# ggplot2

## WEBS

Tutorial: <http://r4ds.had.co.nz/data-visualisation.html>

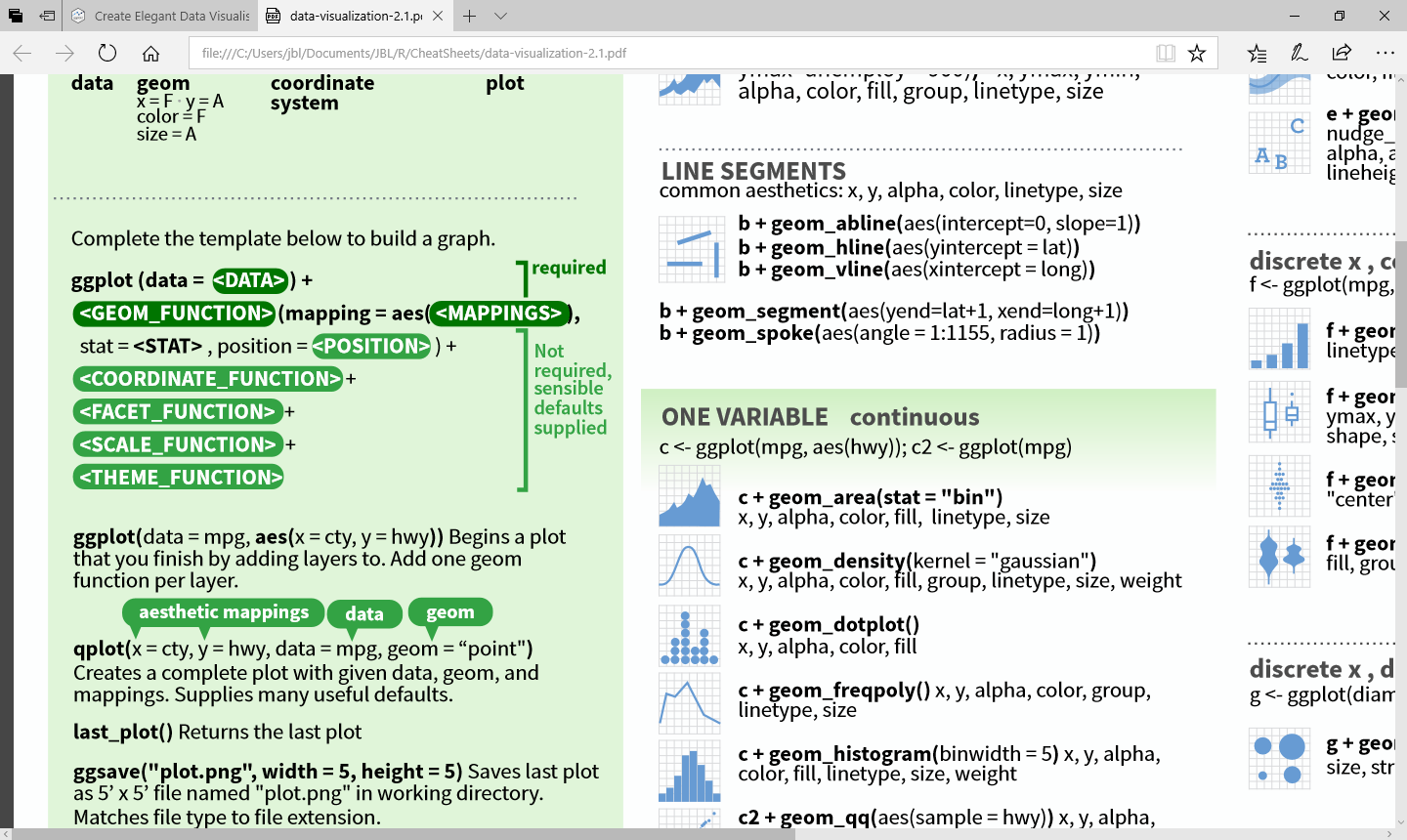
Documentation: <https://ggplot2.tidyverse.org/index.html>

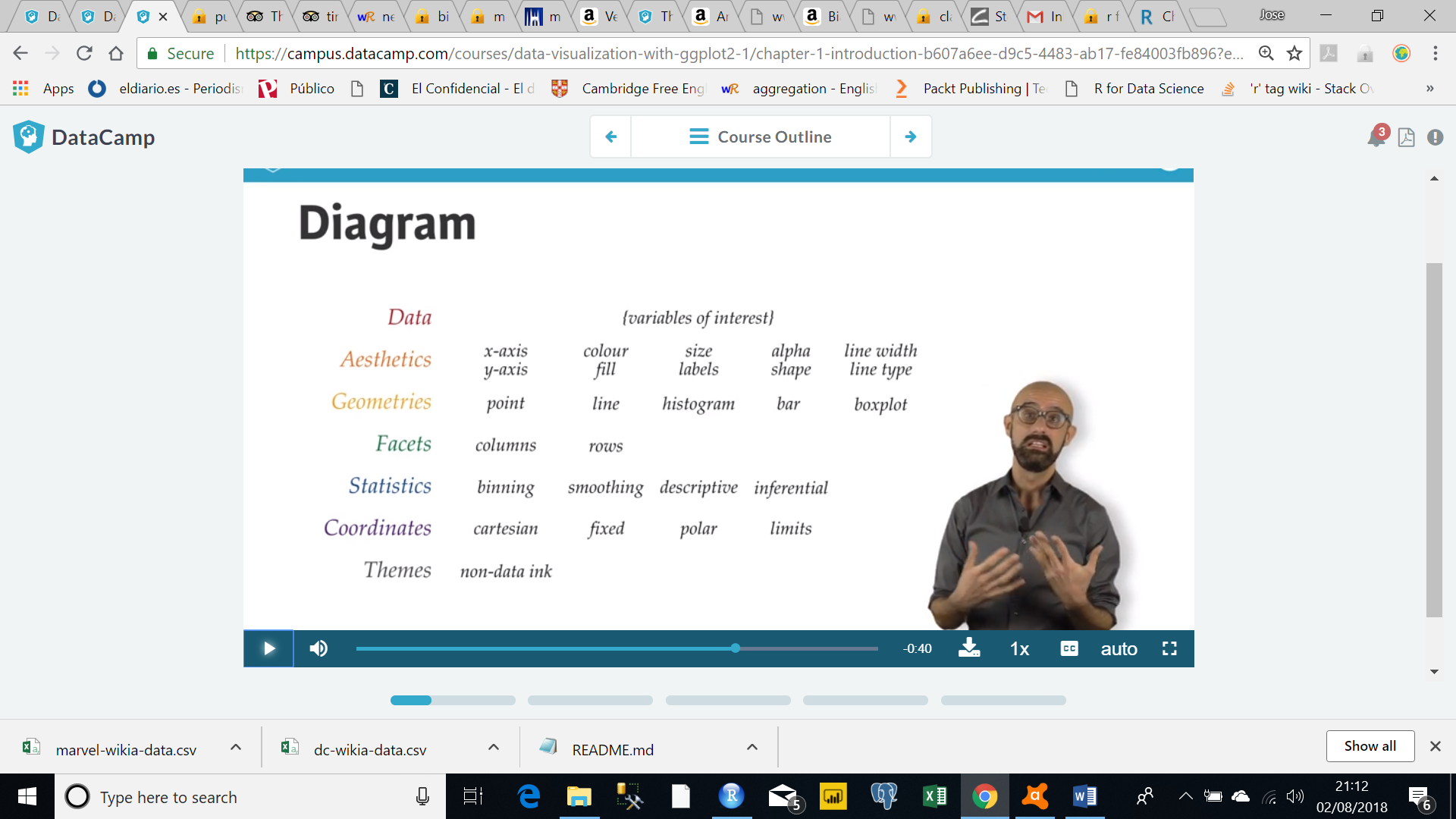
All plots: <https://www.r-graph-gallery.com/all-graphs/>

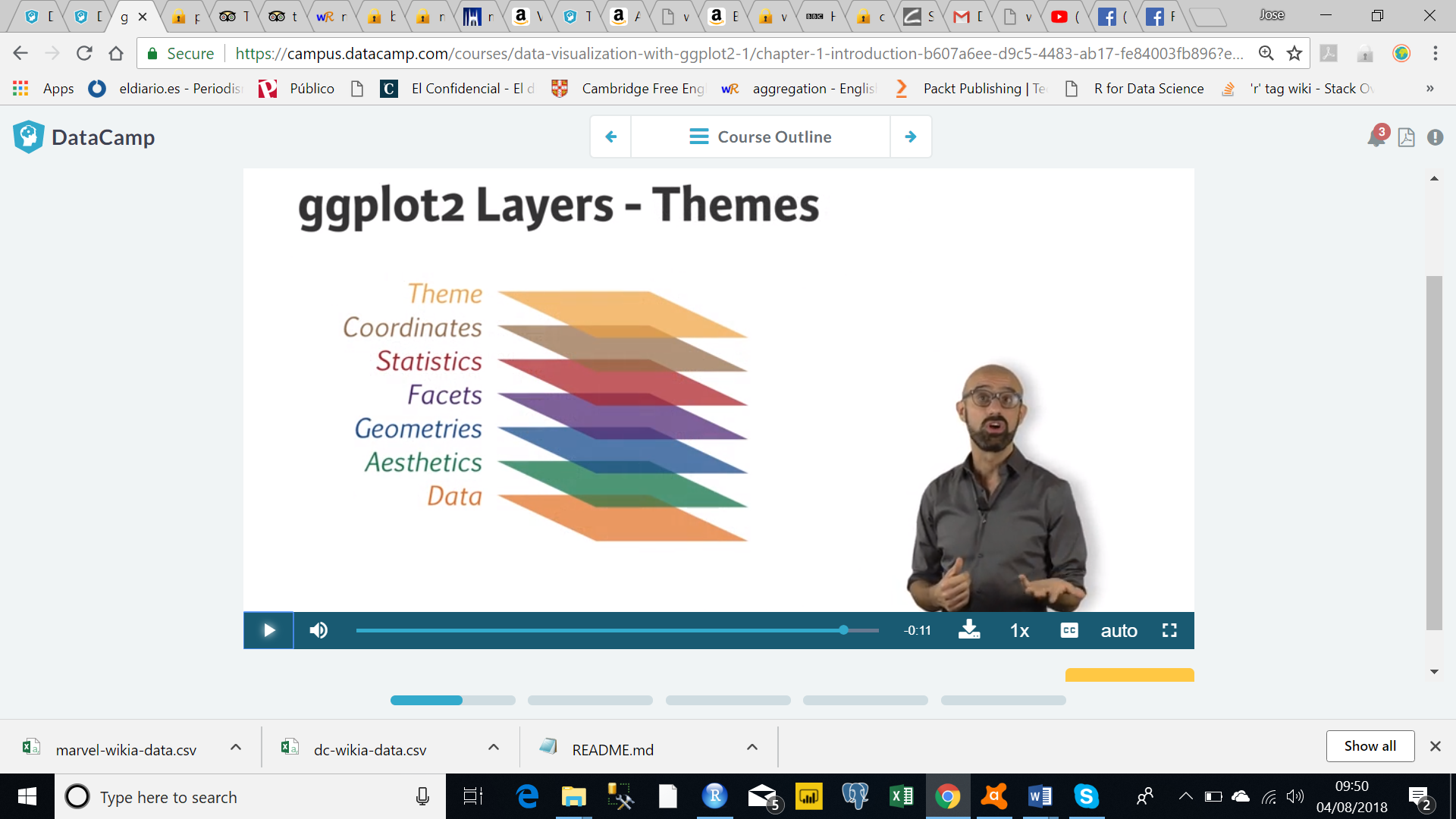
Data.Camp

* Data.camp.Data Visualization with ggplot2 (Part 1)
* DataCamp + Data Visualization with ggplot2
* DataCamp + Introduction to the Tidyverse

## LAYERS







# Plots Classification

([URL](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Lollipop%20Chart))

|  |  |
| --- | --- |
| ([URL](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Lollipop%20Chart)) | <https://www.r-graph-gallery.com/> |
| 1. Correlation | 2 |
| 2. Deviation |  |
| 3. Ranking | 3 |
| 4. Distribution | 1 |
| 5. Composition | 4. Part of a Whole |
| 6. Change | 5. Evolution / 7. Flow |
| 7. Groups |  |
| 8. Spatial | 6. Maps |
|  | 8. Other |

## Correlation

### Scatterplot

### Scatterplot With Encircling

### Jitter Plot

### Counts Chart

### Bubble Plot

### Animated Bubble Plot

### Marginal Histogram / Boxplot

### Correlogram

## Deviation

### Diverging Bars

### Diverging Lollipop Chart

### Diverging Dot Plot

### Area Chart

## Ranking

### [Ordered Bar Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Ordered%20Bar%20Chart)

* A bar chart uses bars to show comparisons between categories of data.
* Bar charts are ideal for visualizing the distribution or proportion of data items when there are more than three categories.
* These bars can be displayed horizontally or vertically.
* A bar graph will always have two axis. One axis will generally have numerical values, and the other will describe the types of categories being compared.
* Clustered Column Chart: A clustered column chart can be used if you need to compare multiple categories of data within individual sub-items as well as between sub-items.

### [Lollipop Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Lollipop%20Chart)

### [Dot Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Dot%20Plot)

### [Slope Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Slope%20Chart)

### [Dumbbell Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Dumbbell%20Plot)

## Distribution

### [Histogram](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Histogram)

### [Density Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Density%20Plot)

### [Box Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Box%20Plot)

### [Dot + Box Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Dot%20+%20Box%20Plot)

### [Tufte Boxplot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Tufte%20Boxplot)

### [Violin Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Violin%20Plot)

### [Population Pyramid](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Population%20Pyramid)

### Ridgeline / Joyplot

## Composition / Part of a whole

### [Waffle Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Waffle%20Chart)

### [Pie Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Pie%20Chart)

### [Treemap](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Treemap)

* Treemaps display hierarchical data as a set of nested rectangles.
* Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches
* A leaf node’s rectangle has an area proportional to a specified dimension of the data.
* It is an alternative way of visualizing the structure of a tree diagram.
* Treemap can be easily done using the Treemap library.

### [Bar Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Bar%20Chart)

## Change

### [Time Series Plots](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20From%20a%20Time%20Series%20Object)

### [From a Data Frame](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20From%20a%20Data%20Frame)

### [Format to Monthly X Axis](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20For%20a%20Monthly%20Time%20Series)

### [Format to Yearly X Axis](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20For%20a%20Yearly%20Time%20Series)

### [From Long Data Format](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20From%20Long%20Data%20Format)

### [From Wide Data Format](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Time%20Series%20Plot%20From%20Wide%20Data%20Format)

### [Stacked Area Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Stacked%20Area%20Chart)

### [Calendar Heat Map](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Calendar%20Heat%20Map)

### [Slope Chart](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Slope%20Chart%202)

### [Seasonal Plot](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Seasonal%20Plot)

## Groups

### [Dendrogram](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Hierarchical%20Dendrogram)

### [Clusters](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Clusters)

## Spatial

### [Open Street Map](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Open%20Street%20Map)

### [Google Road Map](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Google%20Road%20Map)

### [Google Hybrid Map](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Google%20Hybrid%20Map)

# TUTORIAL

**A graphing template**

ggplot(data = <DATA>) +

<GEOM\_FUNCTION>(mapping = aes(<MAPPINGS>))

mpg data frame can be found in ggplot2 (ggplot2::mpg)

11 variables

1. manufacturer
2. model: model name
3. displ: engine displacement, in litres
4. year: year of manufacture
5. cyl: number of cylinders
6. trans: type of transmission
7. drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd
8. cty: city miles per gallon
9. hwy: highway miles per gallon
10. fl: fuel type
11. class: "type" of car

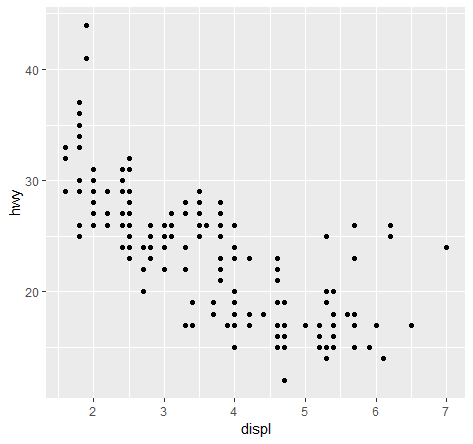
# cty = city miles per gallon

# displ = a car’s engine size, in litres.

# hwy = a car’s fuel efficiency on the highway, in miles per gallon (mpg).

ggplot(mpg1) +

geom\_point(mapping = aes(displ, hwy))



* The plot shows a negative relationship between engine size (displ) and fuel efficiency (hwy).
* you begin a plot with the function ggplot()

ggplot()

* ggplot() creates a **coordinate system** that you can add layers to.
* The first argument of ggplot() is the dataset to use in the graph.
* You complete your graph by adding one or more layers to ggplot()
* ggplot2 comes with many geom functions that each add a different type of layer to a plot.
* Each geom function in ggplot2 takes a mapping argument. This defines how variables in your dataset are mapped to visual properties.

geom\_point()

* The function geom\_point() adds a layer of points to your plot, which creates a scatterplot.

aes()

* The **mapping argument** is always paired with aes(), and the x and y arguments of aes() specify which variables to map to the x and y axes.

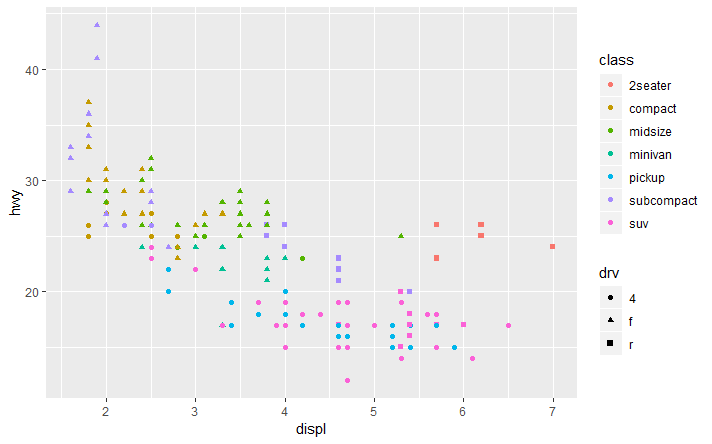
Adding additional variables

1. Aesthetics properties
2. Facets (only categorical)
   1. facet\_wrap() 🡪 1 categorical variable
   2. facet\_grid() 🡪 2 categorical variables
3. blah

### Asthetics

ggplot(cars) +

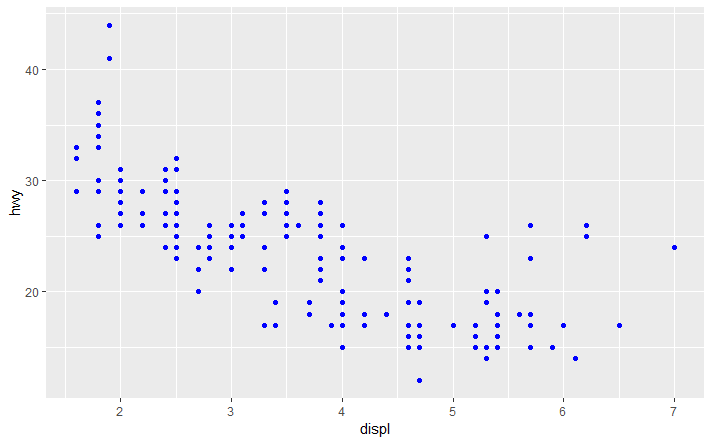
geom\_point(mapping = aes(displ, hwy, color = class, shape = drv))



* You can add a third variable (and a 4th or 5th) to a two-dimensional scatterplot by mapping it to an aesthetic.
* An aesthetic is a visual property of the objects in your plot.
* Aesthetics include **aesthetic properties** such as:
  + Size
    - categorical 🡪 size proportional to the value
    - continuous 🡪 a set of levels
  + Shape
    - only categorical
  + colour
    - categorical 🡪 different colours
    - continuous 🡪 same colour, different gradient
  + alpha (transparency) of your points.
    - Categorical 🡪 Using alpha for a discrete variable is not advised
  + stroke
    - Use the stroke aesthetic to modify the width of the border
* Scaling 🡪 ggplot2 will automatically assign a unique level of the aesthetic (i.e., a unique color, shape and/or size) to each unique value of the variable.
* Maths::scaling 🡪 Scaling is maintaining some relationship, while changing the numbers appropriately.

**ggplot**(data = mpg) +

**geom\_point**(mapping = **aes**(x = displ, y = hwy), color = "blue")



* To set an aesthetic manually, set the aesthetic by name as an argument of your geom function; i.e. it goes outside of aes()

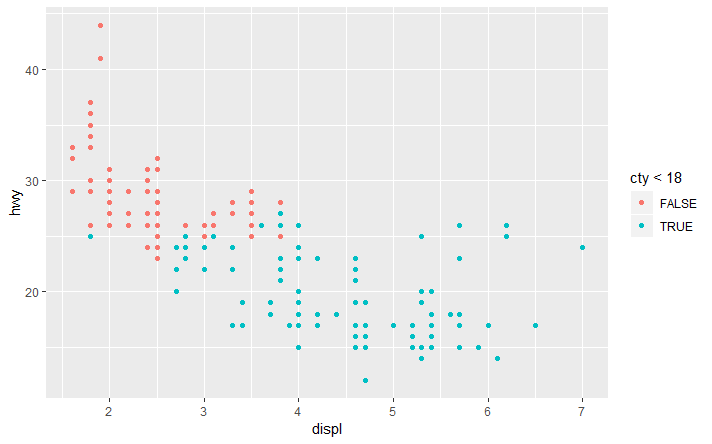
# cty = city miles per gallon

# hwy = a car’s fuel efficiency on the highway, in miles per gallon (mpg).

# displ = a car’s engine size, in litres.

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, colour = cty < 18))



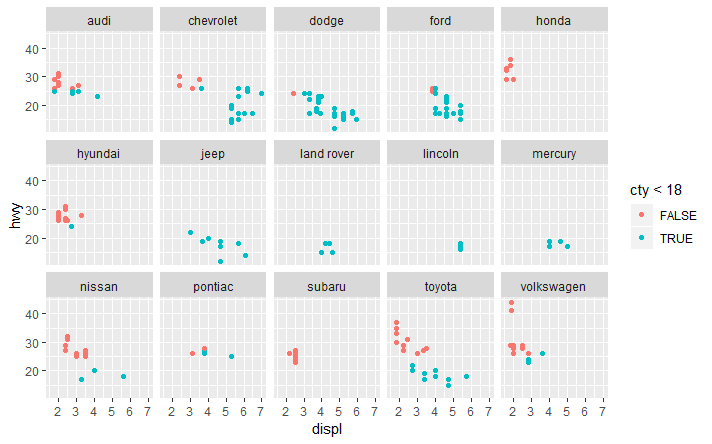
**Facets**

* One way to add additional variables is with aesthetics.
* Another way, particularly useful for categorical variables, is to split your plot into facets, subplots that each display one subset of the data.
* To facet your plot by a single variable, use facet\_wrap().
* The first argument of facet\_wrap() should be a formula, which you create with ~ followed by a variable name (here “formula” is the name of a data structure in R, not a synonym for “equation”).
* The variable that you pass to facet\_wrap() should be discrete.

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, colour = cty < 18)) +

facet\_wrap(~ manufacturer, nrow = 3)

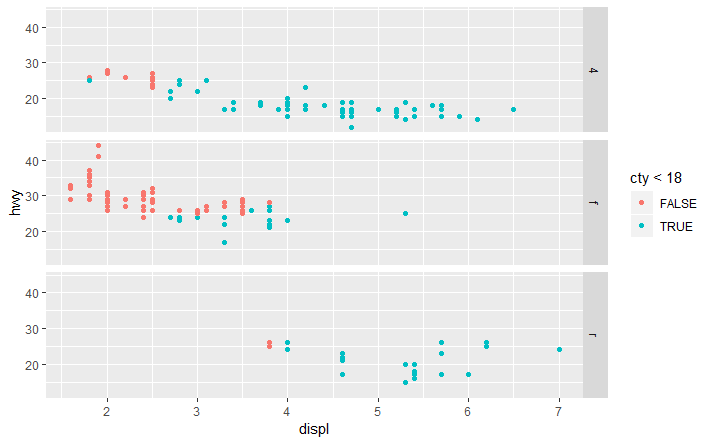


* To facet your plot on the combination of two variables, add facet\_grid() to your plot call.
* The first argument of facet\_grid() is also a formula. This time the formula should contain two variable names separated by a ~.

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, colour = cty < 18)) +

facet\_grid(drv ~ .)



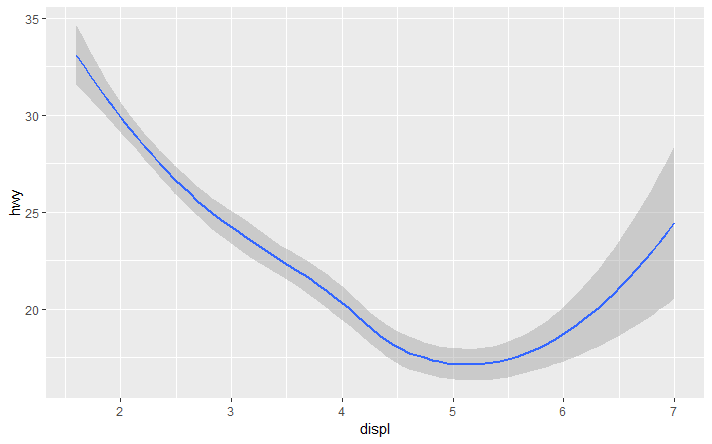
* If you prefer to not facet in the rows or columns dimension, use a . instead of a variable name, e.g. + facet\_grid(. ~ cyl).
* When using facet\_grid() you should usually put the variable with more unique levels in the columns. Why?

**Geometric objects**

* A geom is the geometrical object that a plot uses to represent data.
* People often describe plots by the type of geom that the plot uses.
* For example,
  + bar charts use bar geoms,
  + line charts use line geoms,
  + boxplots use boxplot geoms,
  + Scatterplots break the trend; they use the point geom.
* You can use different geoms to plot the same data.

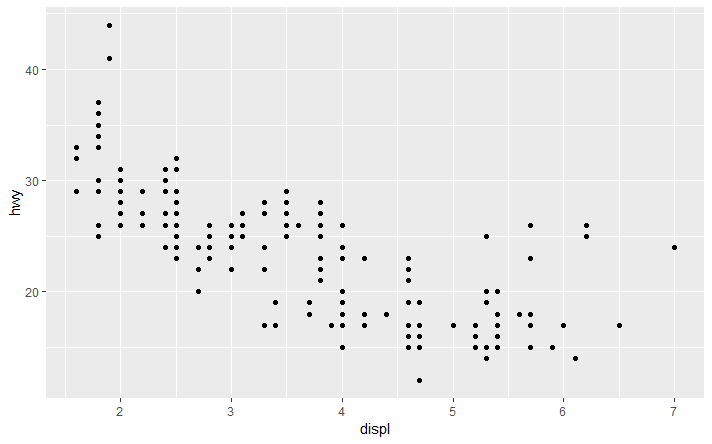
**ggplot**(data = mpg) +

**geom\_smooth**(mapping = **aes**(x = displ, y = hwy))



**ggplot**(data = mpg) +

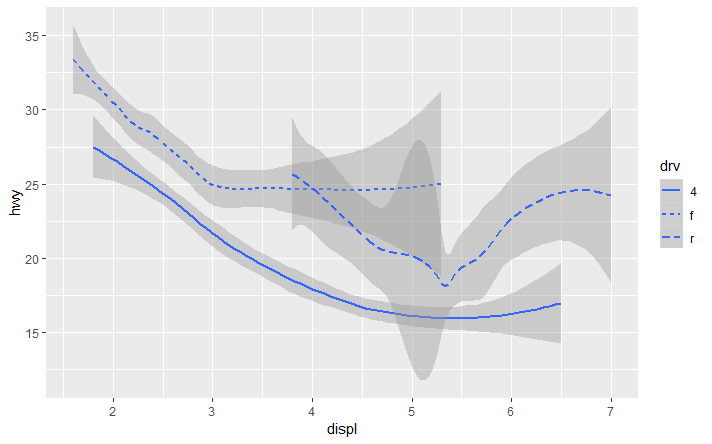
**geom\_point**(mapping = **aes**(x = displ, y = hwy))



* Every geom function in ggplot2 takes a mapping argument.
* However, not every aesthetic works with every geom.
* You could set the shape of a point, but you couldn’t set the “shape” of a line.
* On the other hand, you could set the linetype of a line.
* geom\_smooth() will draw a different line, with a different linetype, for each unique value of the (categorical) variable that you map to linetype.

**ggplot**(data = mpg) +

**geom\_smooth**(mapping = **aes**(x = displ, y = hwy, linetype = drv))

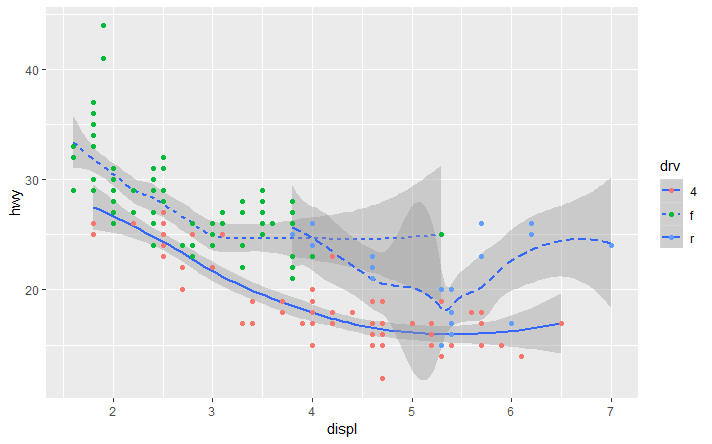


* Notice that this plot contains two geoms in the same graph!

**ggplot**(data = mpg) +

**geom\_smooth**(mapping = **aes**(x = displ, y = hwy, linetype = drv)) +

**geom\_point**(mapping = **aes**(x = displ, y = hwy, colour = drv))

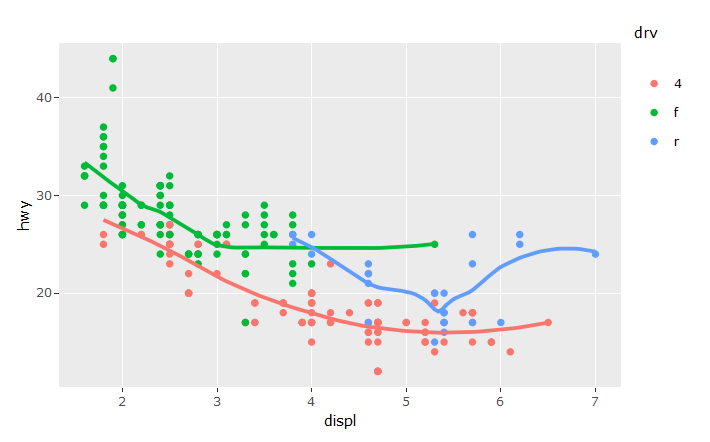


* This, however, introduces some duplication in our code. Solution:

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

**geom\_point**() +

**geom\_smooth**(se = FALSE)

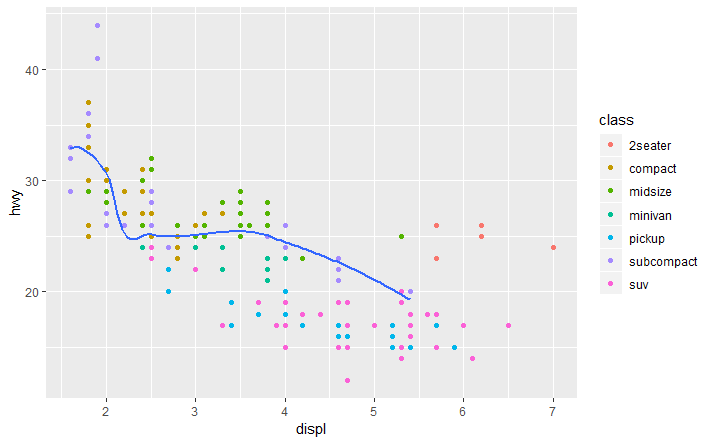


* You can use the same idea to specify different data for each layer.

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

**geom\_point**(mapping = **aes**(color = class)) +

**geom\_smooth**(data = **filter**(mpg, class == "subcompact"), se = FALSE)

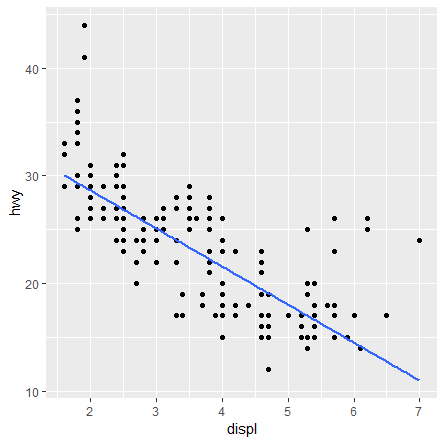


**Linear Regression**

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

geom\_point() +

geom\_smooth(method = "lm", se = FALSE)



* ggplot2 provides over 30 geoms, and extension packages provide even more (see <https://www.ggplot2-exts.org> for a sampling).
* The best way to get a comprehensive overview is the ggplot2 cheatsheet, which you can find at <http://rstudio.com/cheatsheets>.
* To learn more about any single geom, use help: ?geom\_smooth.

**6. Exercises**

ggplot(data = mpg, mapping = aes(x = displ, y = hwy, stroke = 3)) +

geom\_point() +

geom\_smooth()

ggplot(data = mpg, mapping = aes(x = displ, y = hwy, stroke = 3)) +

geom\_point() +

geom\_smooth(mapping = aes(line = drv))

ggplot(data = mpg, mapping = aes(x = displ, y = hwy, stroke = 3, color = drv)) +

geom\_point() +

geom\_smooth(mapping = aes(line = drv),se = FALSE)

Diamonds Dataset (?ggplot2::diamonds)

A data frame with 53940 rows and 10 variables:

1. price - price in US dollars (\$326–\$18,823)
2. carat - weight of the diamond (0.2–5.01)
3. cut - quality of the cut (Fair, Good, Very Good, Premium, Ideal)
4. color - diamond colour, from J (worst) to D (best)
5. clarity - a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
6. x - length in mm (0–10.74
7. y - width in mm (0–58.9)
8. z - depth in mm (0–31.8)
9. depth - total depth percentage = z / mean(x, y) = 2 \* z / (x + y) (43–79)
10. table - width of top of diamond relative to widest point (43–95)

**Statistical transformations**

* Next, let’s take a look at a bar chart.
* Bar charts seem simple, but they are interesting because they reveal something subtle about plots.
* Consider a basic bar chart, as drawn with geom\_bar(). The following chart displays the total number of diamonds in the diamonds dataset, grouped by cut.

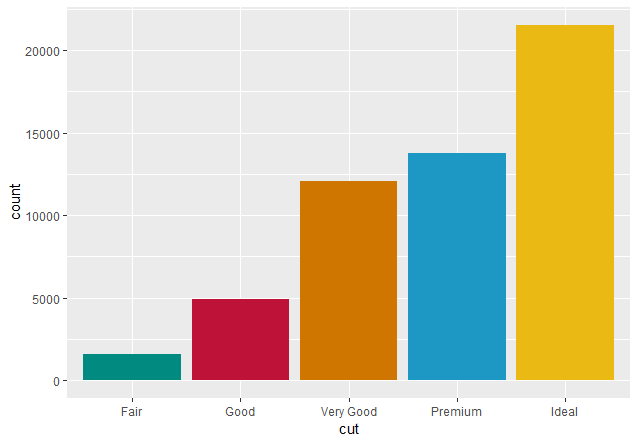
ggplot(data = diamantes) +

geom\_bar(mapping = aes(x = cut), fill = paleta5)

and

ggplot(data = diamantes) +

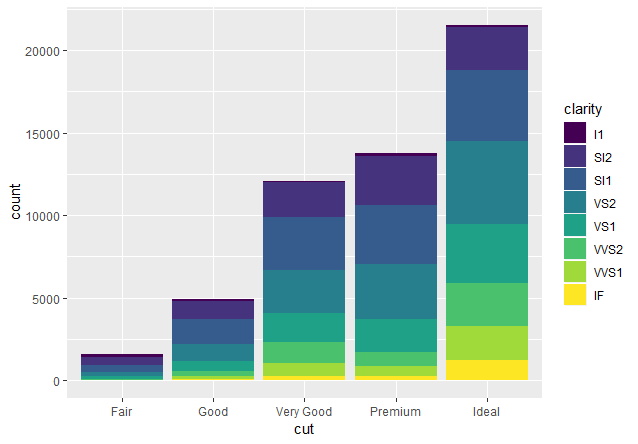
stat\_count(mapping = aes(x = cut), fill = paleta5)



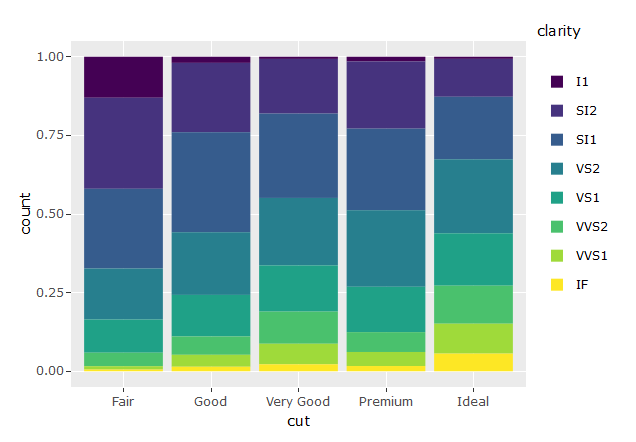
* geom\_bar() uses stat\_count()

ggplot(data = diamantes) +

geom\_bar(mapping = aes(x = cut, fill = clarity))



* If you don’t want a stacked bar chart, you can use one of three other options:
  + "identity",
  + "dodge"
  + "fill"
* position = "identity" will place each object exactly where it falls in the context of the graph. This is not very useful for bars.
* position = "fill" works like stacking, but makes each set of stacked bars the same height.
  + This makes it easier to compare proportions across groups.



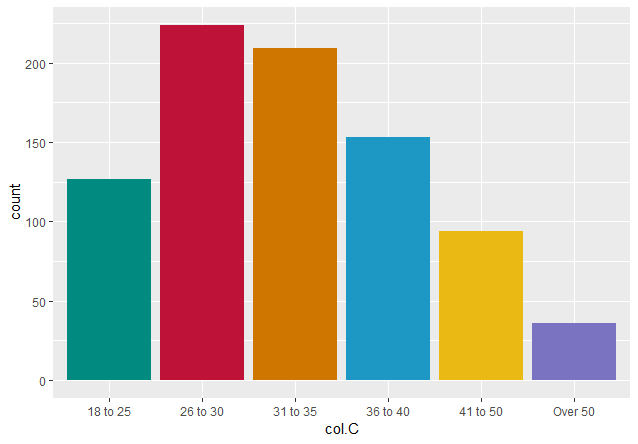
* position = "dodge" places overlapping objects directly beside one another. This makes it easier to compare individual values.

**Tips**

* + theme(axis.text.x = element\_text(angle = 36)) 🡪 gets the levels names to an angle

ggplot(Dubai, aes(x = col.C)) +

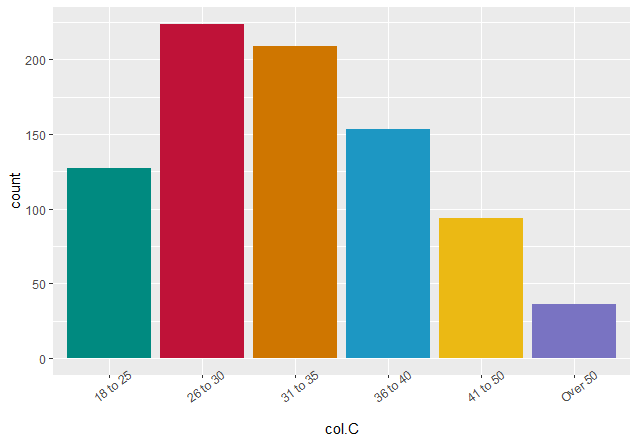
geom\_bar(position = "dodge",fill= Paleta6)



ggplot(Dubai, aes(x = col.C)) +

geom\_bar(position = "dodge",fill= Paleta6) +

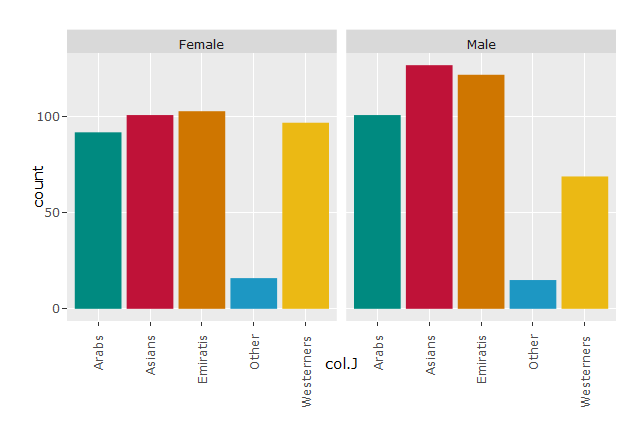
theme(axis.text.x = element\_text(angle = 36))



ggplotly(ggplot(Dubai, aes(x = col.J)) +

geom\_bar(position = "dodge", fill = Paleta5.2) + facet\_wrap(~col.E)

+ theme(axis.text.x = element\_text(angle = 90)))



**VISUALISING**

* The choice of graphs should be informed by the type of data
* **Bar plots** are good for comparing statistics within different **categories**
* **Scatter plots** are useful for comparing two variables which each point representing one observation
* **Line plots** are useful for showing change over time
  + Upward or downward trend over time
* **Histograms** describe the distribution of one-dimensional **numeric** variable
* **Box plots** compare the distributions of **numerical** variables among several categories

# Examples

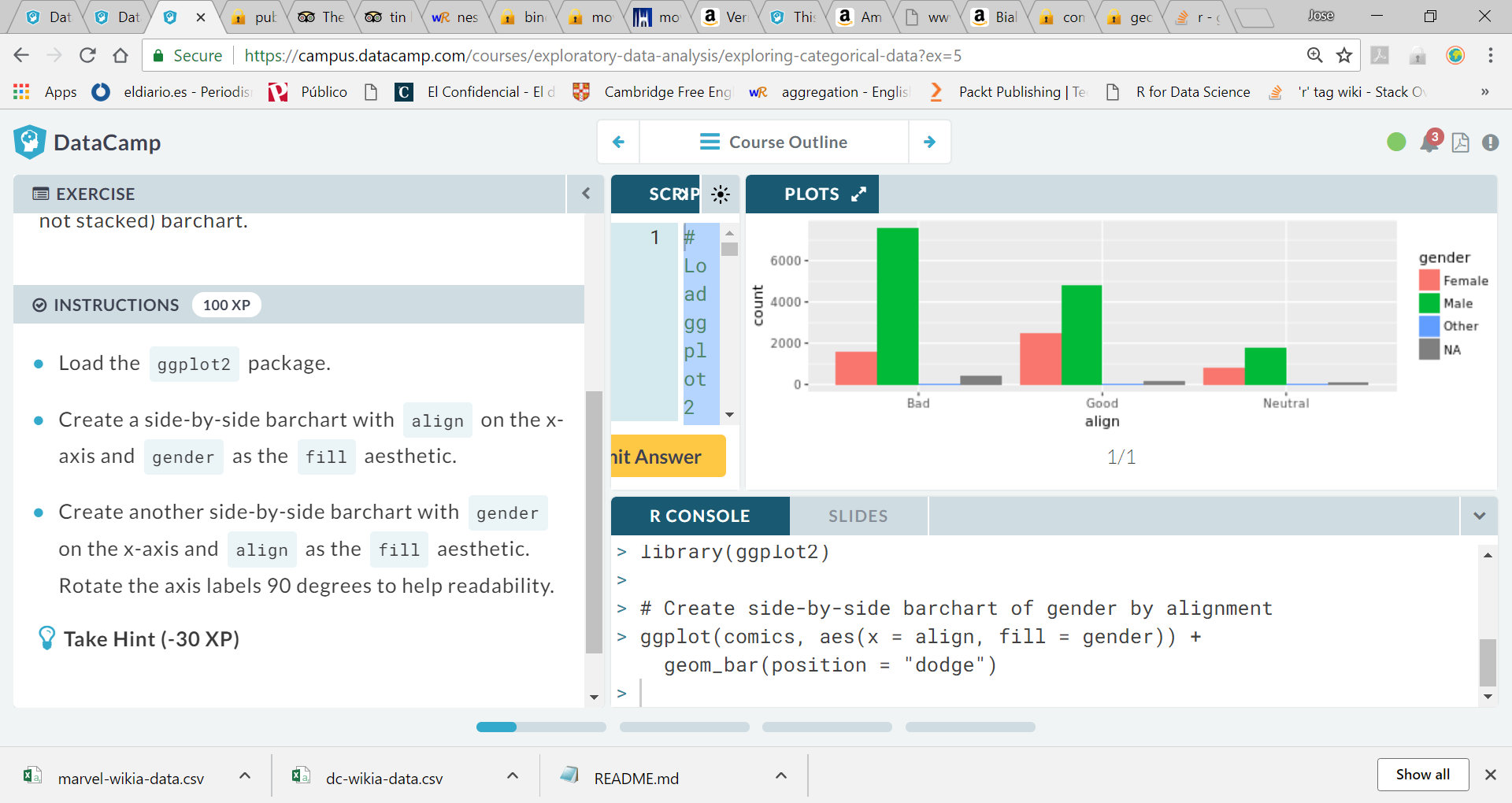
Passing the argument position = "dodge" to geom\_bar() says that you want a side-by-side (i.e. not stacked) barchart.

* Create a side-by-side barchart with align on the x-axis and gender as the fill aesthetic.

# Create side-by-side barchart of gender by alignment

ggplot(comics, aes(x = align, fill = gender)) +

geom\_bar(position = "dodge")



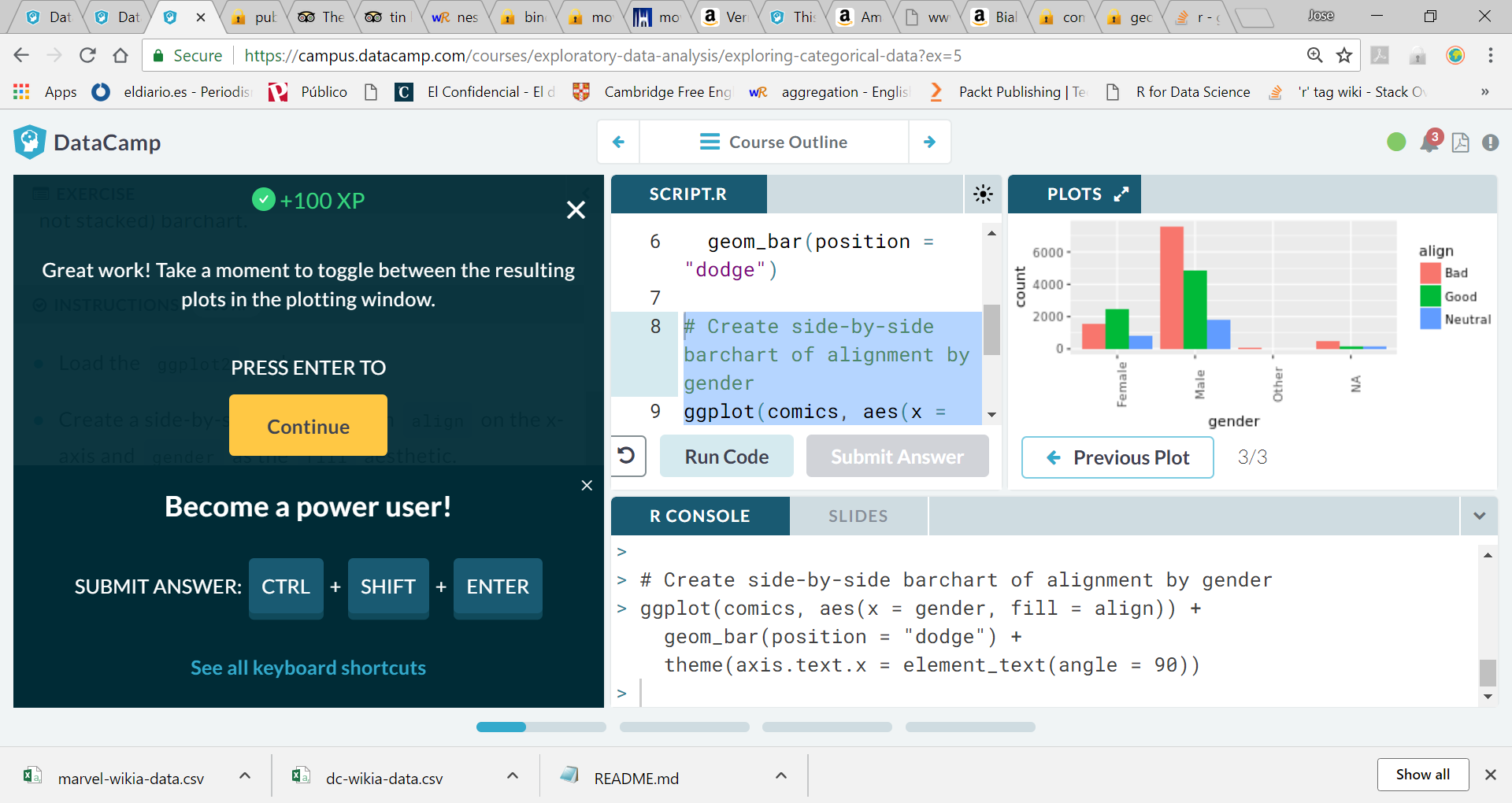
* Create another side-by-side barchart with genderon the x-axis and align as the fill aesthetic. Rotate the axis labels 90 degrees to help readability.

# Create side-by-side barchart of alignment by gender

ggplot(comics, aes(x = gender, fill = align)) +

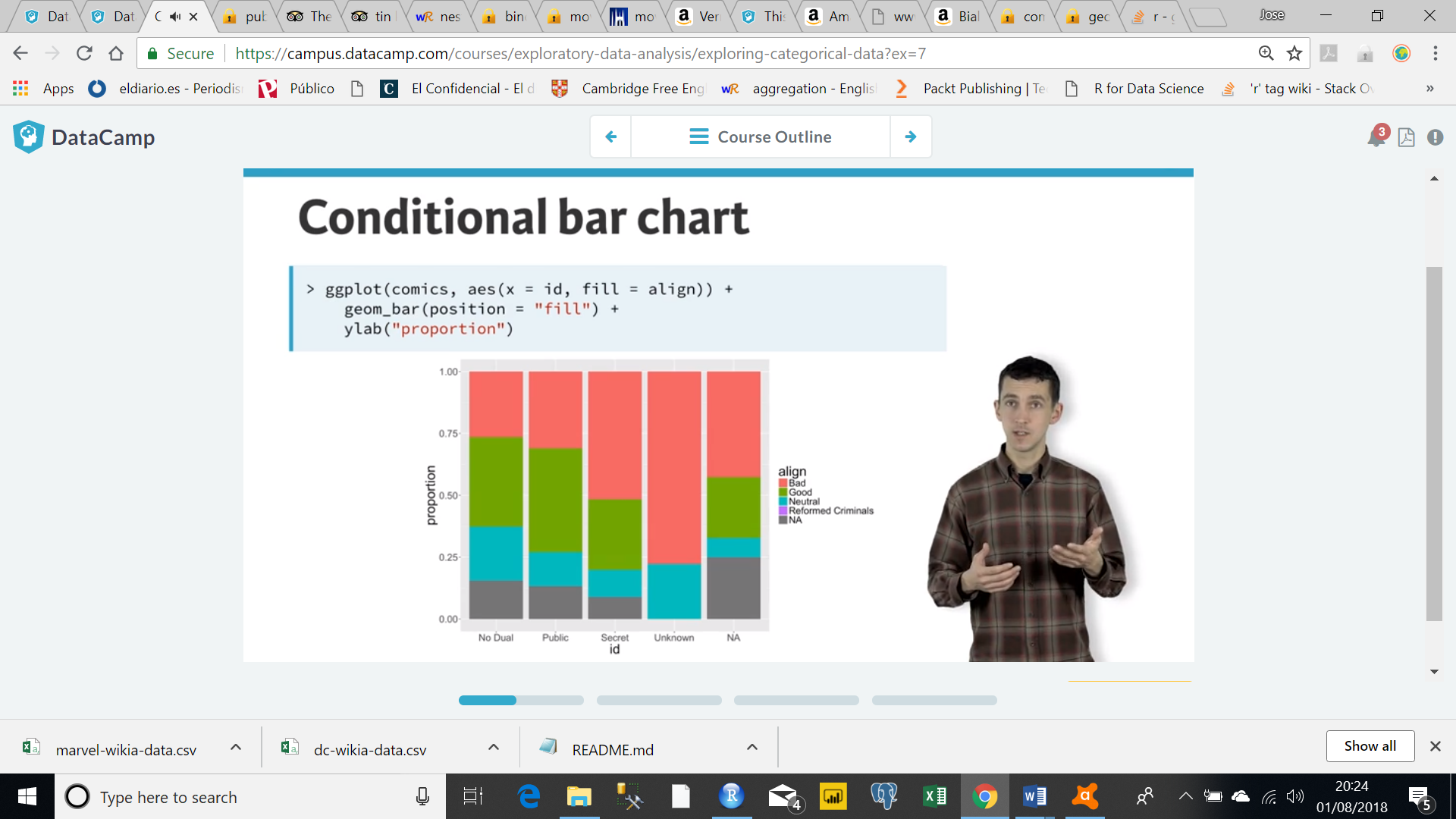
geom\_bar(position = "dodge") +

theme(axis.text.x = element\_text(angle = 90))



ggplot(comics, aes(x = id, fill = align)) + geom\_bar(position = “fill”) + ylab(“proportion”)

Condition on id …

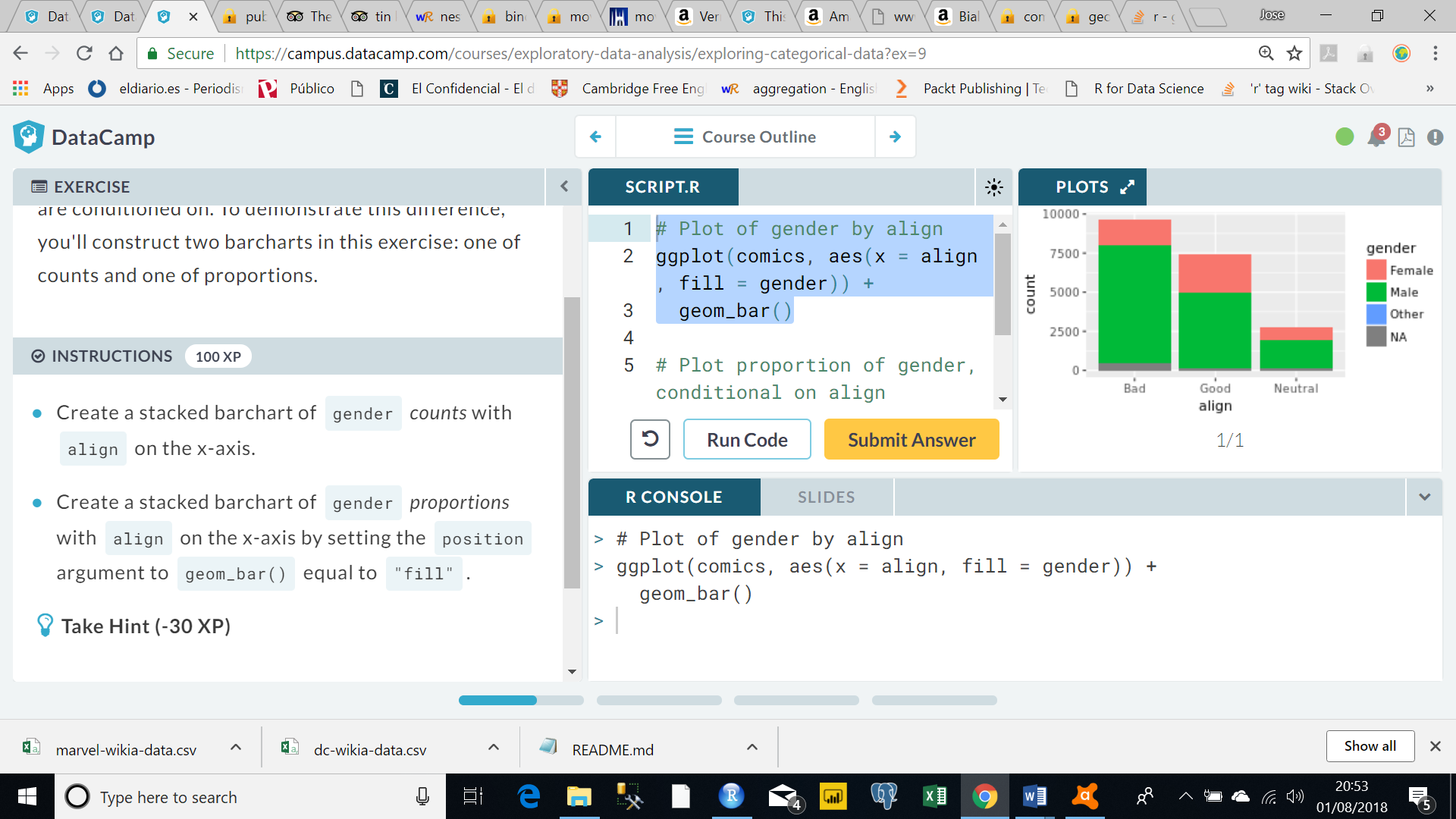


If we condition on alignment …

# Plot of gender by align

ggplot(comics, aes(x = align, fill = gender)) +

geom\_bar()



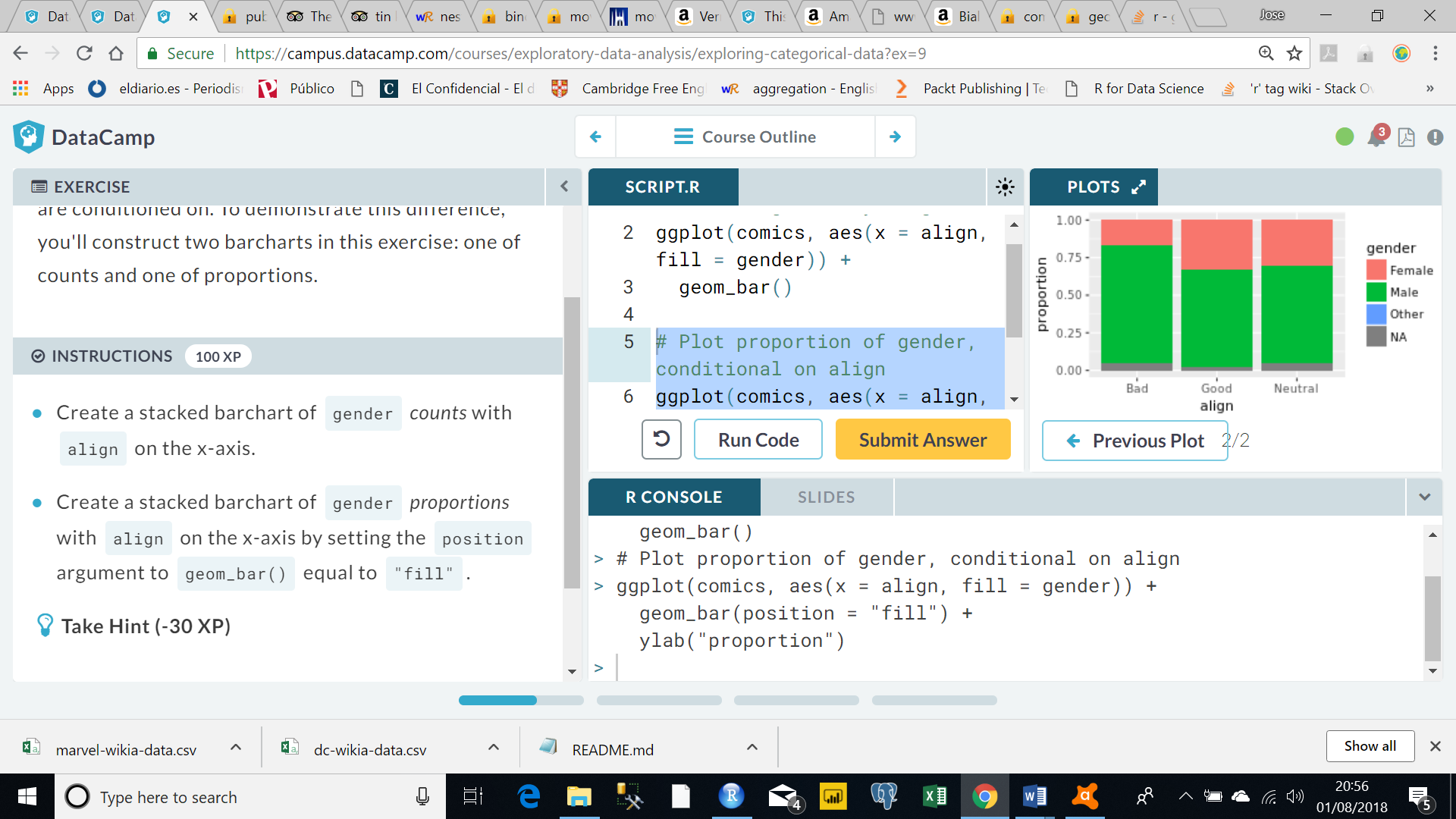
* Create a stacked barchart of gender *proportions* with align on the x-axis by setting the position argument to geom\_bar() equal to "fill".

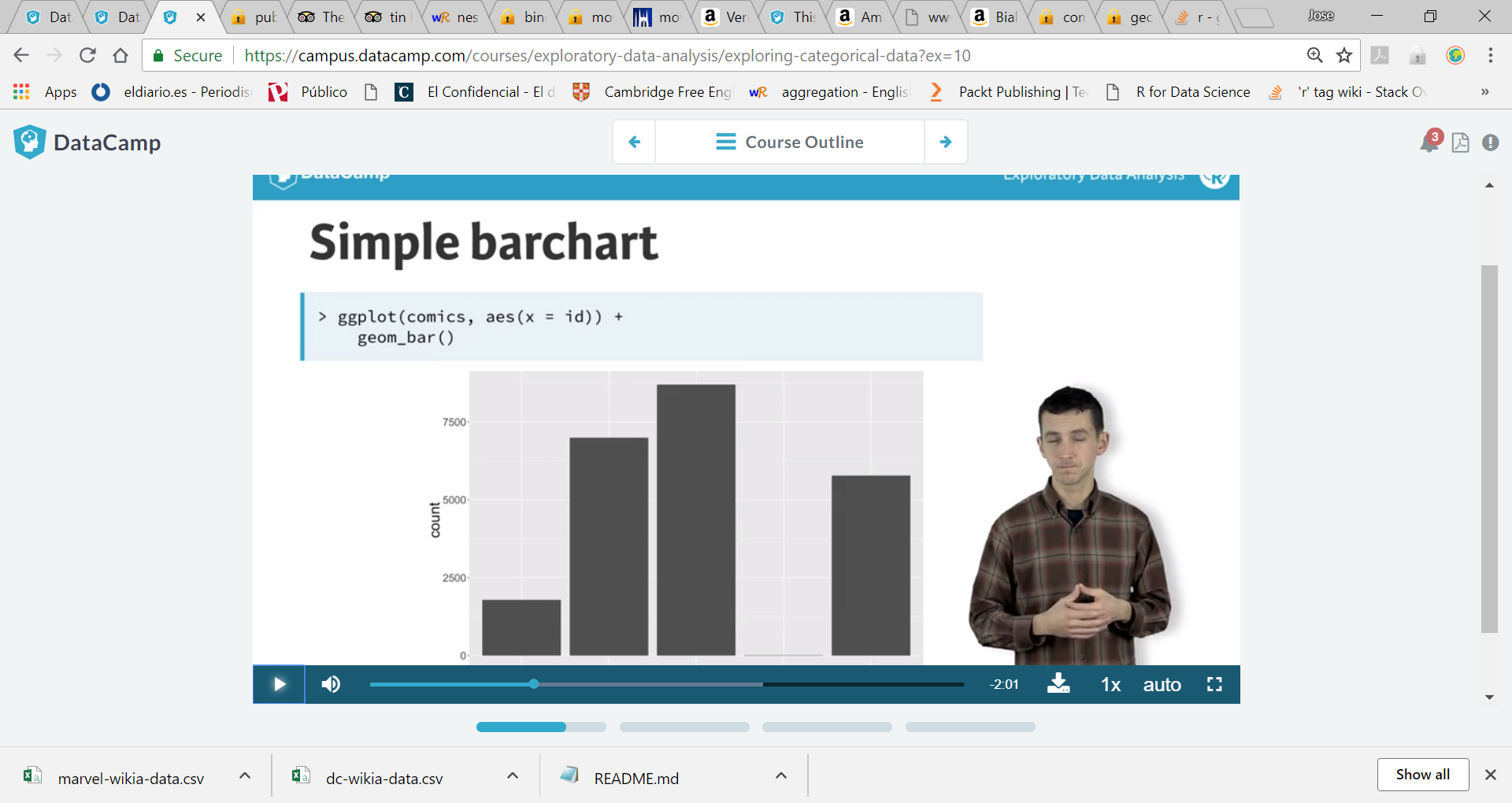
# Plot proportion of gender, conditional on align

ggplot(comics, aes(x = align, fill = gender)) +

geom\_bar(position = "fill") +

ylab("proportion")





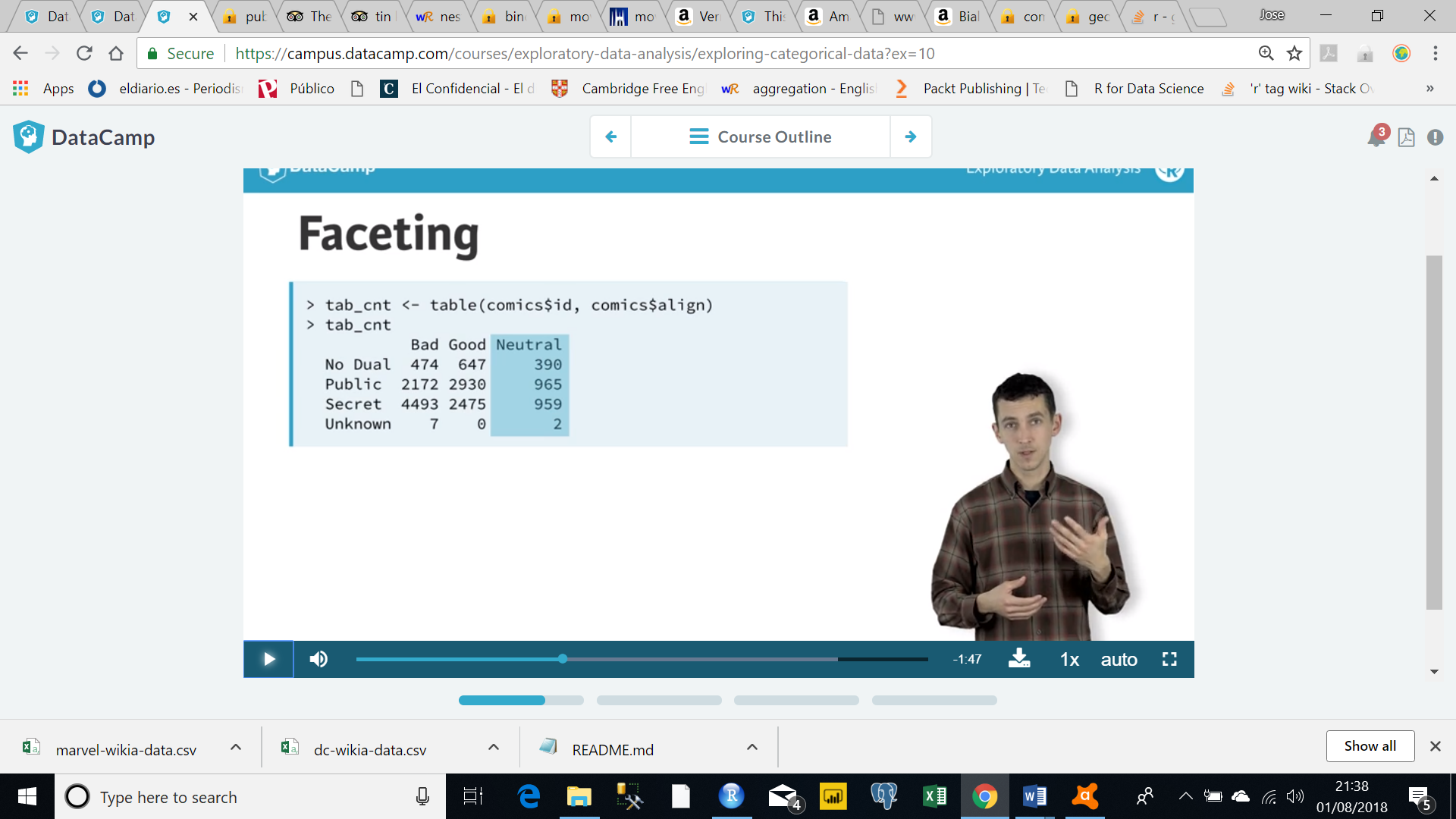
FACETING

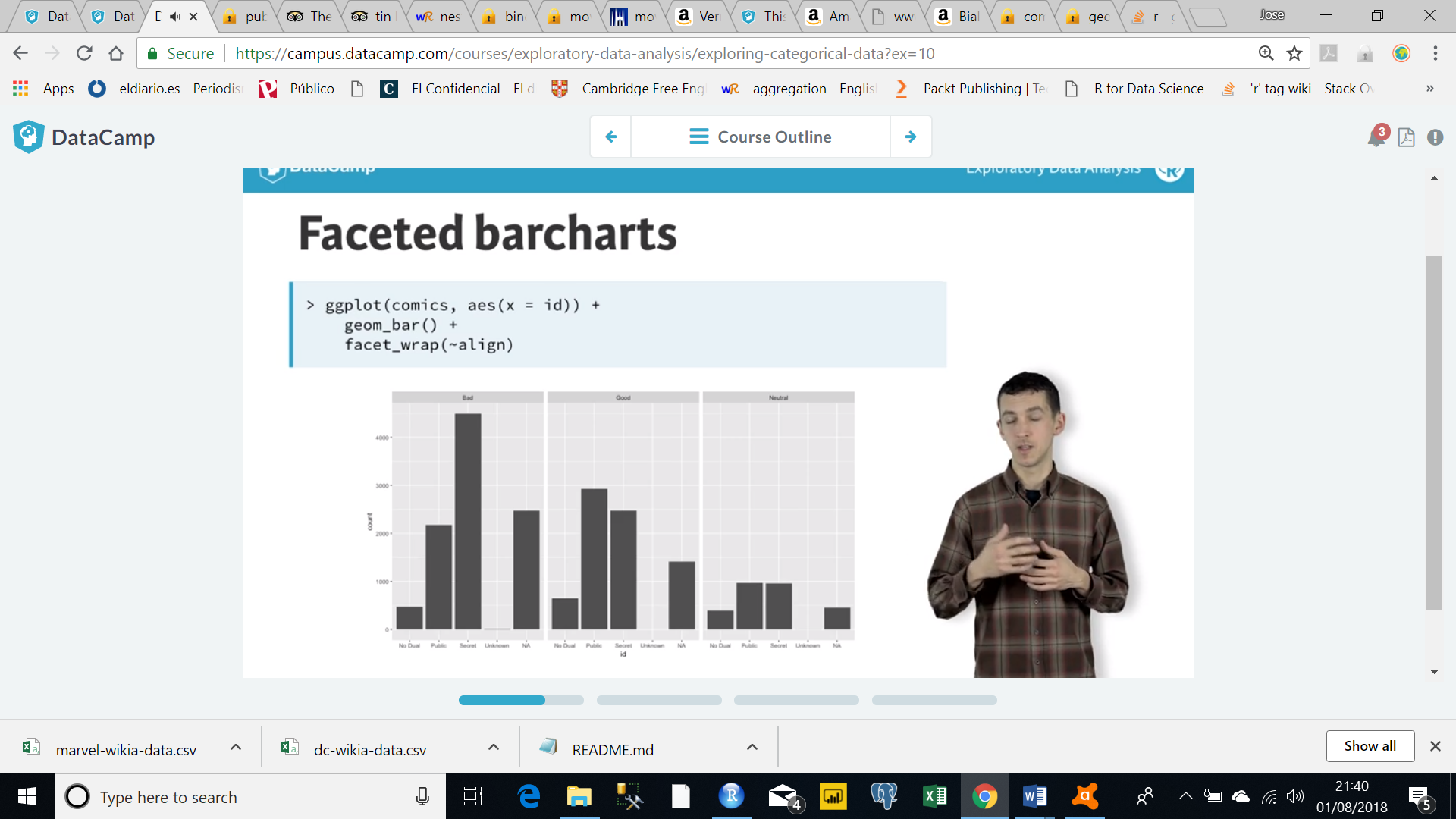
Another useful way to form a distribution of a single variable is to condition it on a particular value of another variable.

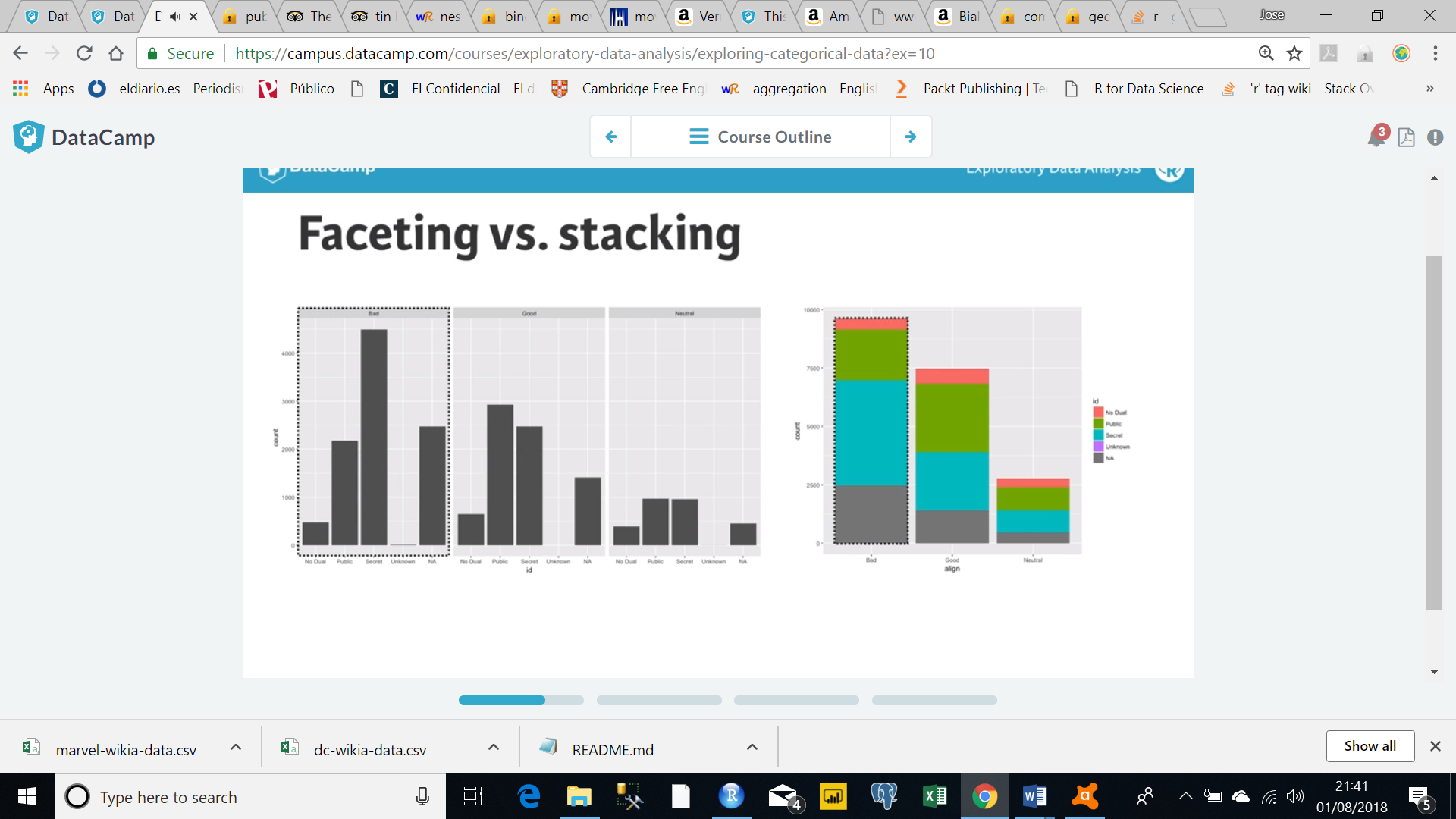
E.g., distribution of comics$id for all comics$align = neutral characters

Two options

1. Filter the dataset and build the barchar using only the cases where comics$align = neutral
2. We can use a technique called FACETING which breaks the data into subsets based on levels of a categorical variable, and then construct a plot for each







PIECHARTS

Task

align

Bad, Good, Natural

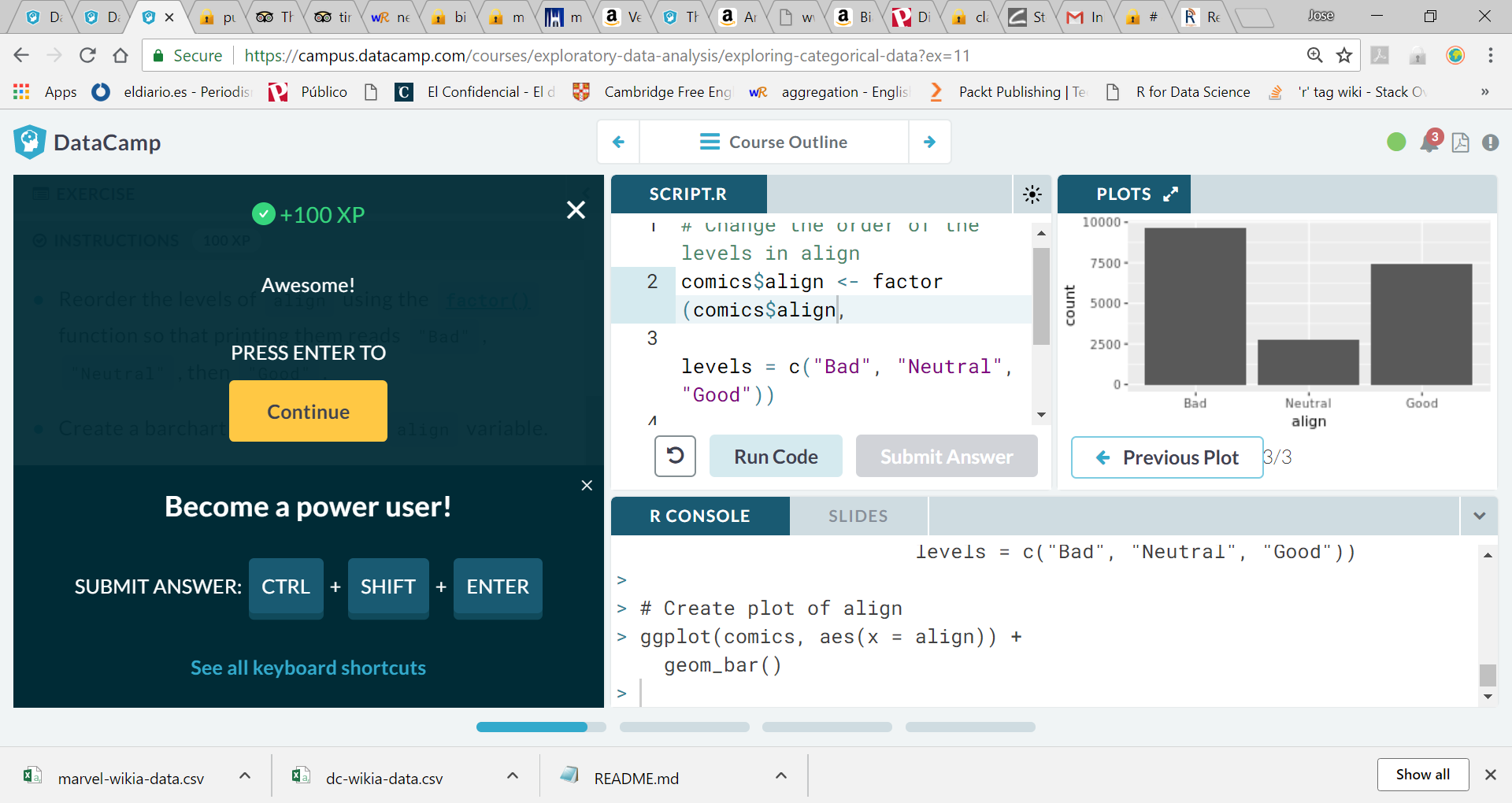
Reorder the levels of align using the factor() function so that printing them reads "Bad", "Neutral", then "Good"

# Change the order of the levels in align

comics$align <- factor(comics$align, levels = c("Bad", "Neutral", "Good"))

# Create plot of align

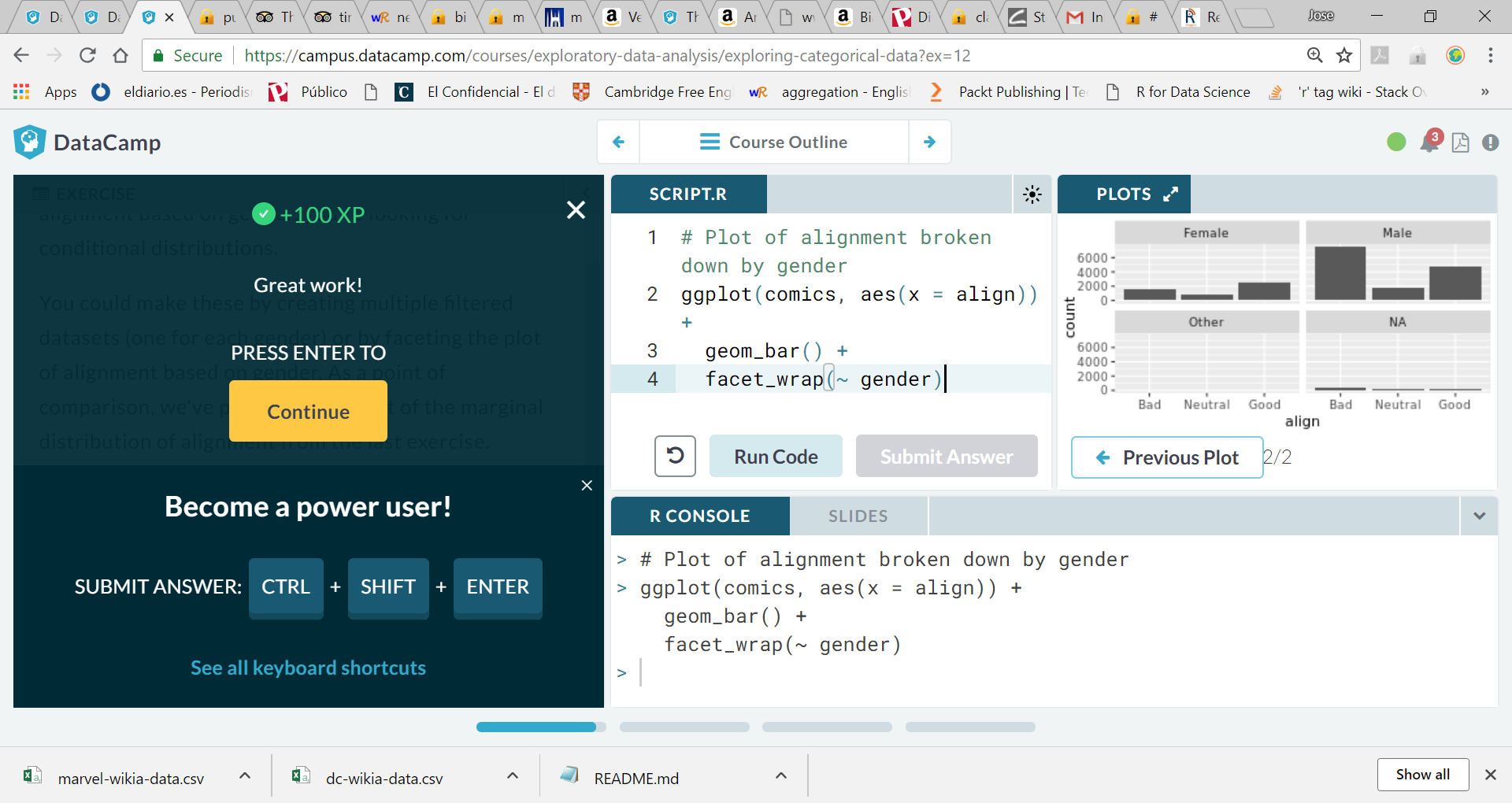
ggplot(comics, aes(x = align)) + geom\_bar()



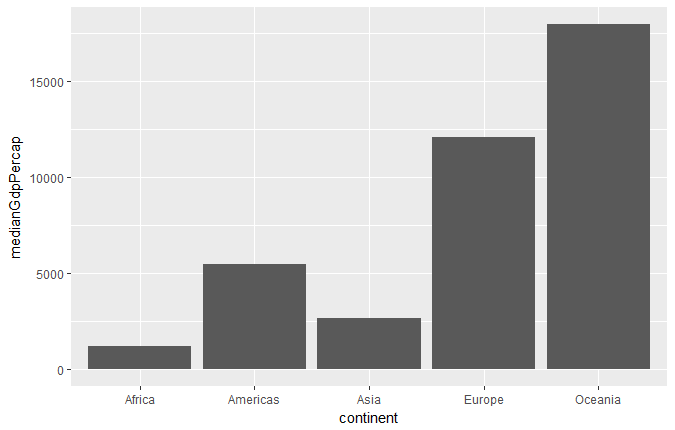
Conditional barchart

Now, if you want to break down the distribution of alignment based on gender, you're looking for **conditional distributions**.

You could make these by creating multiple filtered datasets (one for each gender) or by faceting the plot of alignment based on gender. As a point of comparison, we've provided your plot of the marginal distribution of alignment from the last exercise.



ggplot(by\_continent, aes(x=continent, y=medianGdpPercap)) + geom\_col()



DataCamp + Introduction to the Tidyverse

Course description

* you'll learn the intertwined processes of data manipulation and visualization through the tools dplyr and ggplot2.

## DataCamp + Data Visualization with ggplot2

Introduction

* ggplot2 has become the go-to tool for flexible and professional plots in R.
* We’ll examine the first three essential layers for making a plot –
  + Data,
  + Aesthetics and
  + Geometries.
* Blah

Course

1. Introduction
2. Data
3. Aesthetics
4. Geometries
5. qplot and wrap-up
6. Introduction

* we’ll get you into the right frame of mind for developing meaningful visualizations with R.
* You’ll understand that as a communications tool, visualizations require you to think about your **audience** first.
* You’ll also be introduced to the basics of ggplot2 - the 7 different grammatical elements (layers) and aesthetic mappings.

1. Data

* The structure of your data will dictate how you construct plots in ggplot2.
* We’ll explore the iris dataset from several different perspectives to showcase this concept.
* We’ll see that making your data conform to a structure that matches the plot in mind will make the task of visualization much easier through several R data visualization examples.

1. Aesthetics

* Aesthetic mappings are the cornerstone of the grammar of graphics plotting concept.
* This is where the magic happens - converting continuous and categorical data into visual scales that provide access to a large amount of information in a very short time.
* In this chapter you’ll understand how to choose the best aesthetic mappings for your data.

1. Geometries

* A plot’s geometry dictates what visual elements will be used.
* we’ll familiarize you with the geometries used in the three most common plot types you’ll encounter - scatter plots, bar charts and line plots.
* We’ll look at a variety of different ways to construct these plots.

1. qplot and wrap-up

* In this chapter you'll learn about qplot;
* it is a quick and dirty form of ggplot2.
* It’s not as intuitive as the full-fledged ggplot() function but may be useful in specific instances.
* This chapter also features a wrap-up video and corresponding data visualization exercises.

1. Introduction

ggplot()

* Base data layer
* Aesthetics
* Add geom layers

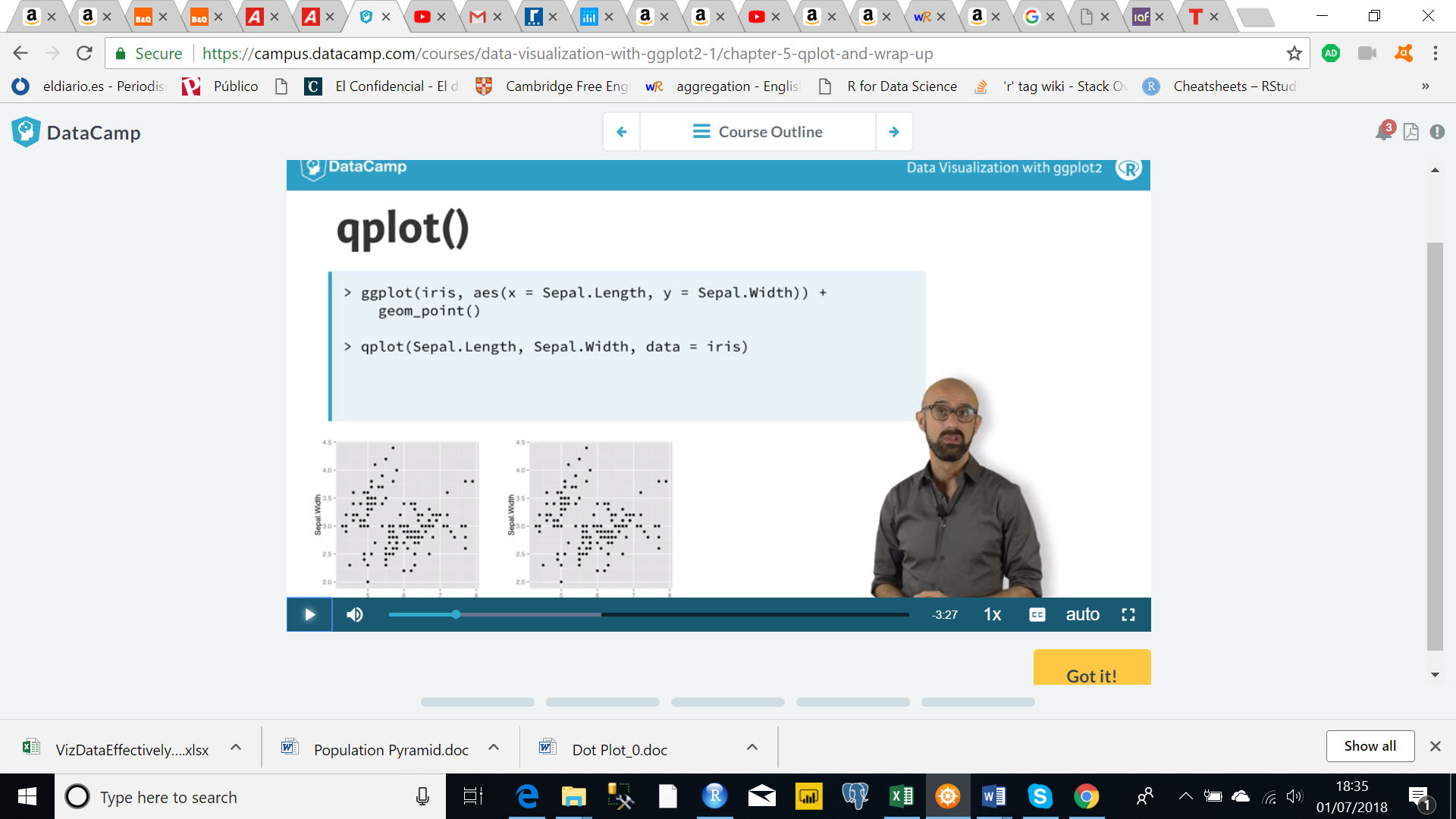
qplot()

* Easy, quick and dirty

This is a scatter plot with two continuous variables

> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom\_point()

> qplot(Sepal.Length, Sepal.Width, data = iris)



## Data Science (Fundraising) Book

**Creating Data Visualisations with ggplot**

* Wickham (2009) completely changed the way R users plotted graphs by using the principles of grammar of graphics (Wilkinson 2006) and creating the ggplot library.
* Like the grammar of language, charts can be broken into individual components.

ggplot(data = <your\_data>,

aes(x = <your\_x\_var>,

y = <your\_y\_var>,

size = <your\_size\_var>,

color = <your\_color\_var>)) + geom\_<of\_your\_choice>

**geom**

* A geometric object or **geom** tells ggplot which **plot** we want to see.
* Some of the most common geom objects are:
  + geom\_bar: frequency bar graph (unless you pass an already calculated column and specify identity = TRUE)
  + geom\_line: adds a line graph layer
  + geom\_point: adds a scatter plot layer in which the X-axis variable doesn’t need to be a continuous variable
  + geom\_text: adds a text layer (useful for annotation or name plots)

**Aesthetics (aes) Mapping**

there are two ways to assign the variable

1. in the ggplot function
   * the code following the initial ggplot call will use those same mappings unless you explicitly supply other values
2. in the geom\_\* function
   * this way you can limit the mappings to that geom
   * this approach becomes useful when you have multiple geom objects

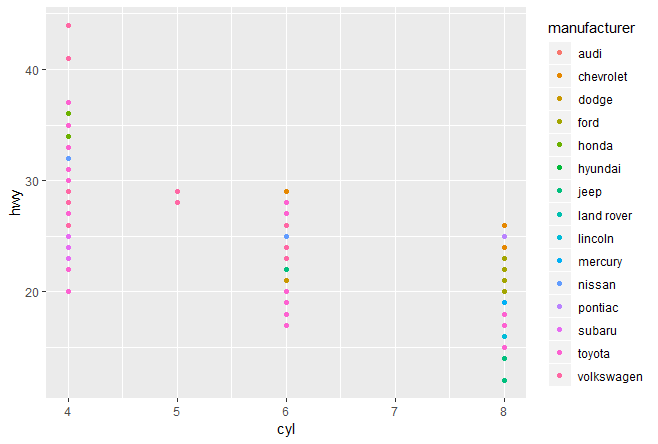
#We will use the mpg dataset that comes with the ggplot package

?mpg #see the documentation for this dataset

glimpse(mpg) #see the data

ggplot(data = mpg, aes(x = cyl, y = hwy, color = manufacturer)) +

geom\_point()



creates the same plot as:

ggplot() +

geom\_point(data = mpg, aes(x = cyl, y = hwy, color = manufacturer))

**Using Scales Functions**

* you can use different scale functions to control how you want the axes and data points on the graphics to look.
* You can change the format of the axes to
  + Discrete
  + Continuous
  + Date
  + Other
* Similarly, you can change the look and feel of the graphed data points by changing the
  + Colours
  + Sizes
  + Shapes
* There are many options, which are best explored by going through the documentation:
* <http://ggplot2.tidyverse.org/reference/#section-scales>
* The following are some common scales:

scale\_x\_continuous or scale\_y\_continuous

* This tells ggplot the variable plotted on the axis is a continuous variable
* We can specify
  + the position of the gridlines,
  + axis tic kmark lables,
  + range of possible values of the axis,
  + and other controls.

**scale\_color\_\*** or **scale\_fill\_\***

* these tells ggplot how and which colour schemes to use for the plotted data points.
* scale\_color\_\* 🡪 colouring unfillable geom objects such as geom\_line and geom\_text
* scale\_fill\_\* 🡪 filling other fillable geom objects such as geom\_bar and geom\_area

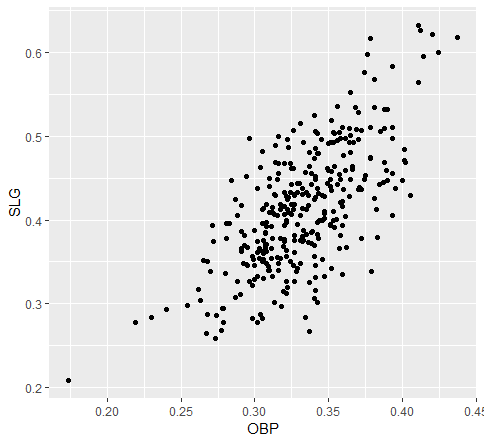
## Tips

### Filtered plot

Use filter() to create a scatterplot for SLG as a function of OBP among players who had at least 200 at-bats.

# Scatterplot of SLG vs. OBP

mlbBat10 %>% filter( AB>=200) %>% ggplot(aes(x = OBP, y = SLG)) + geom\_point()



## Paletas

Convertor: <https://kilianvalkhof.com/2016/css-html/css-hexadecimal-colors-with-transparency-a-conversion-tool/>

Example:

ggplot( …

+ scale\_color\_manual(values= c("#008A80", "#BF1238", "#CF7600"))

**PALETAS**

Paleta5 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914")

Paleta6 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2")

Paleta7 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999")

Paleta8 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4")

15 (5 x 3)

Paleta15 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914")

7

Paleta7 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999")

10

Paleta10 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7")

11

Paleta10 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF")

Paleta5 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914")

Paleta5.2 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914")

Paleta4.10 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3","#008A80", "#BF1238", "#CF7600", "#1D97C3")

Paleta10 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF")

Paleta10.7 <- c("#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF","#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF")

,"#008A80", "#BF1238", "#CF7600", "#1D97C3", "#EBB914", "#7973C2", "#999999","#80A6A4","#7973C2FF","#FF9DA7","#FF9DA7","#CF7600FF")

# Learning ggplot2

* ([URL](https://ggplot2.tidyverse.org/))
* <https://r4ds.had.co.nz/data-visualisation.html>
* <https://r4ds.had.co.nz/graphics-for-communication.html>
* <http://shop.oreilly.com/product/0636920052807.do>
* <https://www.amazon.com/dp/1491978600/>
* <https://www.amazon.co.uk/R-Graphics-Cookbook-Winston-Chang/dp/1449316956/ref=tmm_pap_swatch_0?_encoding=UTF8&qid=1563353642&sr=8-3>
* <https://www.amazon.co.uk/ggplot2-Elegant-Graphics-Data-Analysis/dp/0387981403/ref=tmm_pap_swatch_0?_encoding=UTF8&qid=1563353725&sr=8-3>
* <https://github.com/hadley/ggplot2-book>
* “The Layered Grammar of Graphics”, <http://vita.had.co.nz/papers/layered-grammar.pdf>.

# R for Data Science > 3 Data visualisation

* <https://r4ds.had.co.nz/data-visualisation.html>
* “The simple graph has brought more information to the data analyst’s mind than any other device.” — John Tukey
* ggplot2 implements the **grammar of graphics**, a coherent system for describing and building graphs.
* If we need to be explicit about where a function (or dataset) comes from, we’ll use the special form package::function()

## 1st step – ggplot()

* With ggplot2, you begin a plot with the function ggplot().
* ggplot() creates a coordinate system that you can add layers to.
* The first argument of ggplot() is the dataset to use in the graph.
* So ggplot(data = mpg) creates an empty graph

## 2ns step – more layers

* You complete your graph by adding one or more layers to ggplot().
* ggplot2 comes with many geom functions that each add a different type of layer to a plot.
* Each geom function takes a **mapping argument**, which defines how variables in your dataset are mapped to visual properties.
  + mapping = aes(x = , y = )
* The function geom\_point() adds a layer of points to your plot, which creates a scatterplot.

ggplot(data = <DATA>) +

<GEOM\_FUNCTION>(mapping = aes(<MAPPINGS>))

## 3rd step – Aesthetics by variables

* You can add a third variable to a two-dimensional scatterplot by mapping it to an aesthetic.
* An **aesthetic** is a visual property of the objects in your plot.
* Aesthetics:
  + size,
  + shape: can only be mapped to a categorical variable
  + color
    - when mapped to a continues variable it picks a range of gradients for the same colour
    - in a bar chart this would be the colour of each bar’s border
  + fill 🡪 in a bar chart this would be the colour of each bar
  + group 🡪 you can set the group aesthetic to a categorical variable to draw multiple objects.
  + alph*a* 🡪 controls the transparency of the points
  + stroke 🡪 (only for continuous variables)
  + linetype 🡪 similar to group
  + X and Y: the x and y locations of a point are themselves aesthetics, visual properties that you can map to variables to display information about the data
* You can convey information about your data by mapping the aesthetics in your plot to the variables in your dataset.
* Every geom function in ggplot2 takes a mapping argument.
* However, not every **aesthetic** works with every
  + geom
  + type of variable (categorical / numeric)

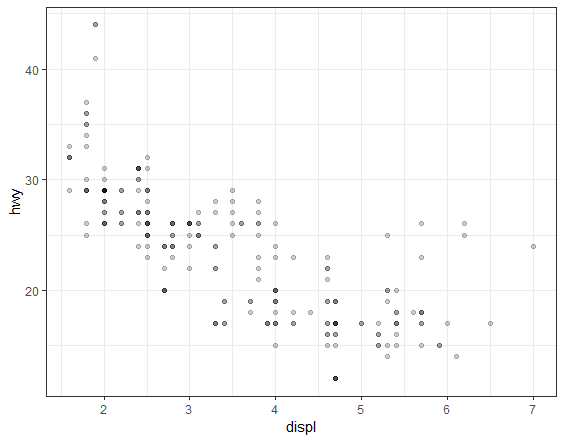
## 4th step – Aesthetics for all

* Here, the color doesn’t convey information about a variable, but only changes the appearance of the plot
* We do this by setting the aesthetic by name as an argument of your geom function outside of aes()
* ggplot(data = mpg) +
* geom\_point(mapping = aes(x = displ, y = hwy), color = "blue")
* Again: color, size, shape & alpha
* Not sure what alpha means here …

D:/JBL/Library/R/R Tutorials/ggplot2/harvard/Rgraphics/dataSets

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy), alpha = 0.2)



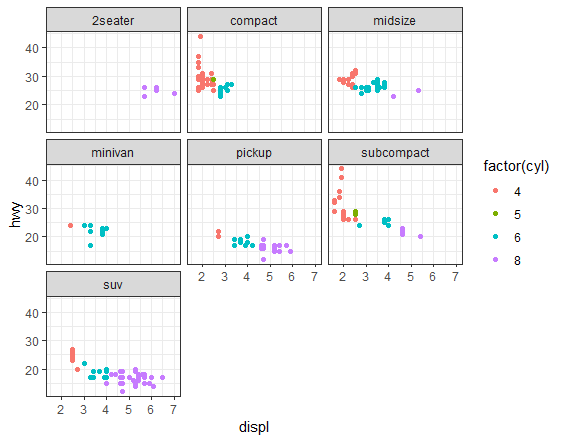
## [5th step: Facets]

### facet\_wrap(~ cat variable)

* One way to add additional variables is with **aesthetics**.
* Another way, particularly useful for categorical variables, is to split your plot into **facets**, subplots that each display one subset of the data.
* To facet your plot by a single variable, use **facet\_wrap()**.
* The first argument of facet\_wrap() should be a formula, which you create with ~ followed by a variable name (here “formula” is the name of a data structure in R, not a synonym for “equation”).
* The variable that you pass to facet\_wrap() should be **discrete**.
* facet\_wrap(~ class, nrow = 2); alternatively ‘ncol’

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, color = factor(cyl))) + facet\_wrap(~ class)



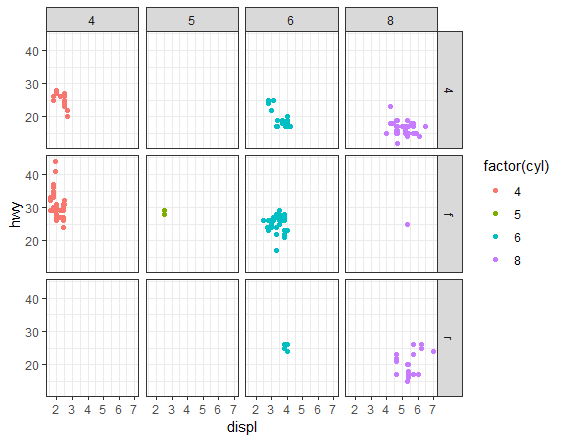
### facet\_grid(cat variable A for rows ~ cat variable B for columns)

* To facet your plot on the combination of two variables

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, color = factor(cyl))) +

facet\_grid(drv ~ cyl)



ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy)) +

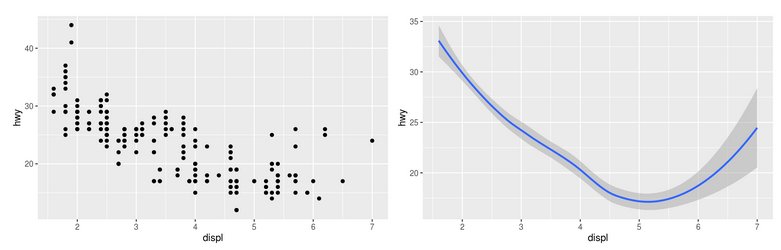
facet\_grid(drv ~ .)

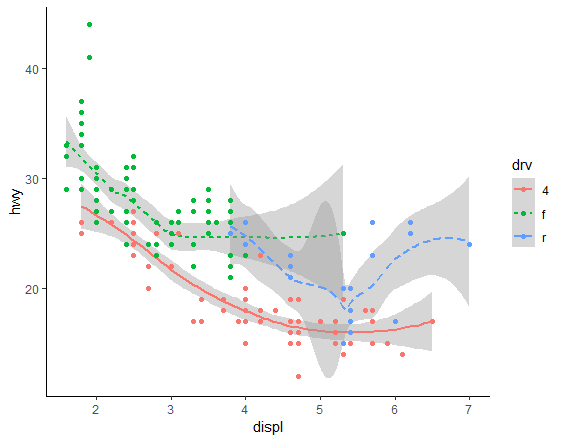
will returns a plot with only the rows by dvr

## 3.6 Geometric objects (geoms)

## Different geoms

* This two plots describe the same data by using different geoms (geom\_point() and geom\_smooth())



* A **geom** is the geometrical object that a plot uses to represent data.
* People often describe plots by the type of geom that the plot uses. For example,
  + bar charts use bar geoms,
  + line charts use line geoms,
  + boxplots use boxplot geoms, etc.
  + Scatterplots break the trend; they use the point geom
* As we see above, you can use different geoms to plot the same data.
* ggplot2 provides over 30 geoms, and extension packages provide even more (see <https://www.ggplot2-exts.org> for a sampling).
* show.legend = FALSE gets rid of the legends bit: 
* To display multiple geoms in the same plot, add multiple geom functions to ggplot()
* 
* The ‘Good’ code below introduces some duplication in our code.

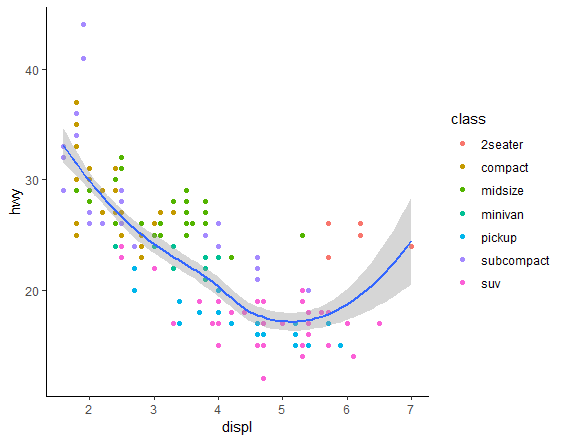
|  |  |
| --- | --- |
| Good | Better |
| ggplot(data = mpg) +  geom\_point(mapping = aes(x = displ, y = hwy)) +  geom\_smooth(mapping = aes(x = displ, y = hwy)) | ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  geom\_point() +  geom\_smooth() |

* This makes it possible to display different aesthetics in different layers.

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

geom\_point(aes(color=class)) +

geom\_smooth()



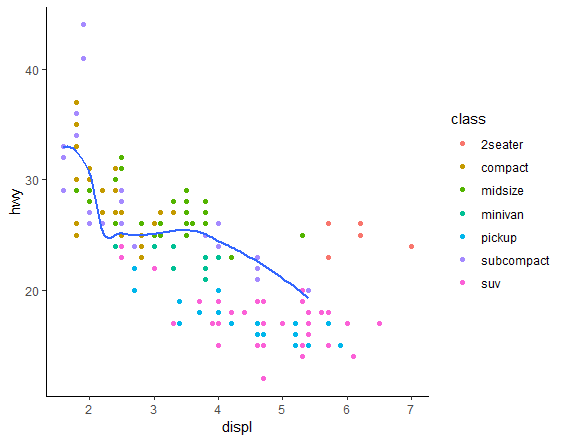
## Using different data for each layer

* You can use the same idea to specify different data for each layer.

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

geom\_point(mapping = aes(color = class)) +

geom\_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)



* se : logical value. If TRUE, confidence interval is displayed around smooth.

3.6.1 Exercises

What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?

## 3.7 Statistical transformations

* **Bar charts**, **histograms**, and **frequency polygons**
  + bin your data and then
  + plot bin counts, the number of points that fall in each bin.
* **smoothers** fit a model to your data and then plot predictions from the model.
* **boxplots** compute a robust summary of the distribution and then display a specially formatted box.

### Bar Charts

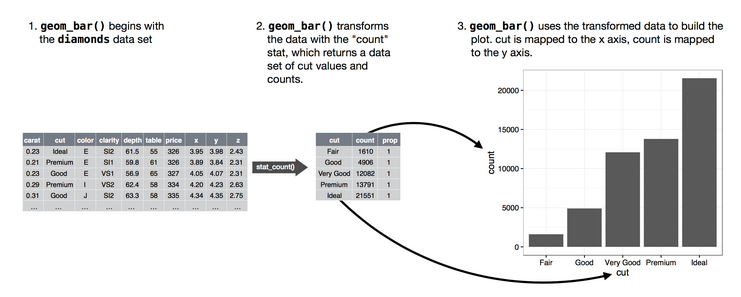
* **Bar charts** seem simple, but they are interesting because they reveal something subtle about plots.
* Consider a basic bar chart, as drawn with geom\_bar().
* The following chart displays the total number of diamonds in the diamonds dataset, grouped by cut.
* The diamonds dataset comes in ggplot2 and contains information about ~54,000 diamonds, including the price, carat, color, clarity, and cut of each diamond.
* The chart shows that more diamonds are available with high quality cuts than with low quality cuts.

### geom\_bar() vs geom\_col()

* There are two types of bar charts: geom\_bar() and geom\_col().
* **geom\_bar()** makes the height of the bar **proportional** to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights).
* If you want the heights of the bars to represent values in the data, use geom\_col() instead.
* geom\_bar() uses stat\_count() by default: it counts the number of cases at each x position.
* geom\_col() uses stat\_identity(): it leaves the data as is.

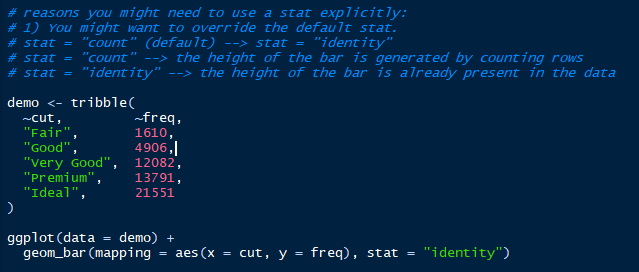
### Stat

* The **algorithm** used to calculate new values for a graph is called a **stat**, short for statistical transformation.
* You can generally use geoms and stats interchangeably.
* For example, you can recreate the previous plot using stat\_count() instead of geom\_bar()
* 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. To see a complete list of stats, try the ggplot2 cheatsheet.
* The figure below describes how this process works with geom\_bar()



* every geom has a default stat; and every stat has a default geom.
* This means that you can typically use geoms without worrying about the underlying statistical transformation.
* There are three reasons you might need to use a stat explicitly:
  1. You might want to override the default stat.
  2. You might want to override the default mapping from transformed variables to aesthetics
  3. You might want to draw greater attention to the statistical transformation in your code.
* adfsdaf

1) You might want to override the default stat

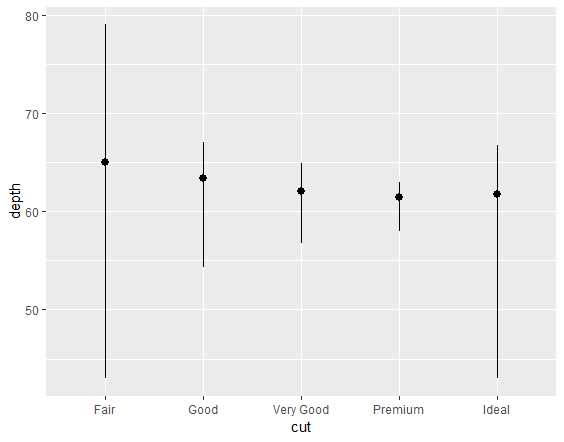
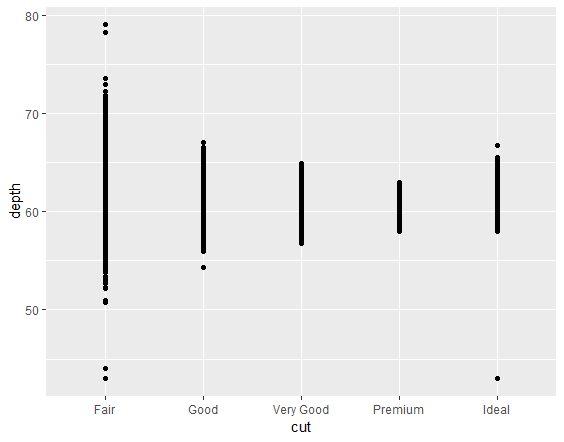


1. You might want to override the default mapping from transformed variables to aesthetics
   * For example, you might want to display a bar chart of proportion, rather than count:

ggplot(data = diamonds) +

geom\_bar(mapping = aes(x = cut, y = ..prop.., group = 1))

1. You might want to draw greater attention to the statistical transformation in your code.



ggplot(data = diamonds) + geom\_point(mapping = aes(x = cut, y = depth))

ggplot(data = diamonds) +

stat\_summary(

mapping = aes(x = cut, y = depth),

fun.ymin = min,

fun.ymax = max,

fun.y = median

)

## TIPS

|  |  |  |  |
| --- | --- | --- | --- |
| Geom | X | Y |  |
| geom\_point() | Discrete / Categorical | Discrete | A scatterplot with two categorical variables is only useful only to see existent combinations |
| geom\_point() | Discrete | Continuous | Similar to a bar chart but with points |
|  |  |  |  |