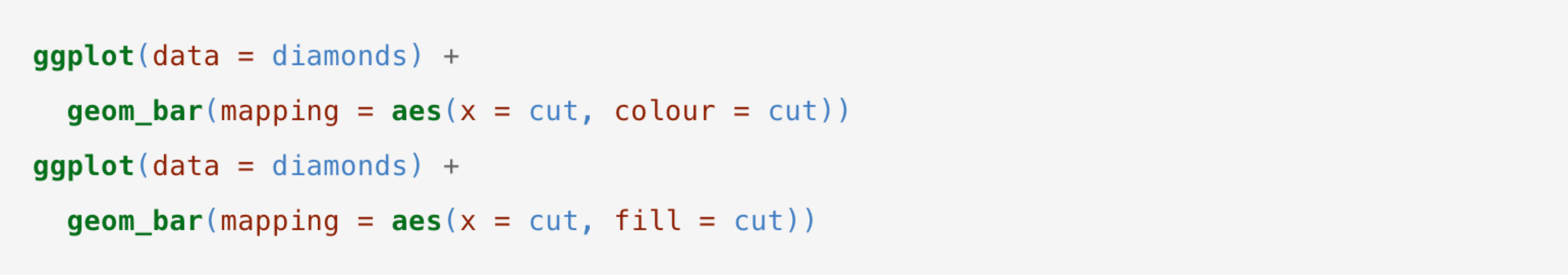


```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = stat(prop), group = 1))
```

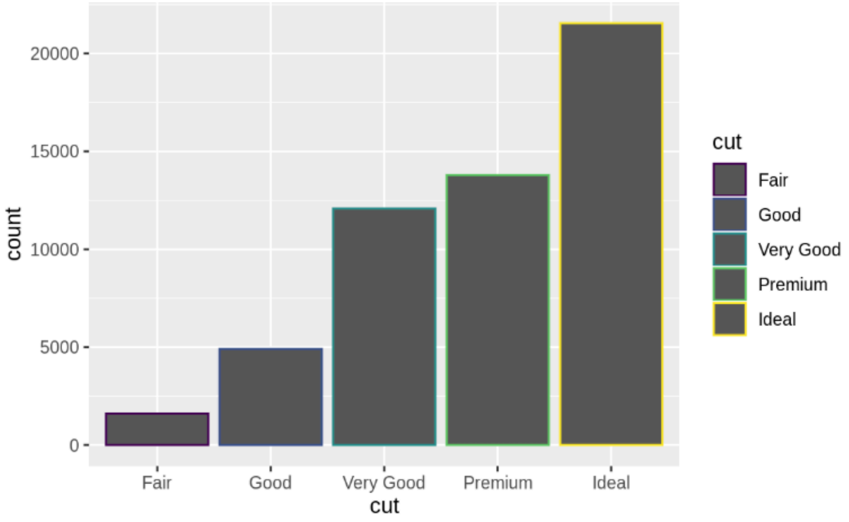


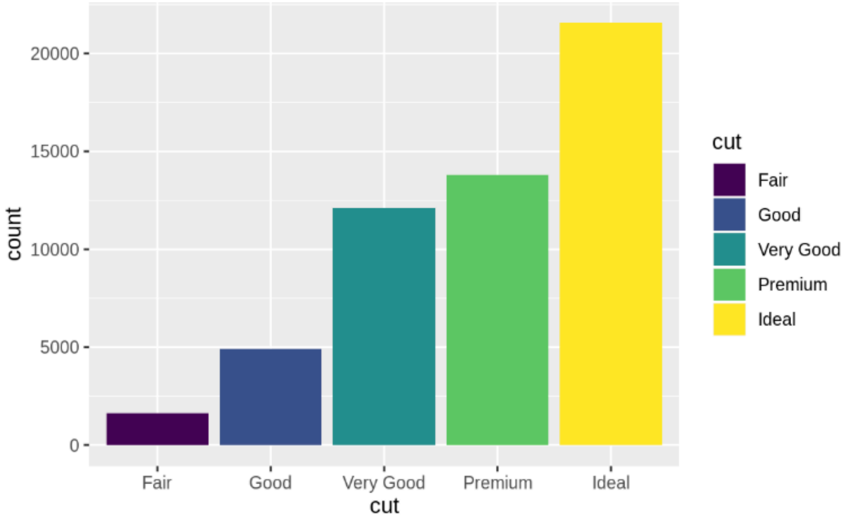
Note: ’ggplot2 provides over 20 stats for you to use. Each stat is a function, so you can get help in the usual way, e.g. ?stat_bin’

Bar plot: adding color



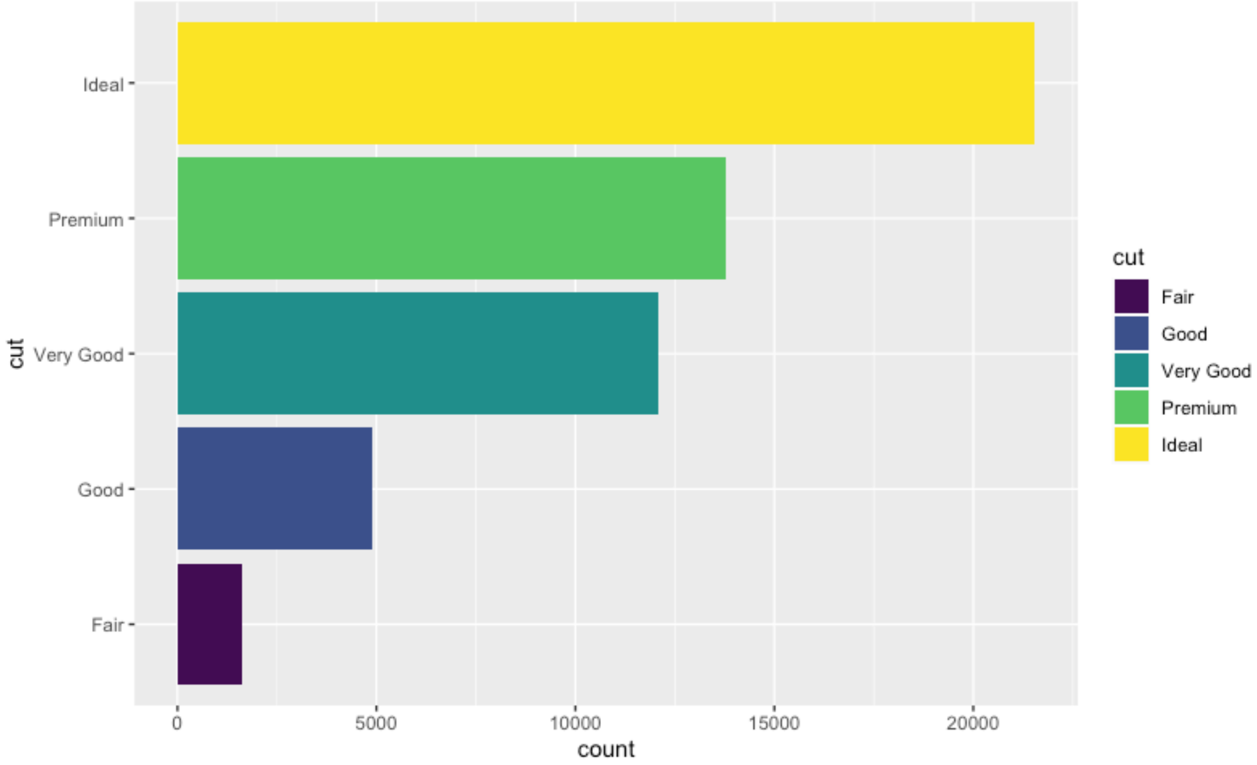
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```



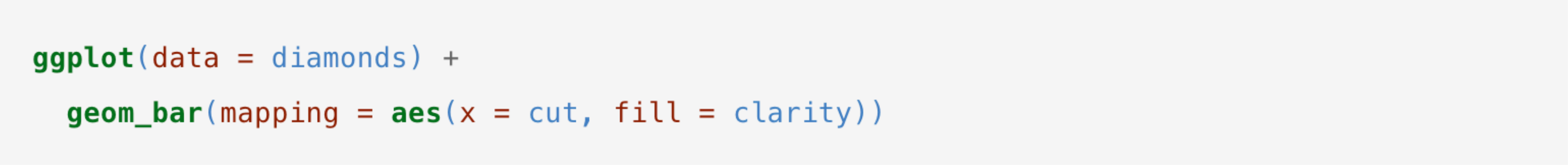


Bar plot: flipped to the side

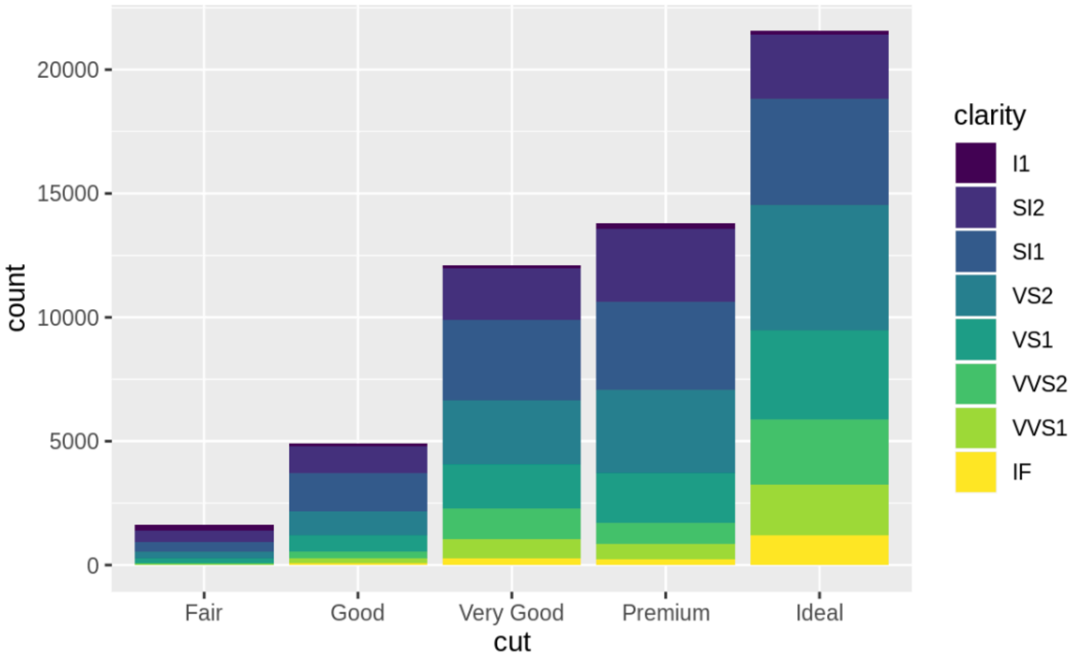
```
bar <- ggplot(data = diamonds) +
geom_bar( mapping = aes(x = cut, fill = cut))

bar + coord_flip()
```



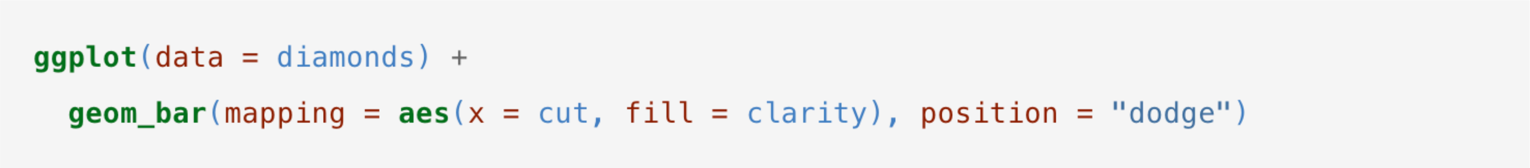
Bar plot: stacked bar plots



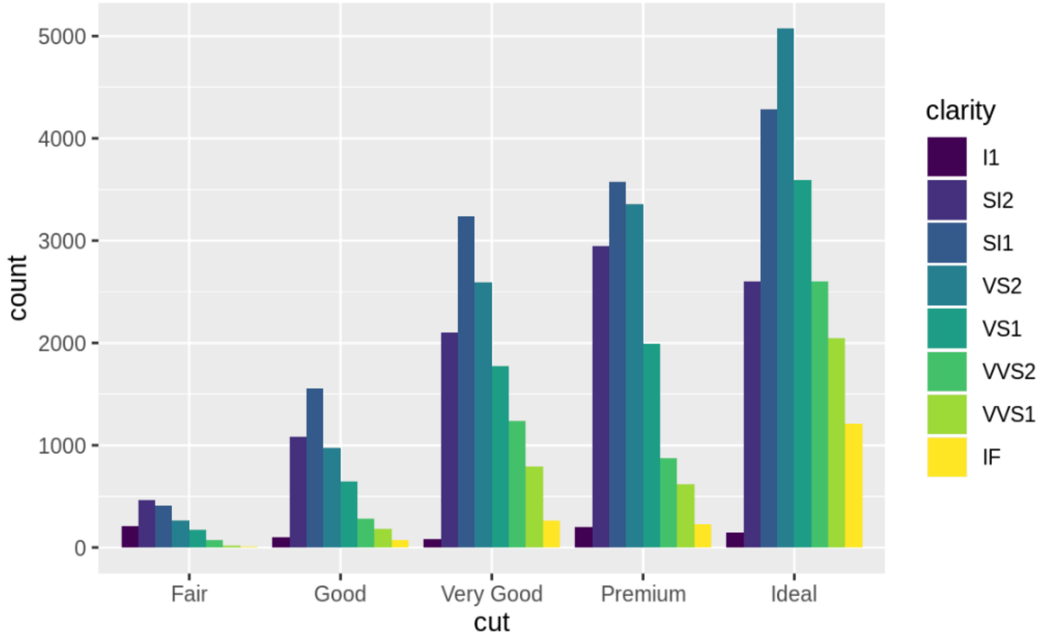
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```



Bar plots: grouped bar plots

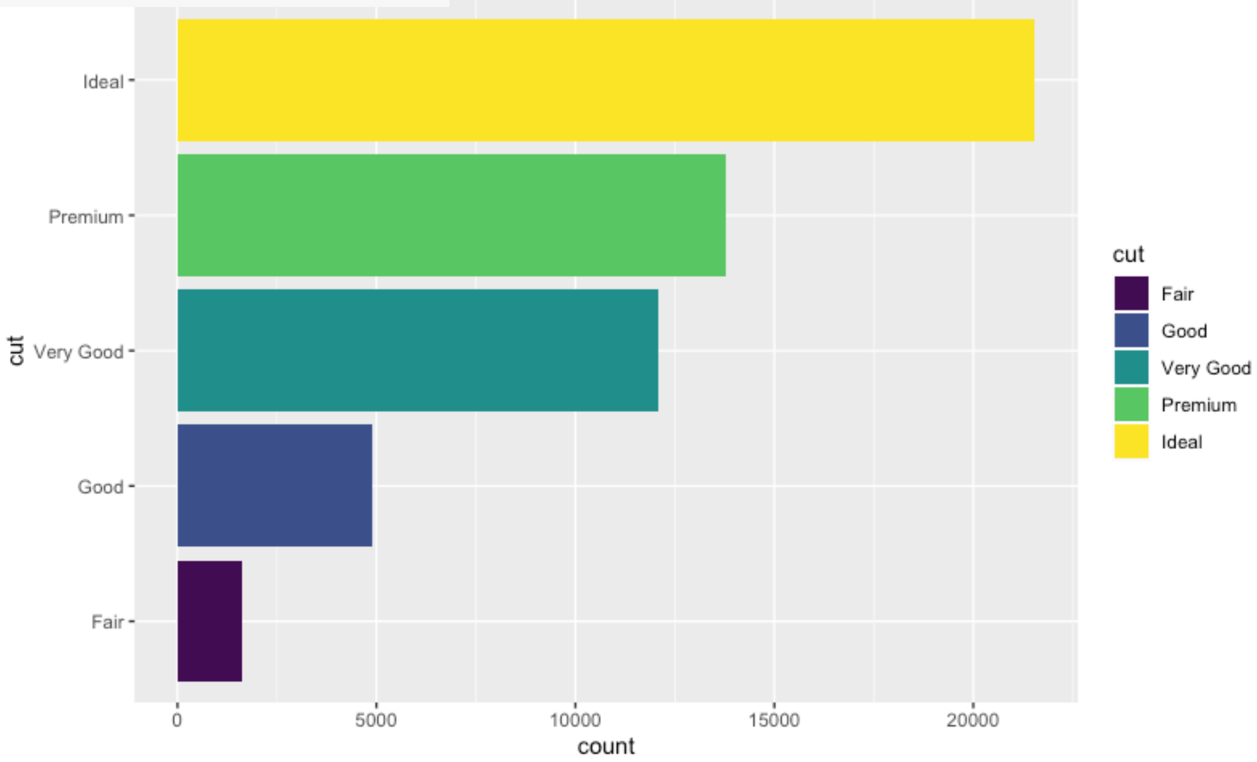


```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



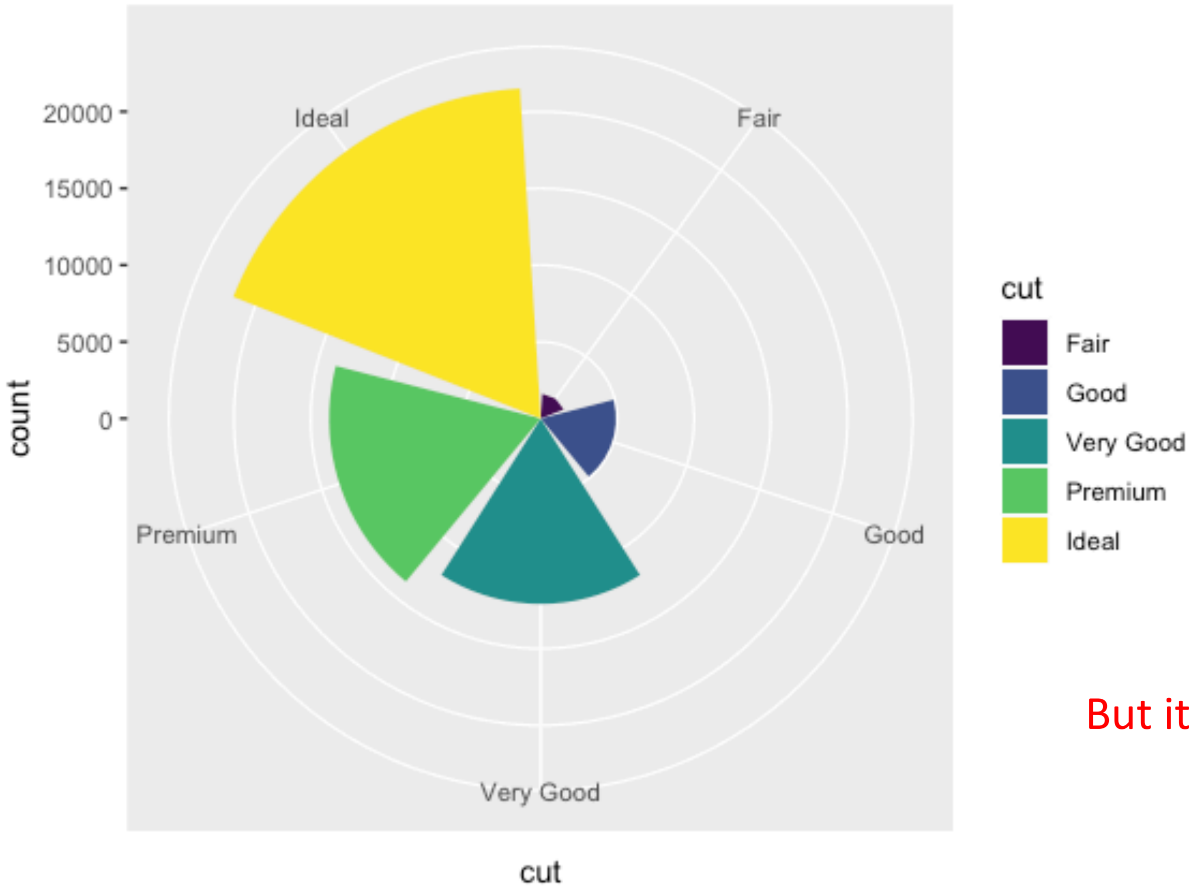
Pie Chart: Remember how we flipped the bar plot?

```
bar <- ggplot(data = diamonds) +
geom_bar( mapping = aes(x = cut, fill = cut))

bar + coord_flip()
```



Pie chart: similarly, we can create a pie chart

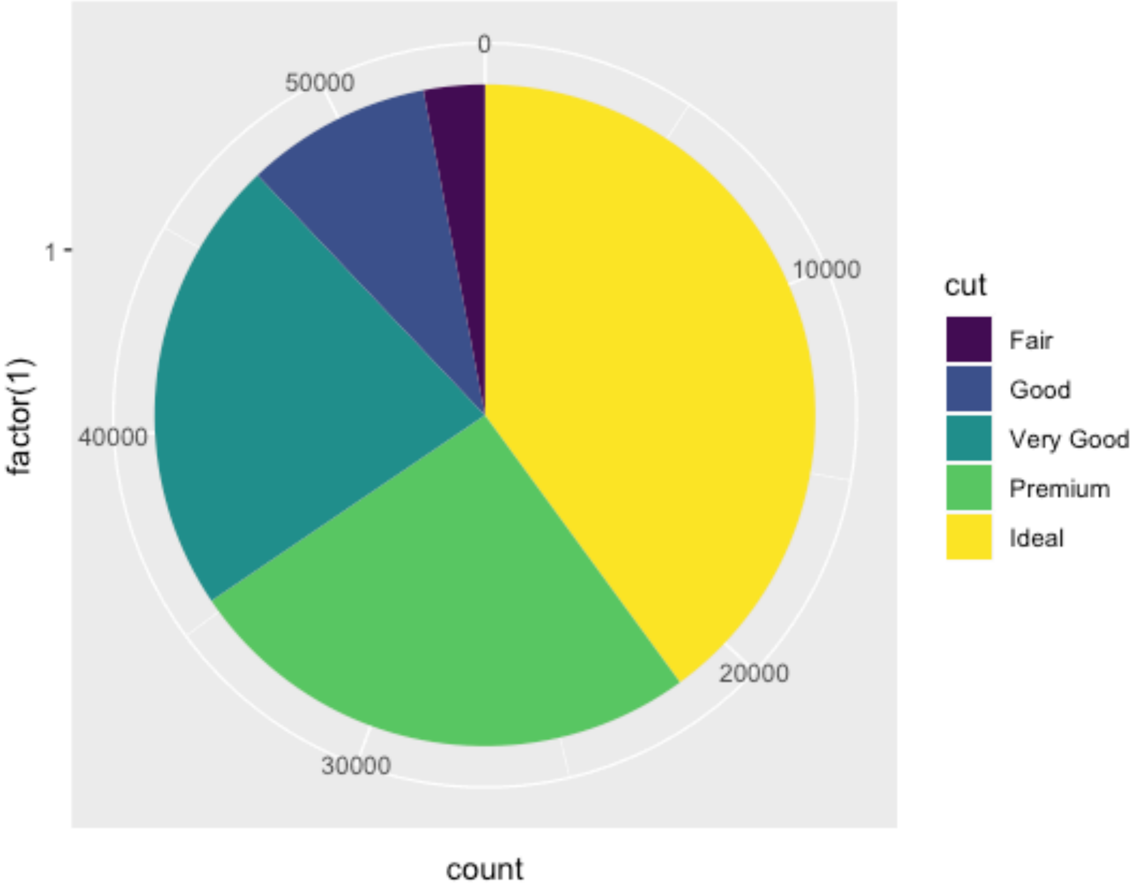
```
bar <- ggplot(data = diamonds) +
 geom_bar( mapping = aes(x = cut, fill = cut))

bar + coord_polar()
```



But it looks funky

Pie Chart

```
ggplot(diamonds, aes(x=factor(1), fill=cut))+
geom_bar(width = 1)+  coord_polar("y")
```



Feedback

Your feedback allows us to keep offering these workshops!

(and is required if you registered via GradProSkills)

https://www.datascientifique.ca/feedback.html

Thank you