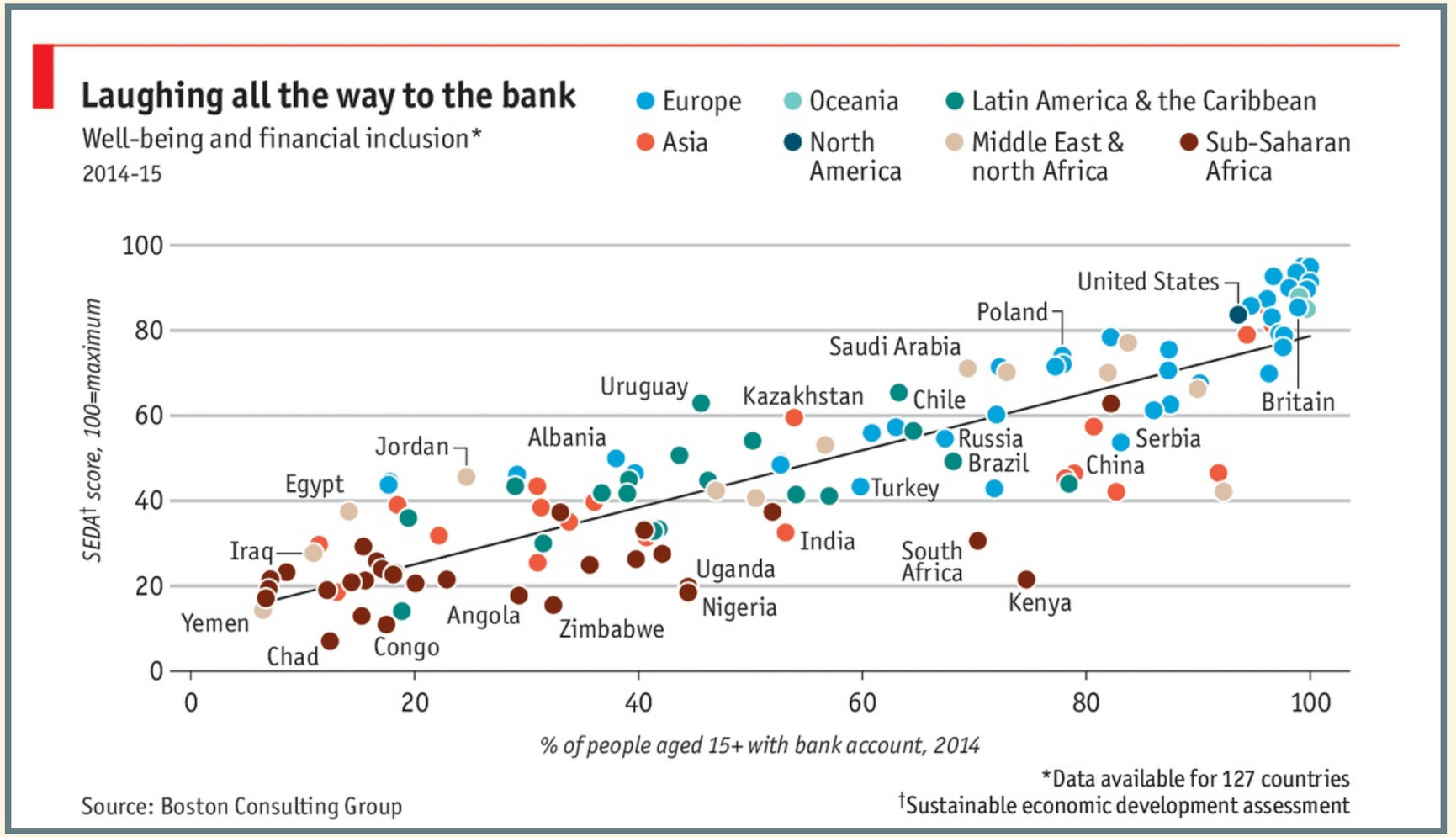
LECTURE 4: GENERATING 2D GRAPHICS WITH R. PART II

OCTOBER 11, 2016

EXERCISE 1: THE ECONOMIST DATA

For practice, you will try to recreate a plot published in the Economist issue of July 20th, 2016 reflecting the relationship between people's well-being and their financial inclusion.



- The original graph is available here
- You will generate this figure step by step through a series of
  exercises using the tools you've just learned and will learn about.

THE PLOT IS GENERATED WITH THE DATA FROM:

- The Boston Consulting Group's report on well-being for each country which includes their Sustainable Economic Development Assessment (SEDA) (well-being) scores[1]
- The World Bank Global Findex database which provides the indices for financial inclusion (including percent of people aged 15 or more with a bank account).

- Download “Lec3_ex.Rmd” file containing instructions for the exercise from the class website.
- Download also the data for this exercise (`EconomistData.csv`). Read the file into R with a following command.

```
# Change the path
dat <- read.csv("../data/EconomistData.csv")
```

- Follow the instructions in the file and complete Exercise 1.

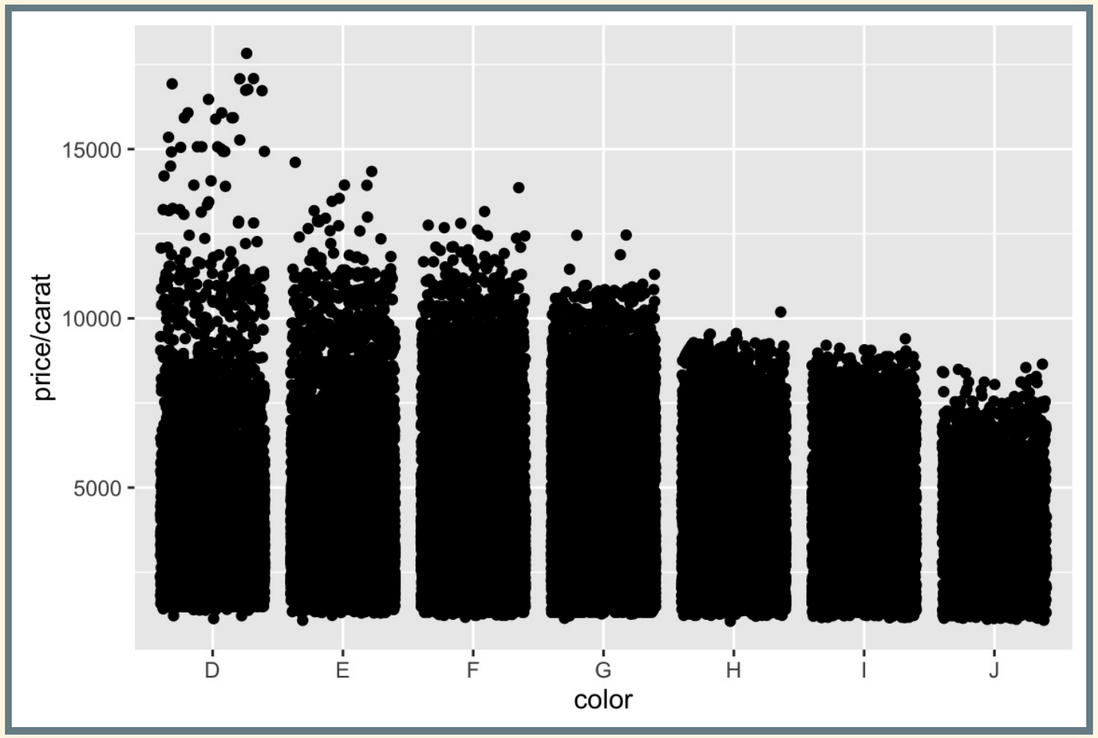
STATISTICAL TRANSFORMATIONS

- Now, we will go back to the `diamonds` dataset and return to the Economist plot later.
- So far we have only plotted **the scatter plots** for the `diamonds` data where each raw data point was mapped to a single (x,y) coordinates pair.
- However, we could be interested in plots that require some **statistical transformations**.

1. Examples of plots involving statistical transformations:
    - boxplots,
    - histograms,
    - smoothers,
    - bar charts
2. These types of plots require further computations or transformations.
    - **boxplots:** the the median, lower and upper quartiles,
    - **histograms:** group the values into bins,
    - **smoothers:** prediction lines / predicted y-values,
    - **bar charts:** number of class occurrences.

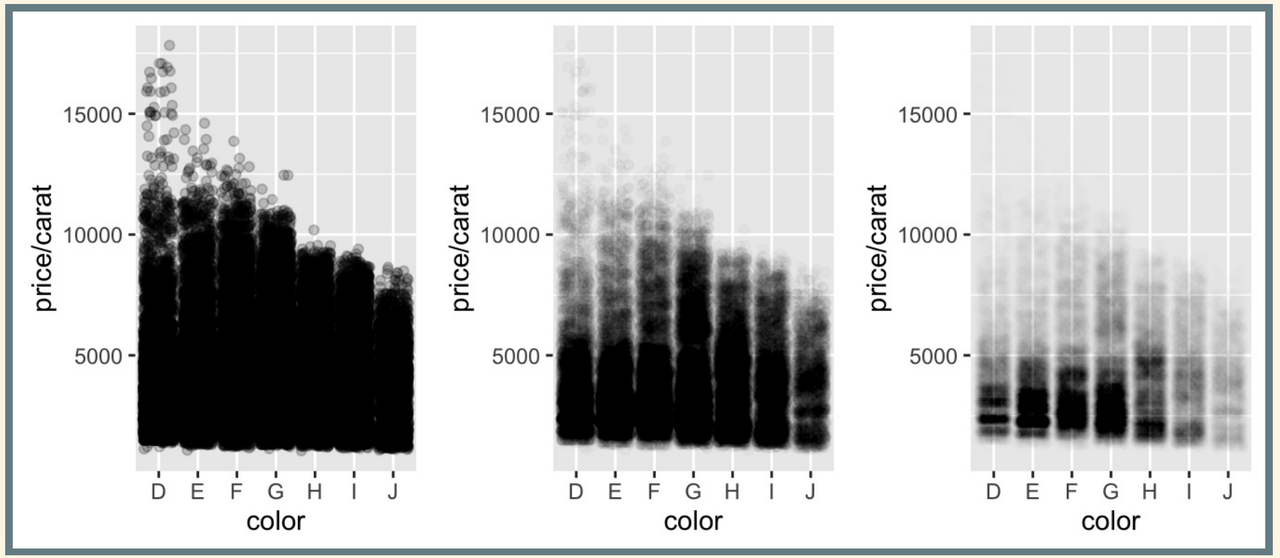
BOXPLOTS AND JITTERED POINTS

```
ggplot(data = diamonds, aes(x = color, y =price/carat)) +
  geom_jitter()
```



ALPHA PARAMETER FOR TRANSPARENCY

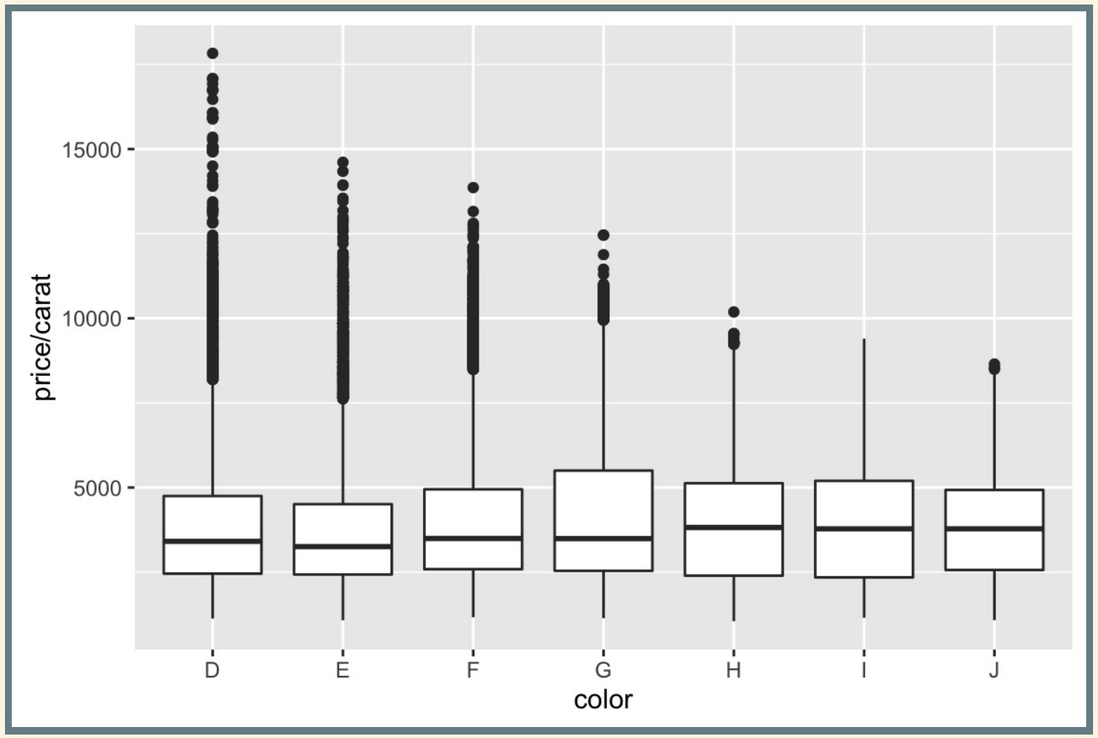
```
j1 <- ggplot(data = diamonds, aes(x = color, y =price/carat)) +
  geom_jitter(alpha = I(1/5))
j2 <- ggplot(data = diamonds, aes(x = color, y =price/carat)) +
  geom_jitter(alpha = I(1/50))
j3 <- ggplot(data = diamonds, aes(x = color, y =price/carat)) +
  geom_jitter(alpha = I(1/200))
# We use grid.arrange from gridExtra to display multiple plots
library(gridExtra)
grid.arrange(j1, j2, j3, ncol = 3)
```



BOX PLOT TRANSFORMATION

Plotting a summary (less data) can be more insightful.

```
ggplot(data = diamonds, aes(x = color, y =price/carat)) +
  geom_boxplot()
```

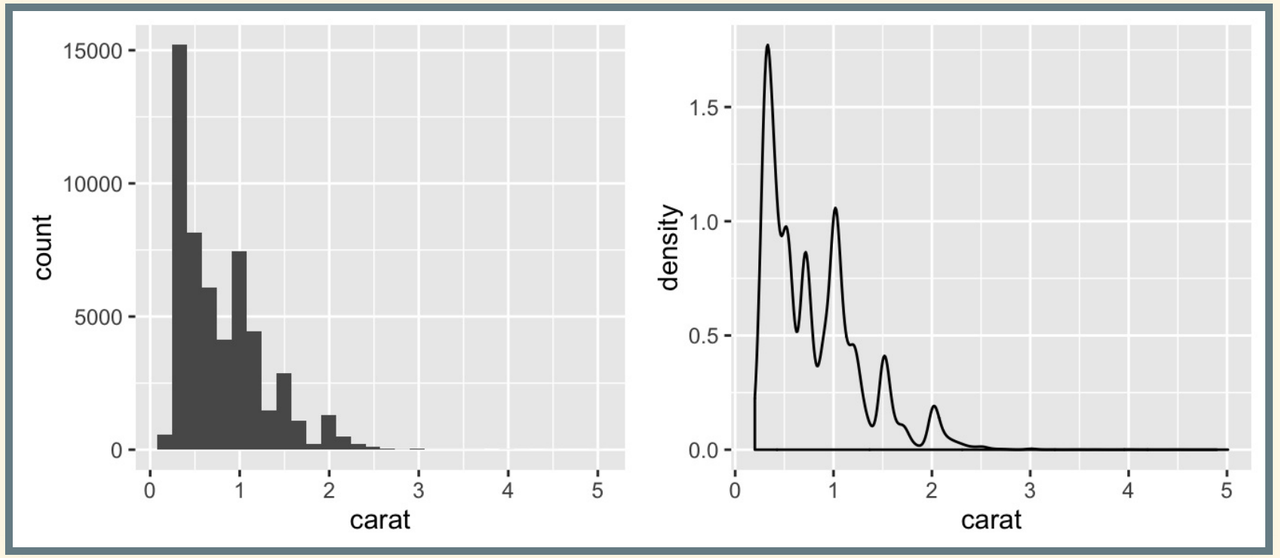


HISTOGRAM AND DENSITY PLOTS

Below we plot the distribution of the weights (carat) of the diamonds.

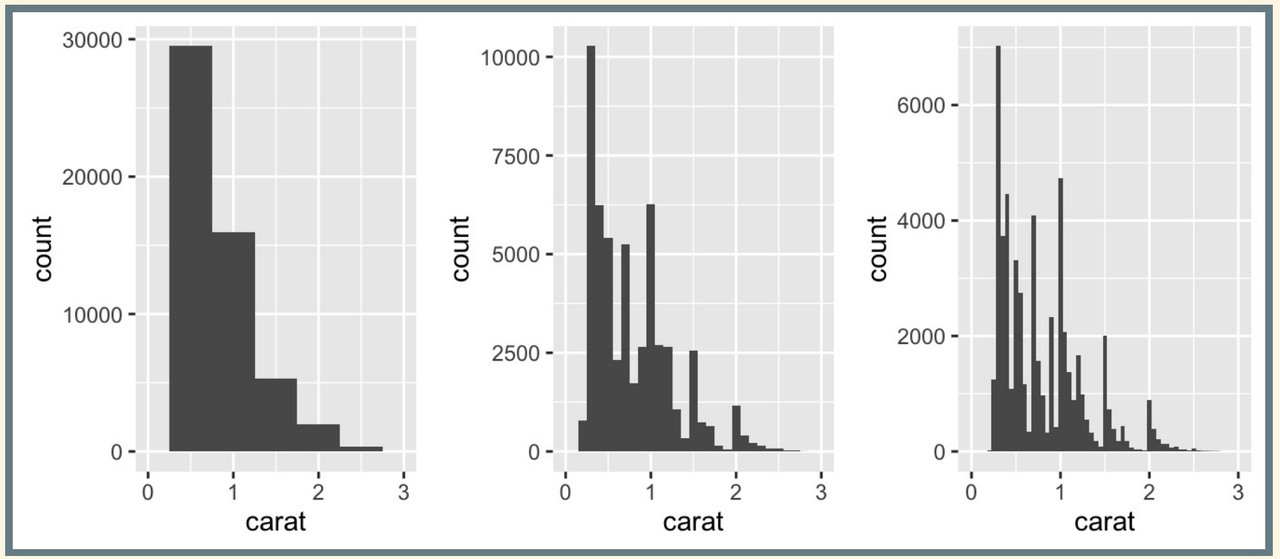
```
h <- ggplot(data = diamonds, aes(x = carat)) + geom_histogram()
d <- ggplot(data = diamonds, aes(x = carat)) + geom_density()
grid.arrange(h, d, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

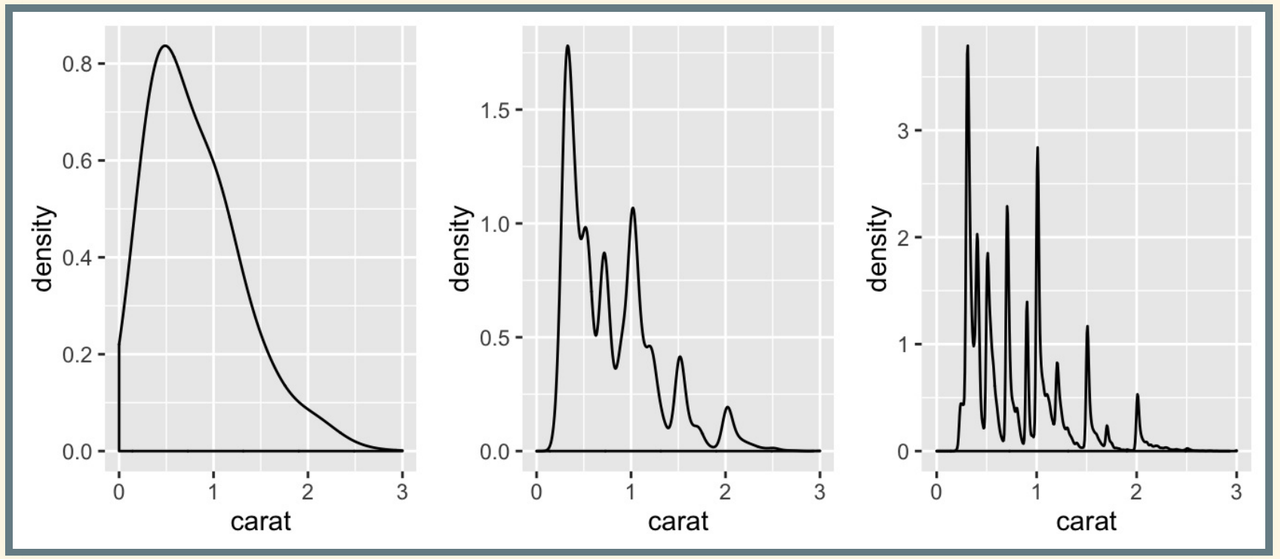


- For the density plot, the `bw` (the smoothing bandwidth) and `adjust` argument controls **the degree of smoothness** (high values of adjust produce smoother plots).
- For the histogram, the smoothness is controlled with `bins` **and** `binwidth` arguments. (Break points can also be specified explicitly, using the `breaks` argument.)

```
p <- ggplot(data = diamonds, aes(x = carat)) + xlim(0, 3)
h1 <- p + geom_histogram(binwidth = 0.5)
h2 <- p + geom_histogram(binwidth = 0.1)
h3 <- p + geom_histogram(binwidth = 0.05)
grid.arrange(h1, h2, h3, ncol = 3)
```

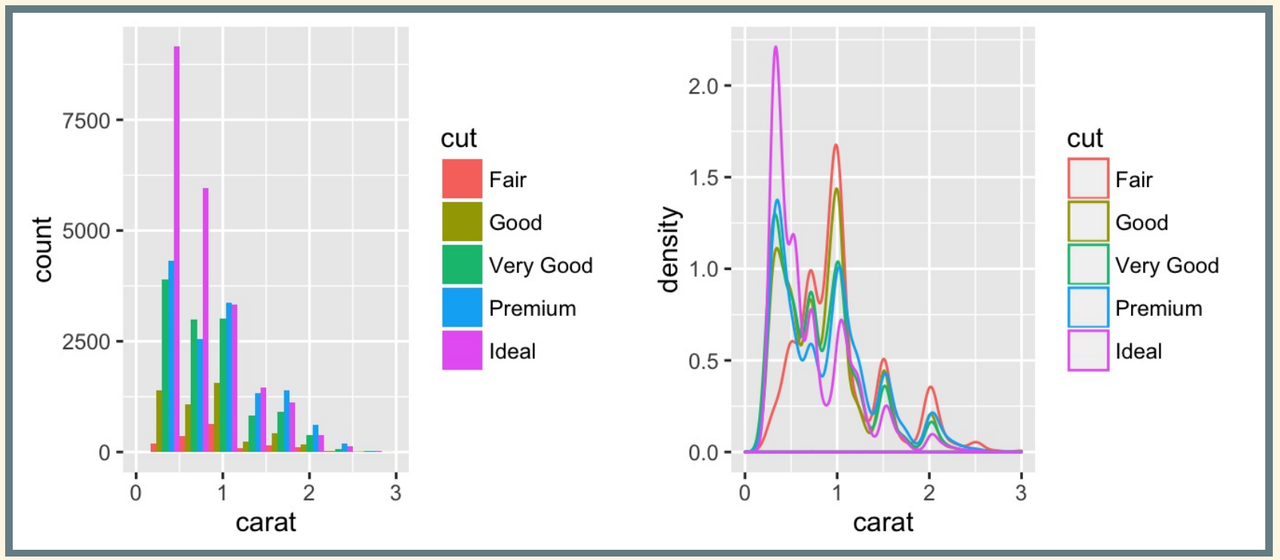


```
d1 <- p + geom_density(adjust = 5)
d2 <- p + geom_density(adjust = 1)
d3 <- p + geom_density(adjust = 1/5)
grid.arrange(d1, d2, d3, ncol = 3)
```



HISTOGRAMS CAN BE BROKEN DOWN

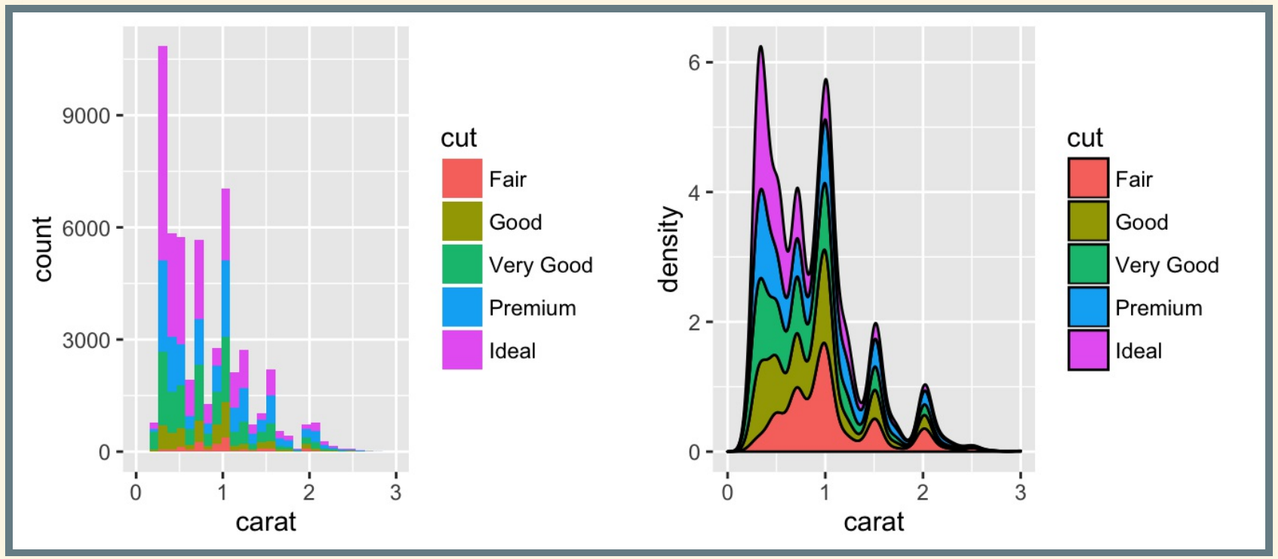
```
# Here we show grouping by diamonds cut.
h <- p + geom_histogram(aes(fill = cut), position = "dodge", bins = 10)
d <- p + geom_density(aes(color = cut))
grid.arrange(h, d, ncol = 2)
```



Instead of the marginal distribution, we can plot the components **stacked** on top of each other to see the contribution from each of group.

```
h <- p + geom_histogram(aes(fill = cut), position = "stack")
d <- p + geom_density(aes(fill = cut), position = "stack")
grid.arrange(h, d, ncol = 2)
```

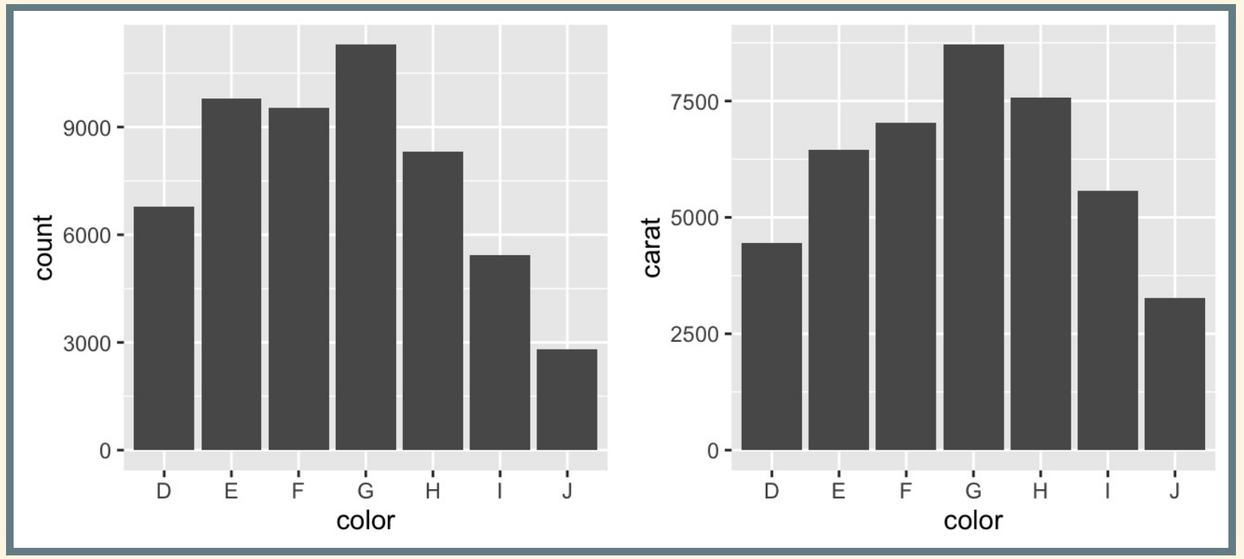
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



BAR CHARTS

- A discrete analogue of histogram is the bar chart, `geom_bar()`.
- Instead of partitioning the values into bins like histograms, the bar geom **counts the number of instances of each discrete class**. The counts are then plotted as columns for each distinct class.
- If you'd like include **unequal weights** for different observations, you can use the `weight` aesthetic.

```
b1 <- ggplot(diamonds, aes(x = color)) + geom_bar()
b2 <- ggplot(diamonds, aes(x = color)) + geom_bar(aes(weight = carat)) +
grid.arrange(b1, b2, ncol = 2)
```



The left plot shows counts and the right plot is the count weighted by `weight = carat` to show the total weight of diamonds of each color.

- As you see, in ggplot2 (unlike with barchart()) it is **not necessary tabulate** your values beforehand.
- However, if you summarized your data beforehange, you can still use `geom_bar` but using identity transformation `stat = "identity` rather than the default `stat = "count"`.

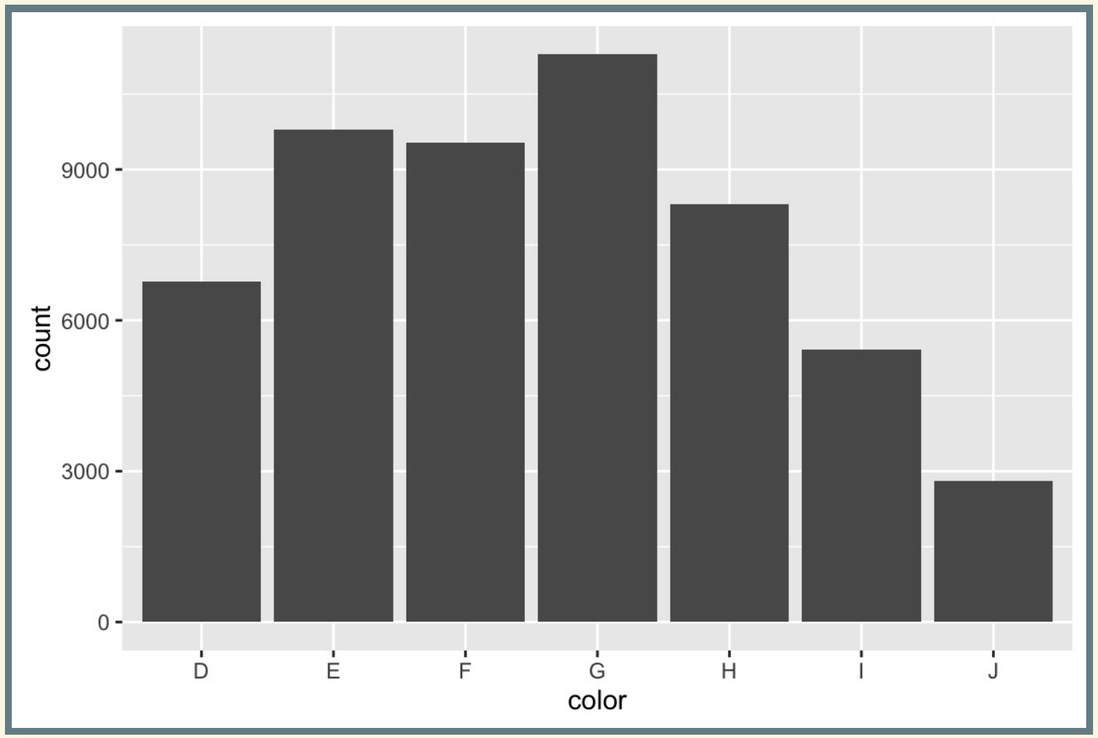
```
diamond.counts <- table(diamonds["color"])
df <- data.frame(diamond.counts)
colnames(df) <- c("color", "count")
df
```

```
##   color count
## 1     D  6775
## 2     E  9797
## 3     F  9542
## 4     G 11292
## 5     H  8304
## 6     I  5422
## 7     J  2808
```

```r
# The default option generates an error:
ggplot(df, aes(x=color, y=count)) +  geom_bar()

# Error: stat_count() must not be used with a y aesthetic.
```

```r
# You need to do the following:
ggplot(df, aes(x=color, y=count)) +  geom_bar(stat="identity")
```

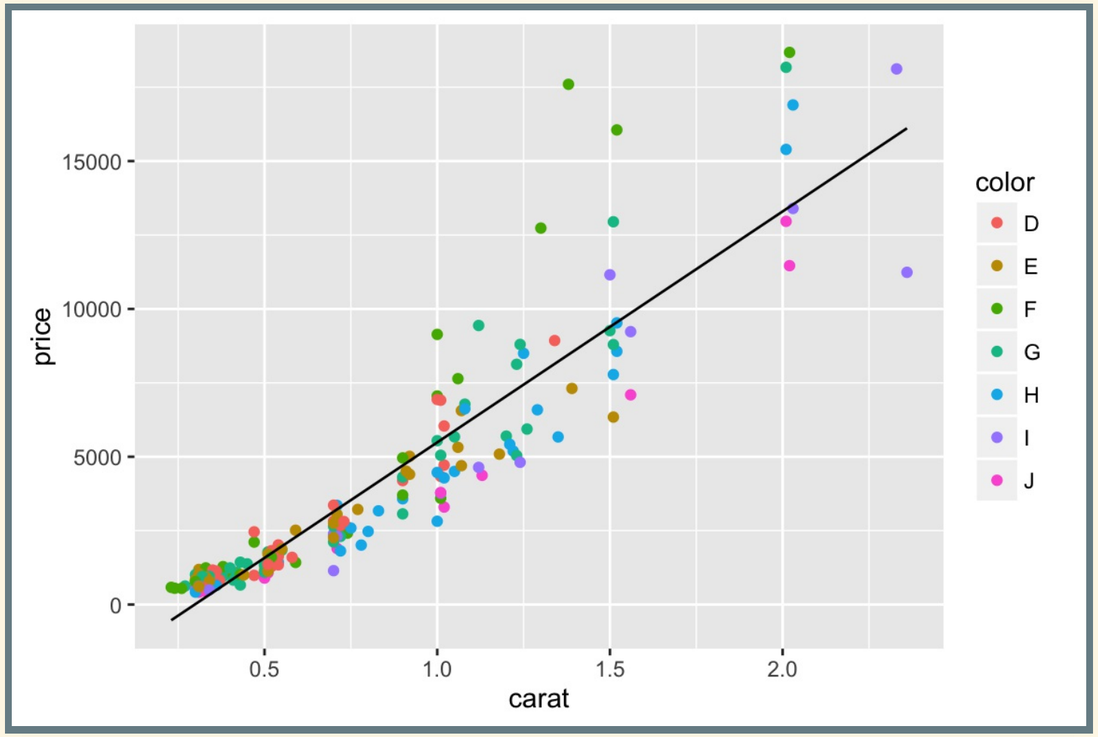


PREDICTION LINES

```
# Set seed to make the random sampling reproducible
set.seed(12345) # Make the sample reproducible
dsmall <- diamonds[sample(nrow(diamonds), 200), ]
# Fit a linear model with lm() function and use the formula y ~ x
dsmall$pred.price <- predict(lm(price ~ carat, data = dsmall))
```

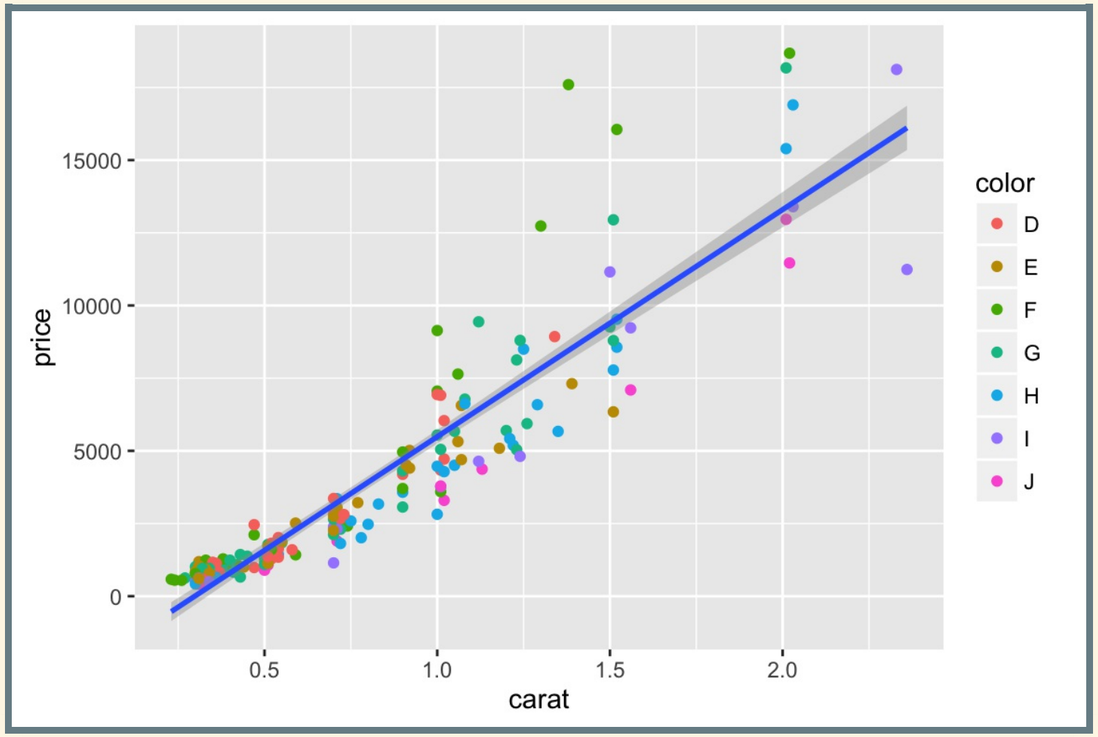
REGRESSION LINES CAN AND ADDED TO THE PLOT:

```
# Compute the predicted y-values using a lm() for linear models.
p1 <- ggplot(dsmall, aes(x = carat, y = price))
p1 + geom_point(aes(color = color)) + geom_line(aes(y = pred.price))
```



PREDICTION LINES WITH GGPLOT2

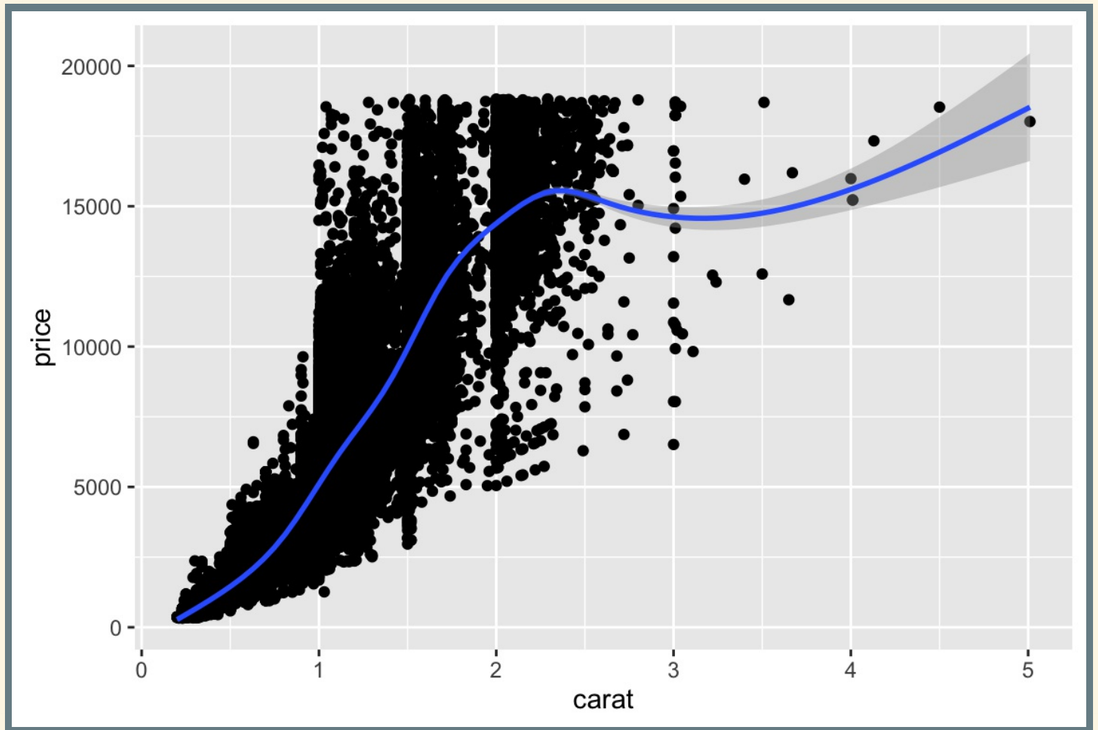
```
p1 + geom_point(aes(color = color)) + geom_smooth(method = "lm")
```



SMOOTHERS

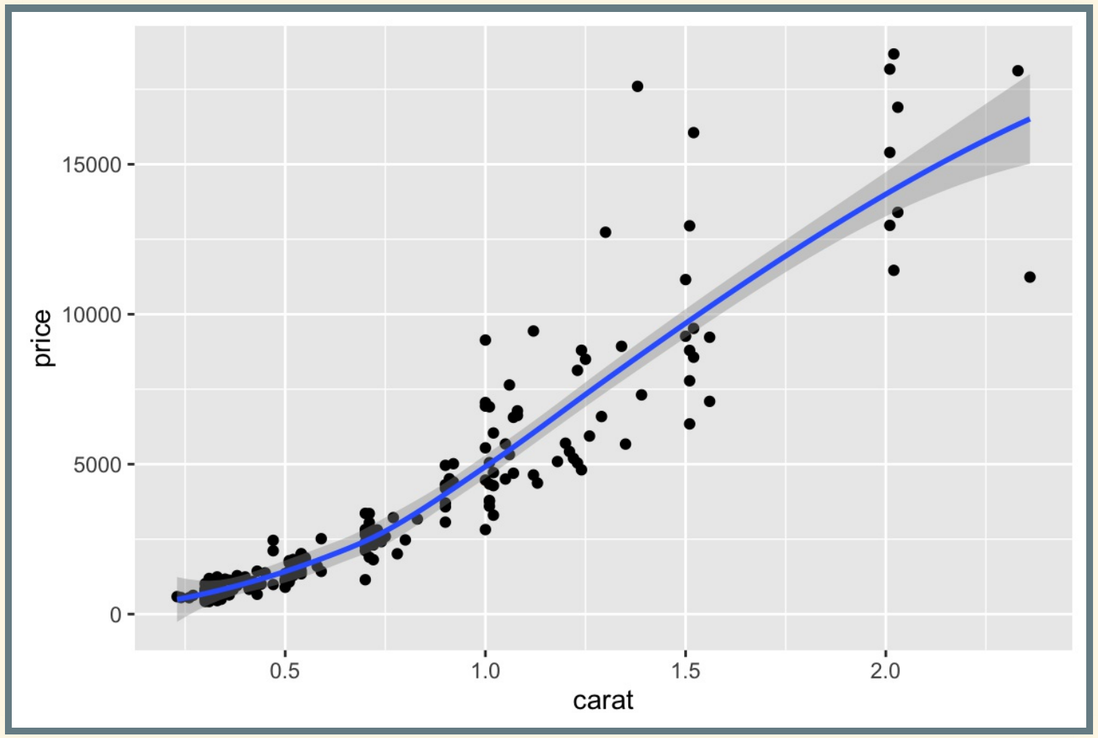
Smoothers usually help discerning patterns in the data, especially when you have many data points.

```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point() + geom_smooth()
```



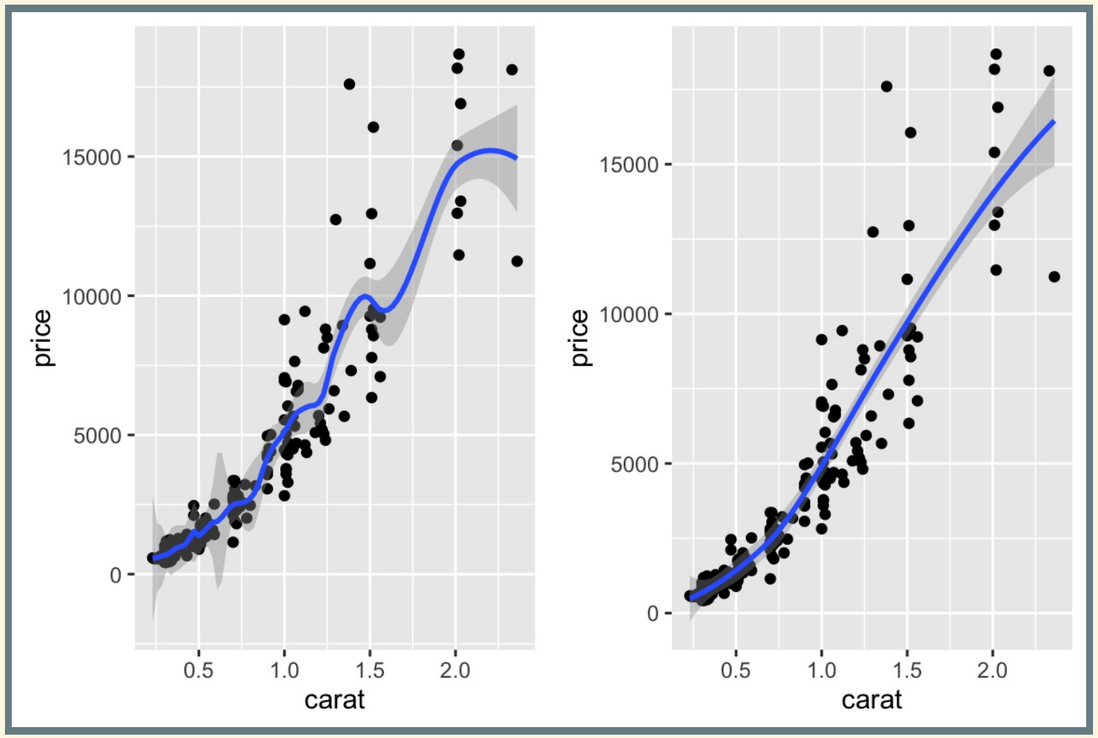
For the small subset of the diamonds dataset we have:

```
ggplot(dsmall, aes(x = carat, y = price)) +
  geom_point() + geom_smooth()
```



Changing the span argument, we can obtain more or less wiggly curve (smaller span results in more wiggliness).

```
grid.arrange(p1 + geom_point() + geom_smooth(span = 0.2),
             p1 + geom_point() + geom_smooth(span = 0.7), ncol = 2)
```



- By default `geom_smooth` when applied to datasets with a few observations (`n < 1000`) uses a `method = "loess"` for fitting a curve.
- This method uses a smooth local regression. More details about the algorithm used can be found in `?loess`.
- Loess does not work well for large datasets (it's $O(n^2)$ in memory), and so an alternative smoothing algorithm is used when n is greater than 1,000.

EXERCISE 2: THE ECONOMIST DATA

- Go back to “Lec3_ex.Rmd”
- Complete the exercise 2

SCALES

AESTHETIC MAPPING VARIABLE SCALING

- **Aesthetic mapping** (aes()) is responsible for assigning an aesthetic to a variable. It doesn't however specify how mapping should be done.
- For example, `aes(shape = x)` or `aes(color = z)` do not specify what shapes or what colors should be used. To choose colors/shapes/sizes etc. you need to **modify the corresponding scale**.

GGPLOT2 SCALES INCLUDE

- position
- color and fill
- size
- shape
- line type

- Scales are modified with a series of functions using a `scale_<aesthetic>_<type>()` naming scheme. E.g. `scale_color_discrete()`.
- Try typing `scale_<tab>()` to see a list of scale modification functions.

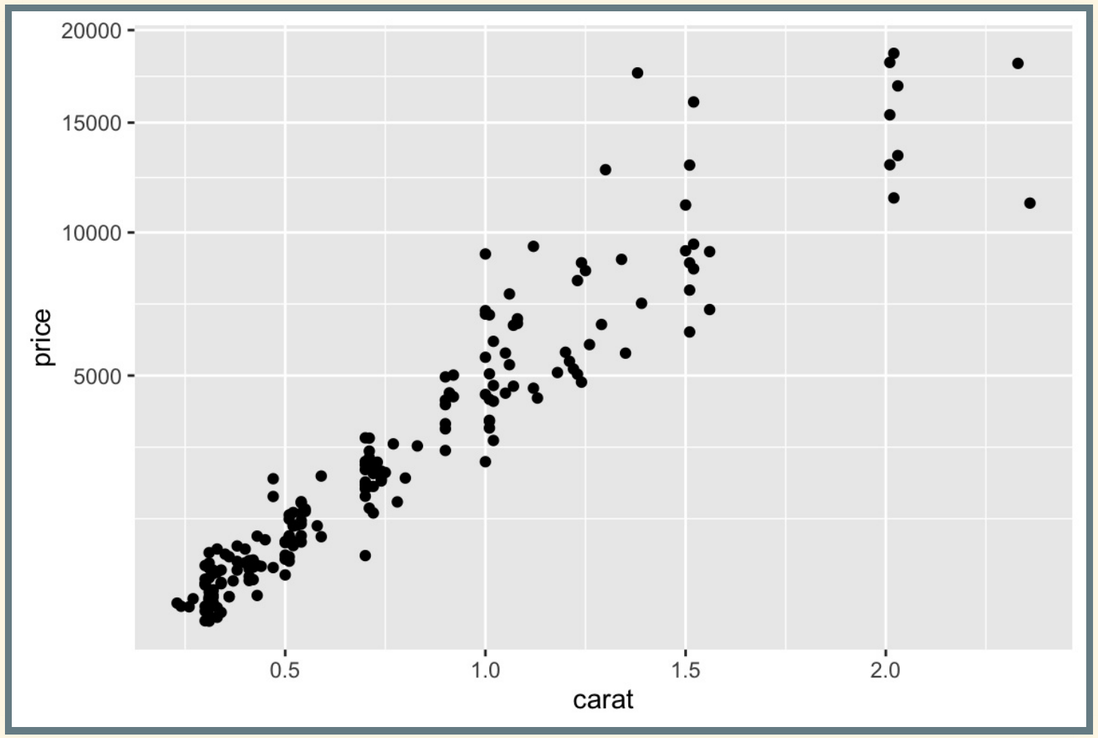
Common Scale Arguments:

- **name**: the first argument gives the axis or legend title
- **limits**: the minimum and maximum of the scale
- **breaks**: the points along the scale where labels should appear
- **labels**: the labels that appear at each break

SCALE: AXES

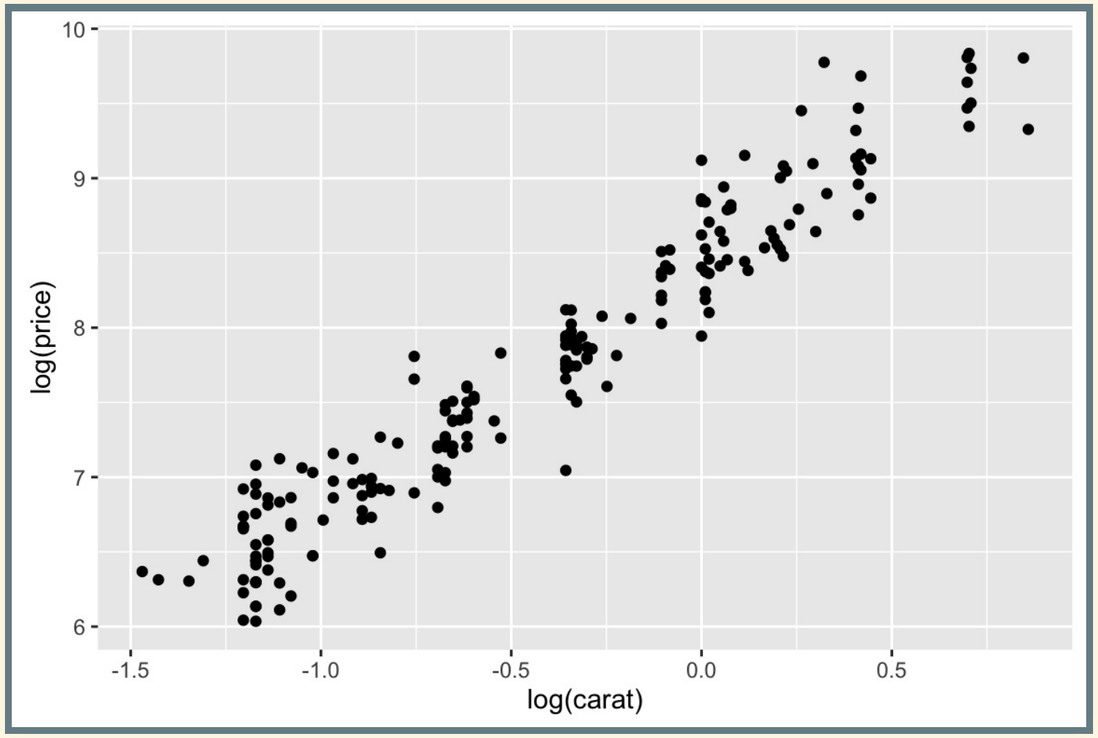
Square root transformation of the y-axis:

```
p1 <- ggplot(dsmall, aes(x = carat, y = price))
p1 + geom_point() + scale_y_sqrt()
```



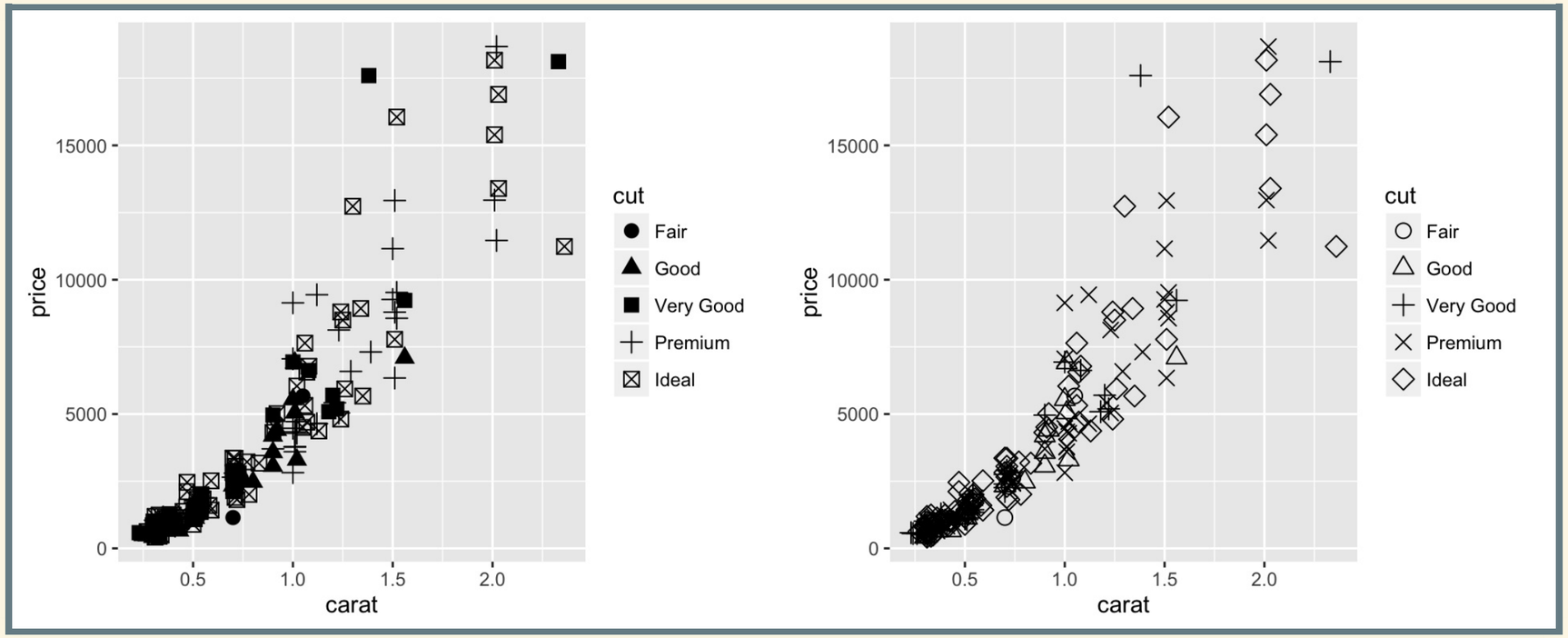
Note that the previus plot is the same as (just with different values on the axes):

```
ggplot(dsmall, aes(x = log(carat), y = log(price))) + geom_point()
```



SCALE: SHAPES

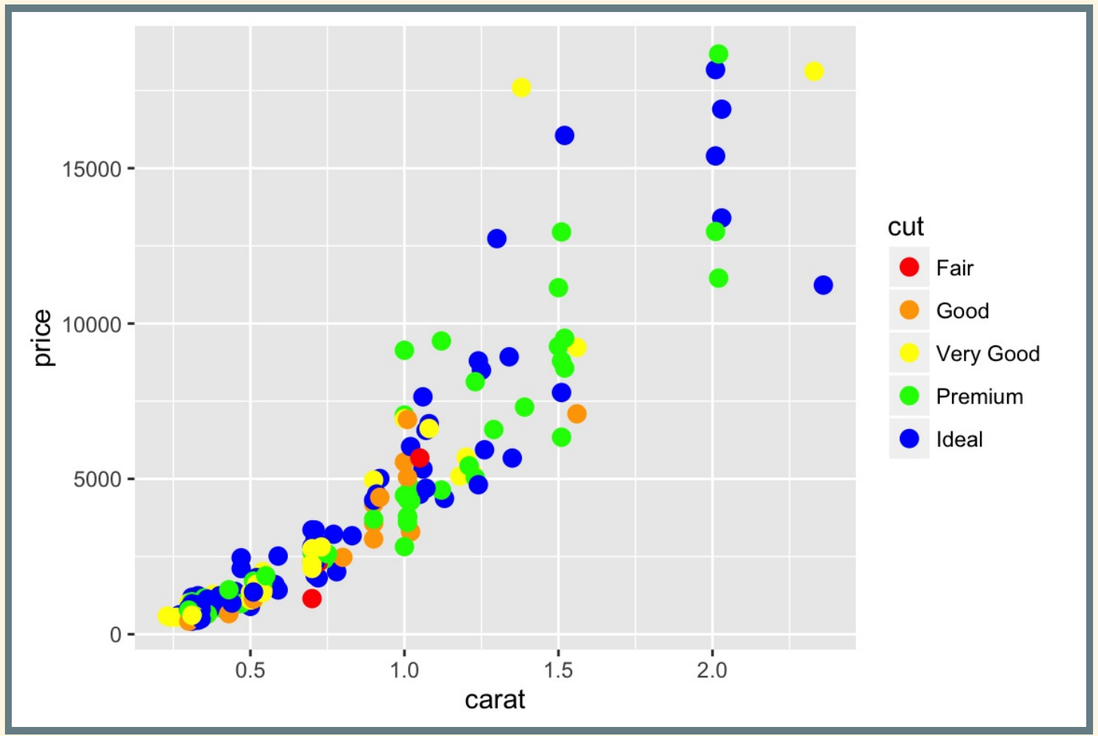
```
p11 <- p1 + geom_point(aes(shape = cut), size = 3)
p12 <- p1 + geom_point(aes(shape = cut), size = 3) +
  scale_shape_manual(values = c(1:5))
grid.arrange(p11, p12, ncol = 2)
```



SCALE: COLORS

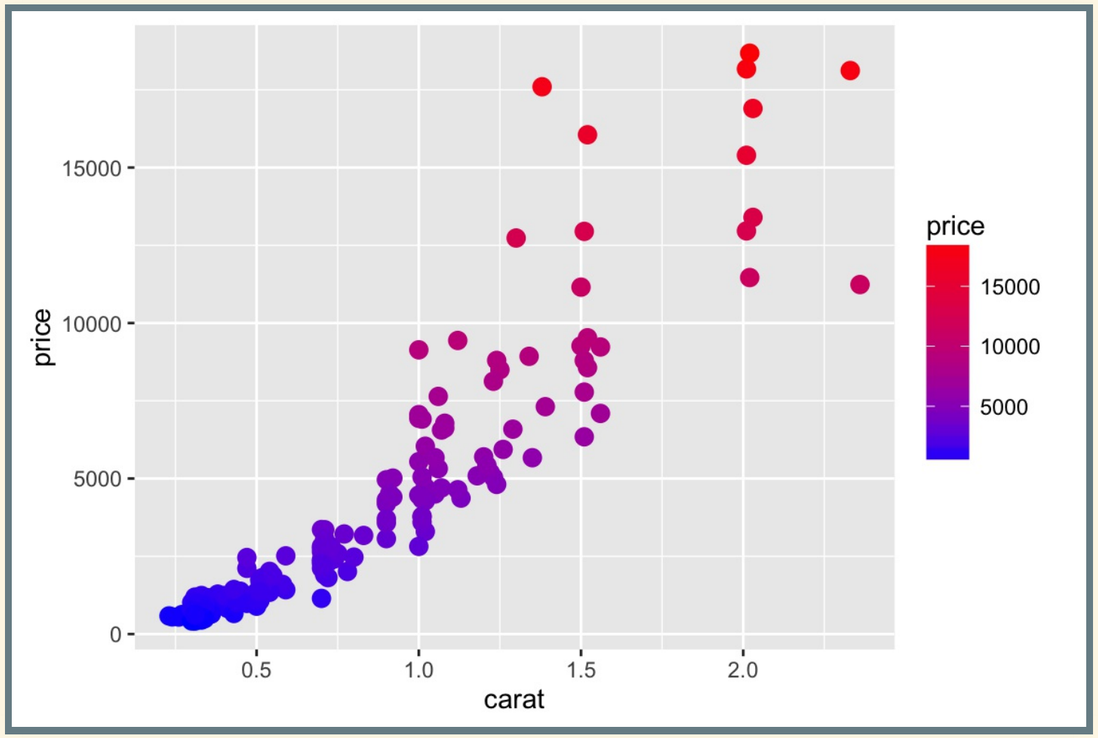
- To choose specific colors for **discrete** variables we can use `scale_color_manual`

```
p1 + geom_point(aes(color = cut), size = 3) +
  scale_color_manual(values = c("red", "orange", "yellow", "green", "blu
```



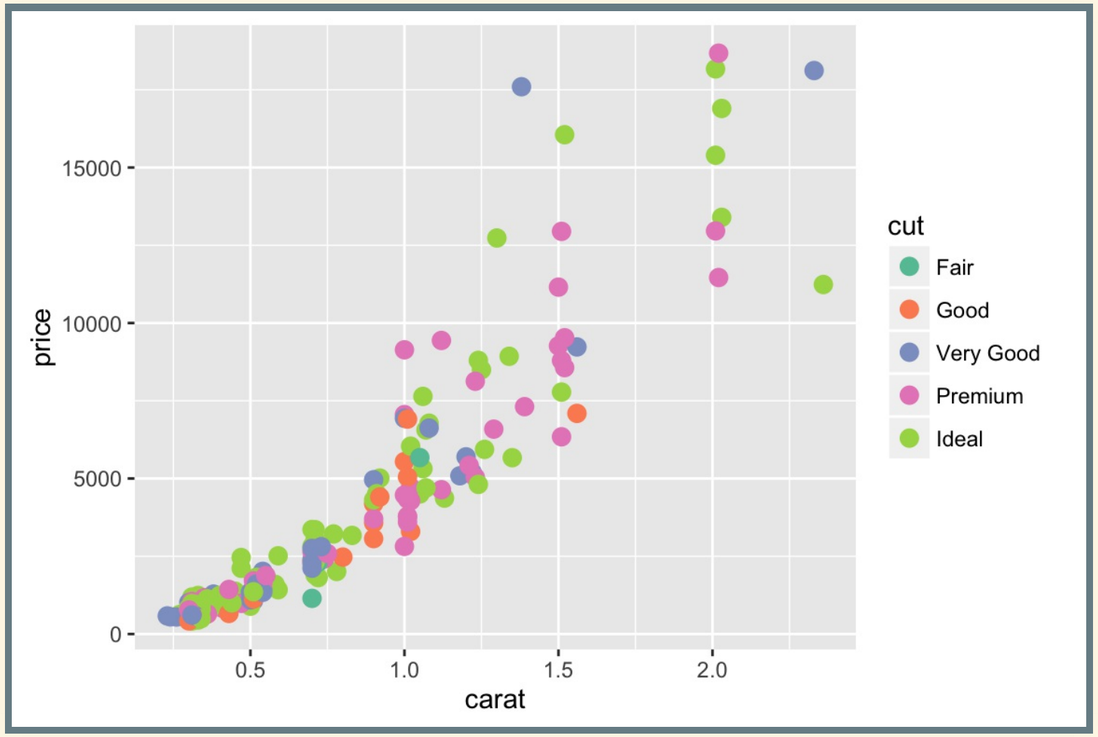
- For **continuous** variables you can also use `scale_color_gradient`, and specify the ends of the spectrum:

```
p1 + geom_point(aes(color = price), size = 3) +
  scale_color_gradient(low = "blue", high = "red")
```



- `scale_color_brewer` is a very useful function that can be used to set colors for **discrete** variables.
- It gives you a choice between many predefined and pretty **color palettes**.

```
p1 + geom_point(aes(color = cut), size = 3) +
  scale_color_brewer(palette = "Set2")
```

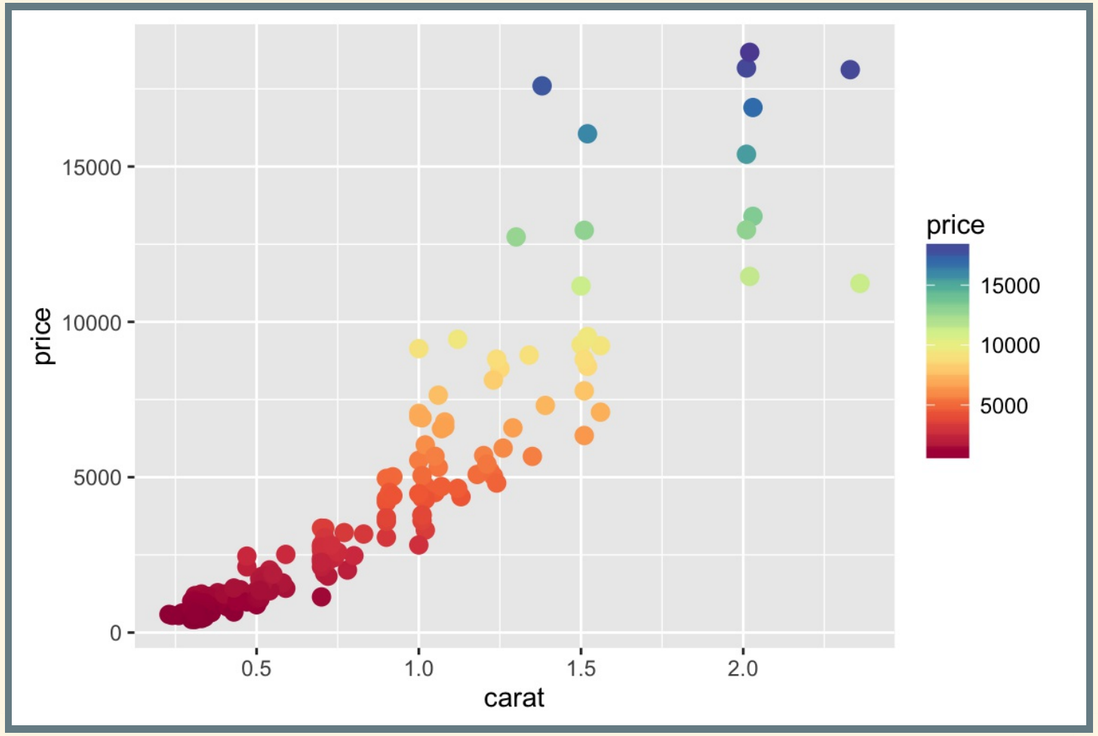


Unfortunately `scale_color_brewer` doesn't work for continuous variables:

```
# scale_color_brewer() does not work with continuous variables
# and will result in an error
p1 + geom_point(aes(shape = price), size = 3) +
  scale_color_brewer(palette = "Spectral")
# Error: A continuous variable can not be mapped to shape
```

- we can get around this issue using the `RColorBrewer` package and `scale_color_gradientn function, which **interpolates the colors** from the brewer palettes.

```
library(RColorBrewer)
p1 + geom_point(aes(color = price), size = 3) +
  scale_color_gradientn(colours = brewer.pal(name = "Spectral", n = 10))
```



… there are other unconventional schemes such as, this one based on Wes Anderson movies:

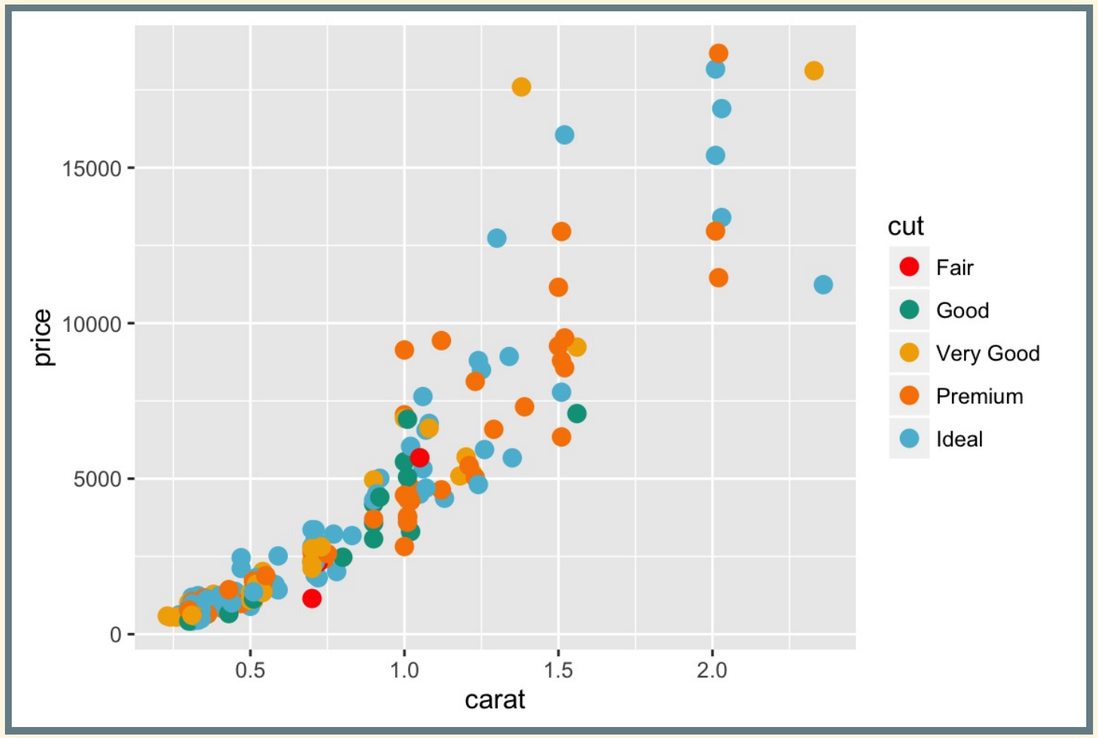
```
#install.packages("wesanderson")
library(wesanderson)
names(wes_palettes)
```

```
##  [1] "GrandBudapest"  "Moonrise1"      "Royal1"         "Moonrise2"
##  [5] "Cavalcanti"     "Royal2"         "GrandBudapest2" "Moonrise3"
##  [9] "Chevalier"      "Zissou"         "FantasticFox"   "Darjeeling"
## [13] "Rushmore"       "BottleRocket"   "Darjeeling2"
```

WES ANDERSON COLOR PALETTE:

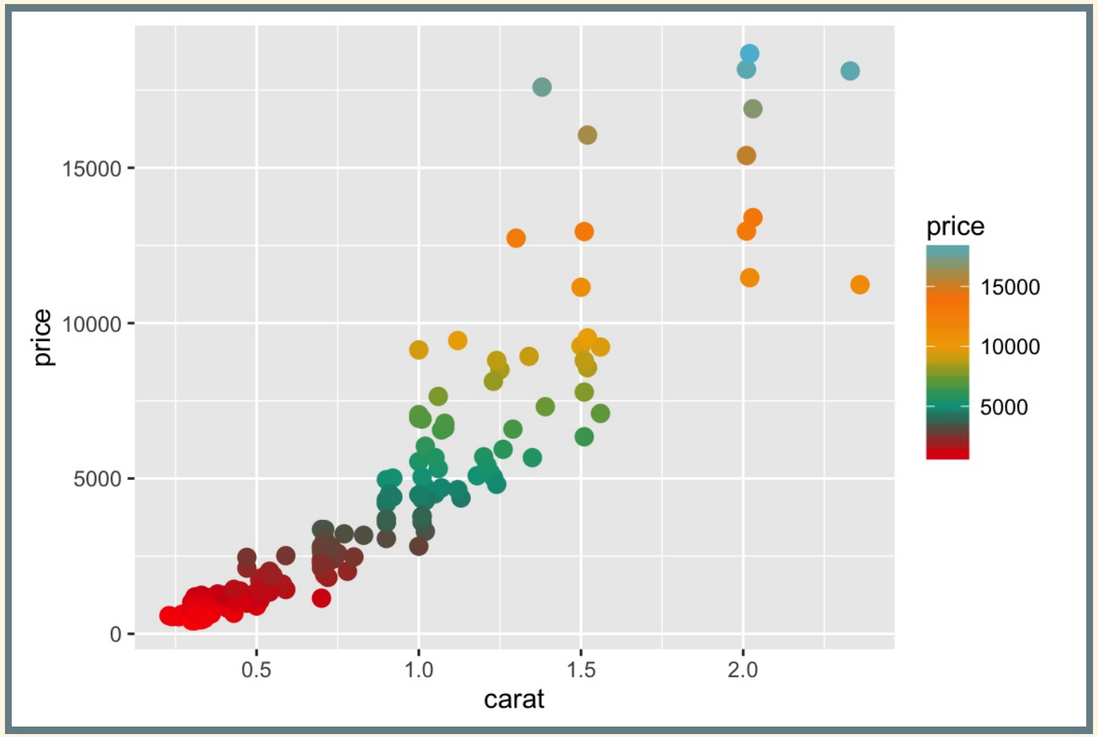
*For discrete variables

```
p1 + geom_point(aes(color = cut), size = 3) +
  scale_color_manual(values = wes_palette("Darjeeling", n = 5))
```



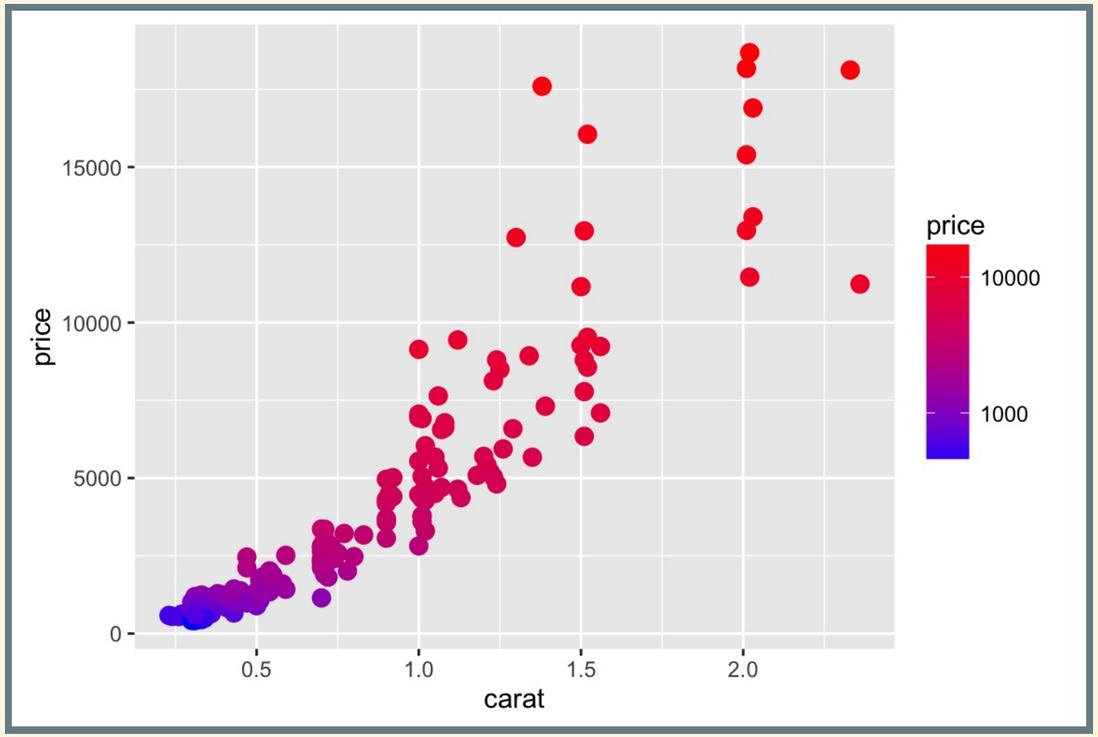
For continuous variables:

```
p1 + geom_point(aes(color = price), size = 3) +
  scale_color_gradientn(colours = wes_palette("Darjeeling", 100, type =
```



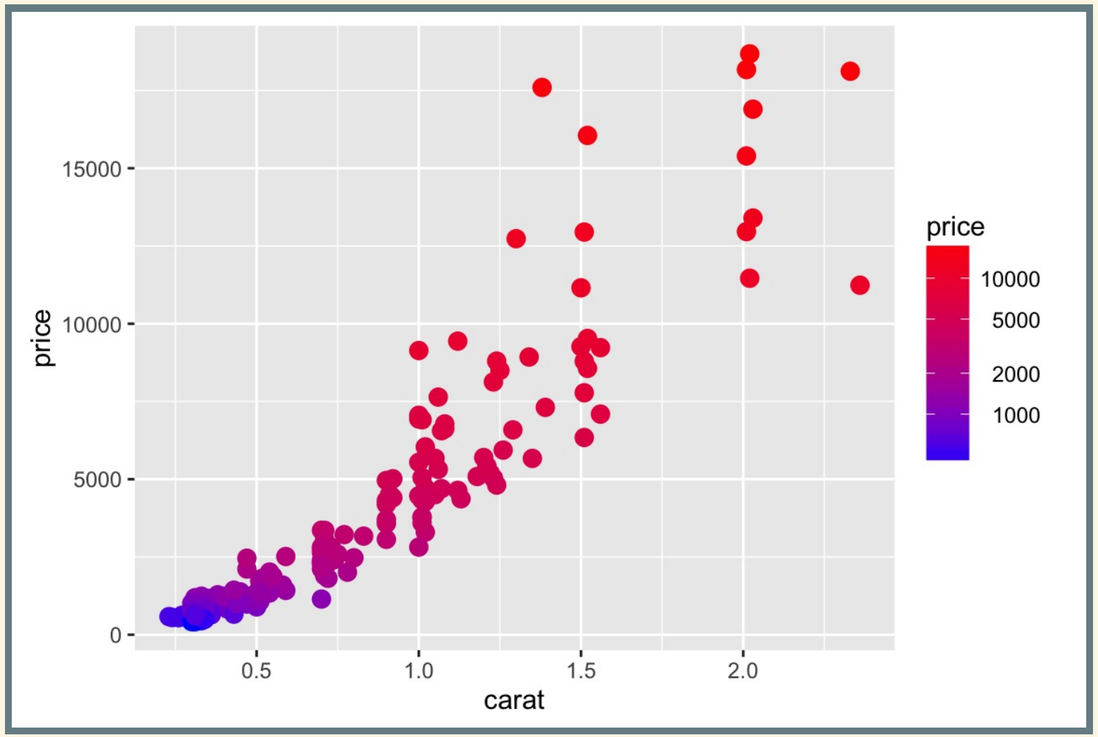
- You can also scale the values of the variable corresponding to color.

```
p1 + geom_point(aes(color = price), size = 3) +
  scale_color_gradient(low = "blue", high = "red", trans = "log10")
```



- and add your own breaks

```
p1 + geom_point(aes(color = price), size = 3) +
  scale_color_gradient(low = "blue", high = "red", trans = "log10",
                       breaks = c(1000, 2000, 5000, 10000),
                       labels = c("  1000", "  2000", "  5000", "10000")
```



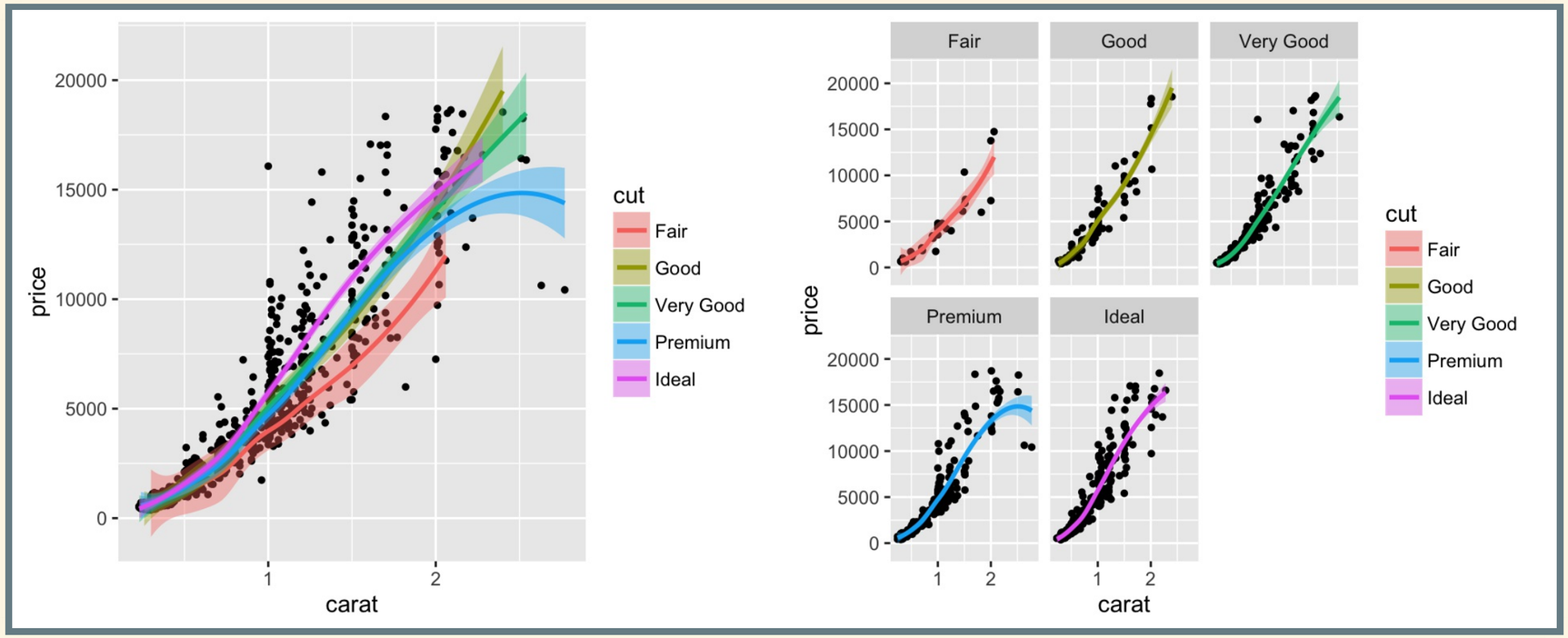
EXERCISE 3: THE ECONOMIST DATA

- Go to “Lec3_ex.Rmd”
- Complete Exercise 3

FACETING

- **Facetting** allows you to split up your data by one or more variables and plot the subsets of data together.

```
dsmall <- diamonds[sample(nrow(diamonds), 1000), ]
p0 <- ggplot(data = dsmall, aes(x = carat, y = price)) +
  geom_point(size = 1) +
  geom_smooth(aes(colour = cut, fill = cut))
p1 <- p0 + facet_wrap(~ cut)
grid.arrange(p0, p1, ncol = 2)
```

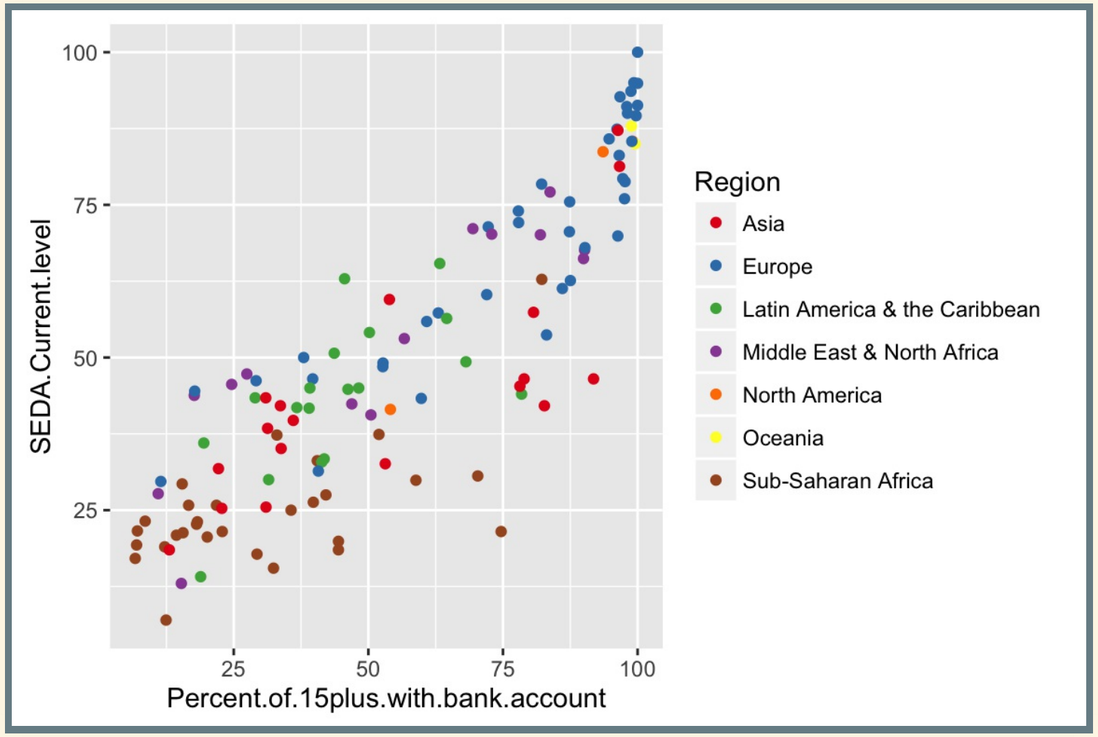


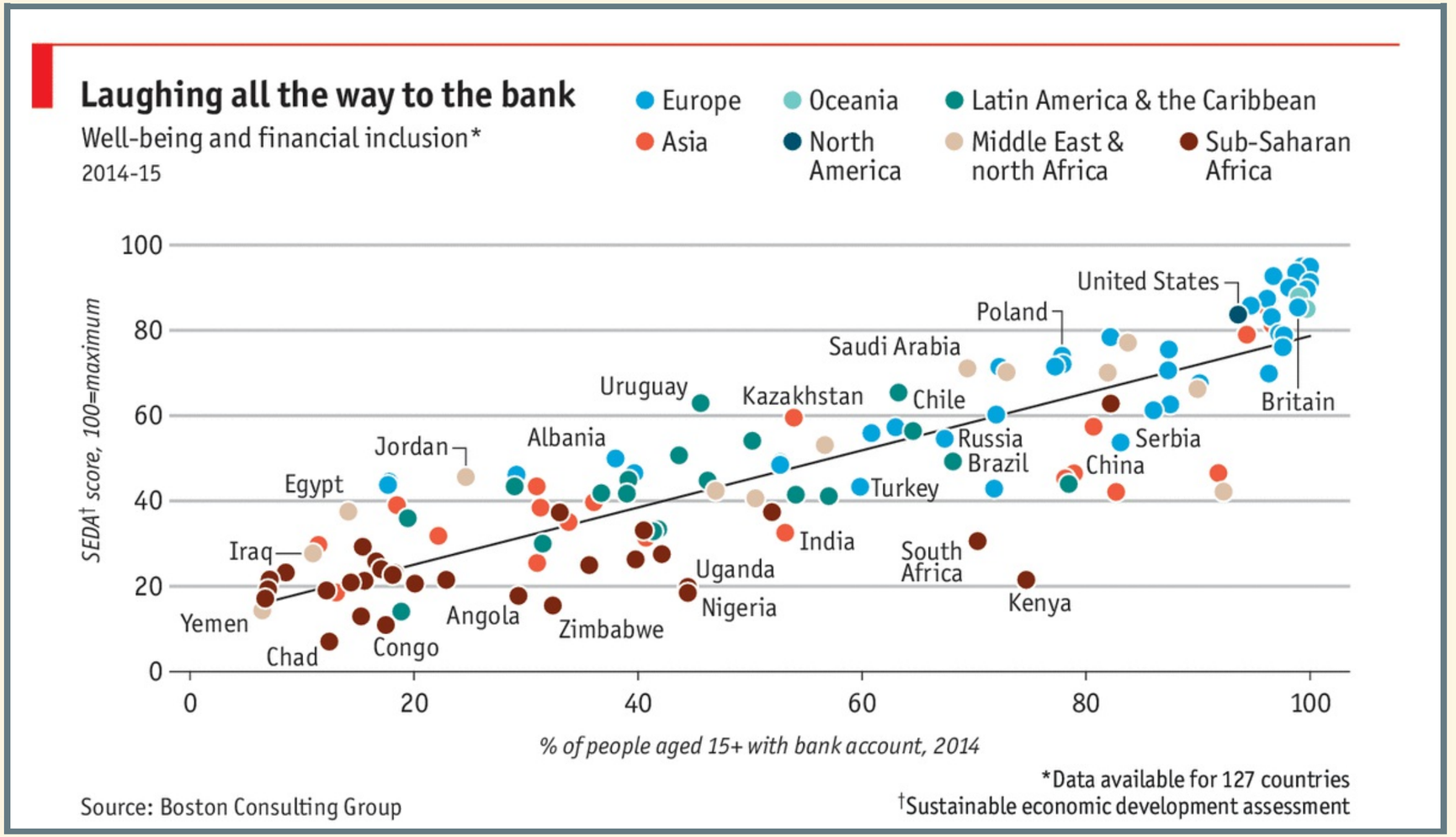
EXCERCISE 4: THE ECONOMIST DATA

- Complete Exercise 4 in “Lec3_ex.Rmd”

FINISH THE ECONOMIST PLOT

```
pEc <- ggplot(dat, aes(Percent.of.15plus.with.bank.account, SEDA.Current
(pEc <- pEc + geom_point(aes(color = Region)) + scale_color_brewer(palet
```




Laughing all the way to the bank
Well-being and financial inclusion*
2014-15
Europe
Asia
Oceania
North America
Latin America & the Caribbean
Middle East & north Africa
Sub-Saharan Africa
SEDA† score, 100=maximum
% of people aged 15+ with bank account, 2014
United States
Poland
Saudi Arabia
Uruguay
Kazakhstan
Chile
Britain
Jordan
Albania
Russia
Serbia
Brazil
China
Egypt
Turkey
Iraq
India
South Africa
Uganda
Nigeria
Kenya
Yemen
Angola
Zimbabwe
Chad
Congo
0
20
40
60
80
100
*Data available for 127 countries
†Sustainable economic development assessment
Source: Boston Consulting Group

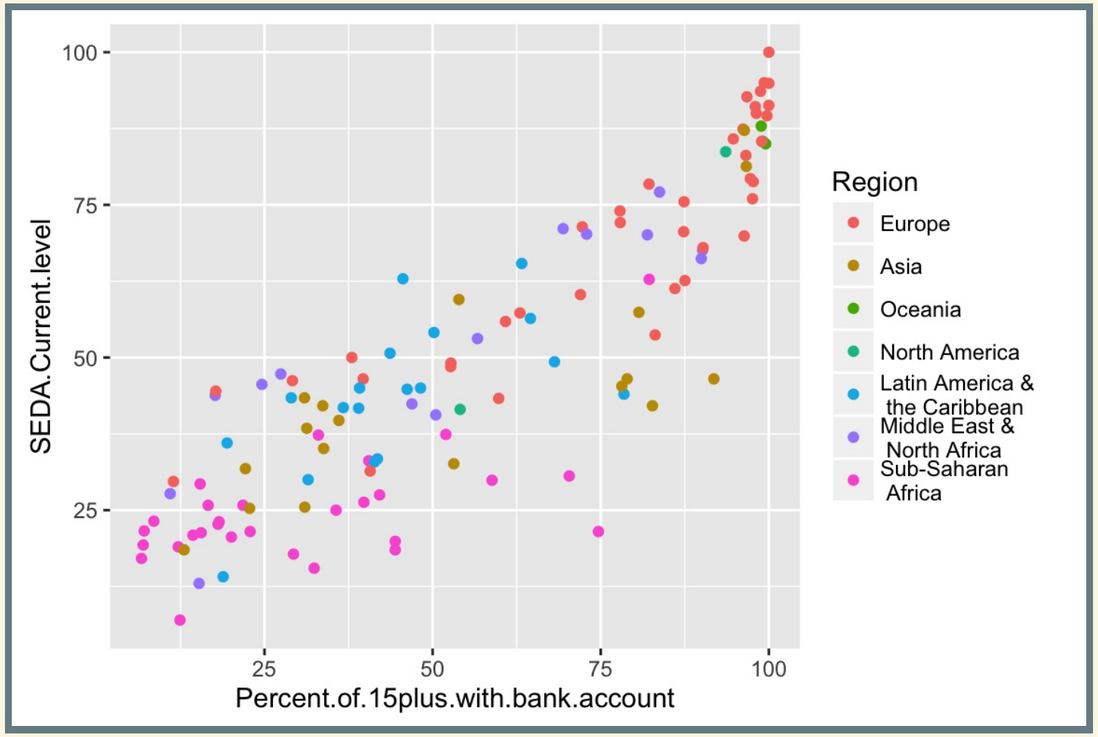
To complete the graph we need to:

- add a trend line
- change the axis labels
- change the order of the Region labels
- change the coloring of the points
- label selected points
- change color legend’s position
- adjust the axes ratio
- fix the tick marks
- match the plot’s theme with the Economist theme
- add notes

CHANGE ORDER OF THE REGIONS

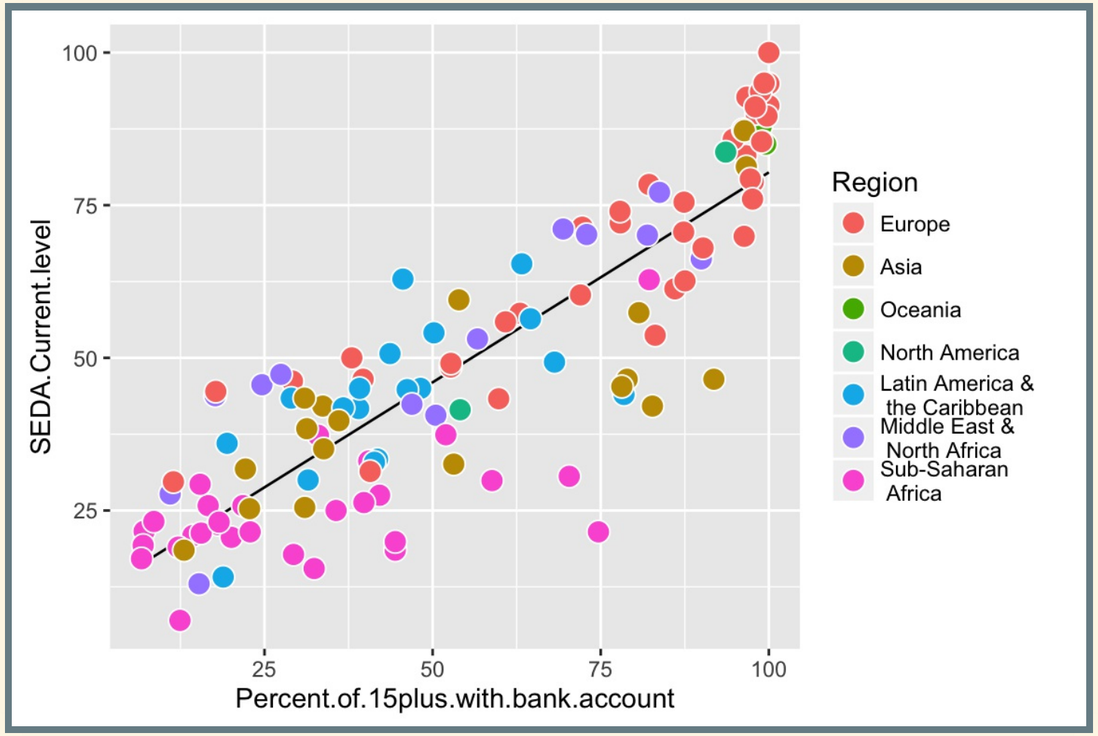
```
dat$Region <- as.character(dat$Region)
dat$Region <- factor(dat$Region,
                     levels = c("Europe", "Asia", "Oceania",
                                "North America",
                                "Latin America & the Caribbean",
                                "Middle East & North Africa",
                                "Sub-Saharan Africa"),
                     labels = c("Europe", "Asia", "Oceania",
                                "North America",
                                "Latin America & \n the Caribbean",
                                "Middle East & \n North Africa",
                                "Sub-Saharan \n Africa"))
```

```
pEc <- ggplot(dat, aes(Percent.of.15plus.with.bank.account, SEDA.Current
pEc + geom_point(aes(color = Region))
```



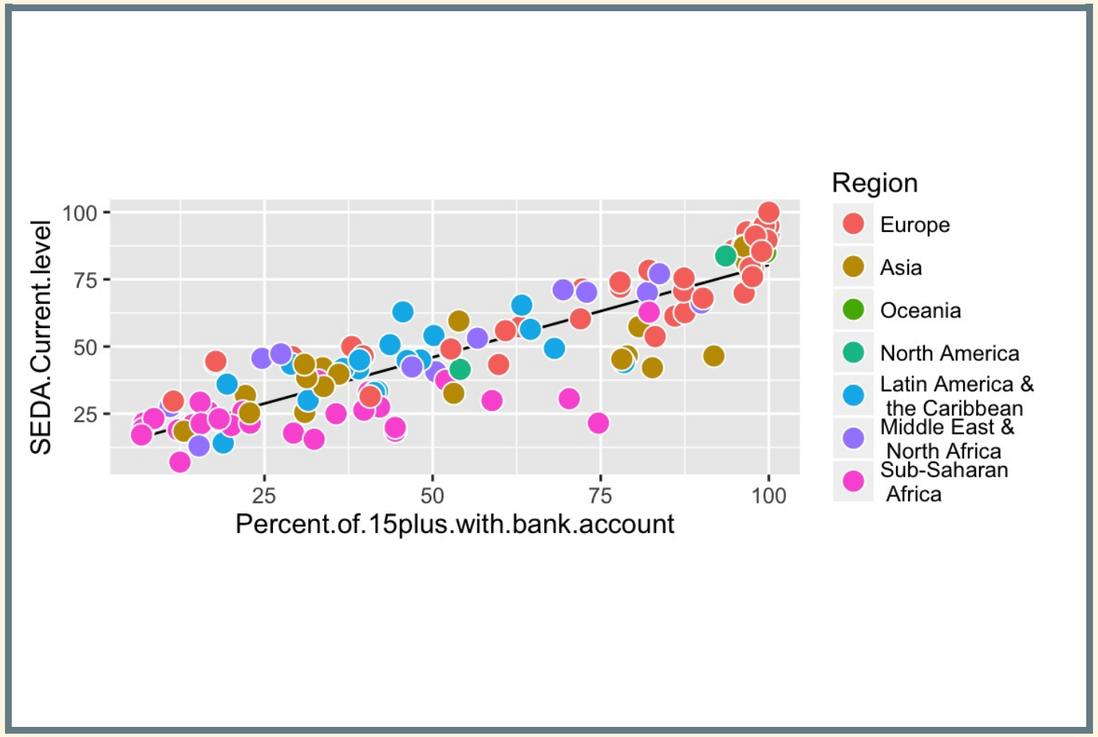
ADD THE LINEAR TREND

```
pEc <- pEc + geom_smooth(method = "lm", se = FALSE, col = "black", size
(pEc <- pEc + geom_point(aes(fill = Region), color = "white", shape = 21
```



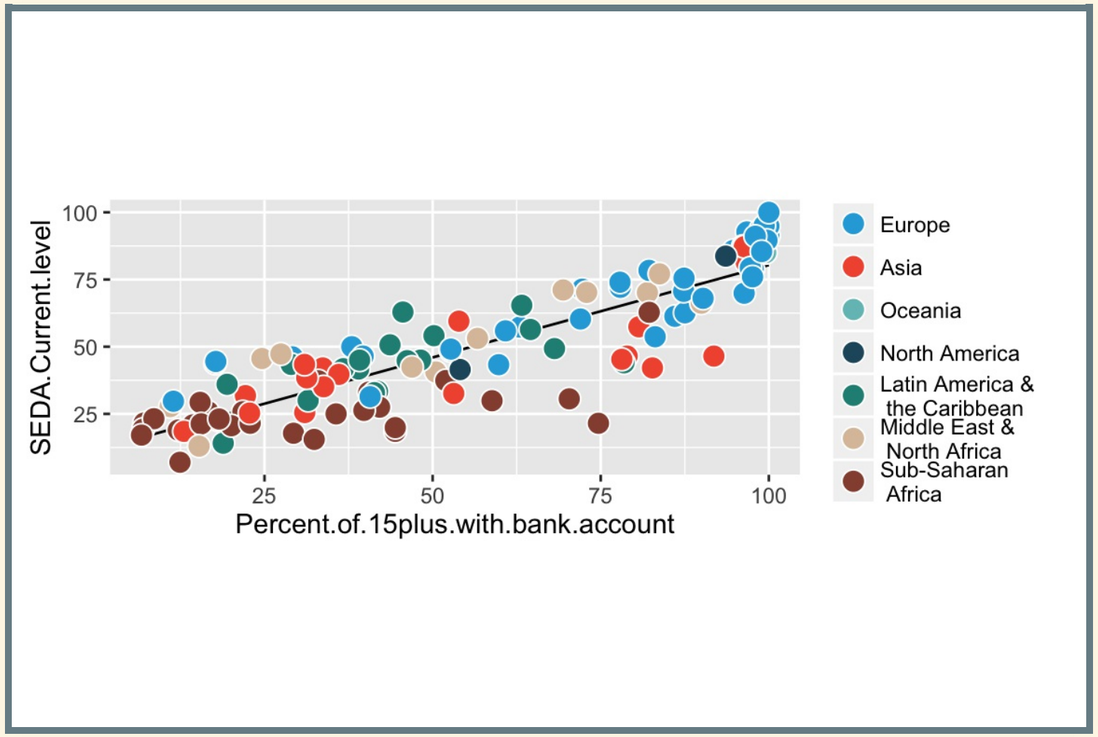
CHANGE THE AXES RATIO.

```
(pEc <- pEc + coord_fixed(ratio = 0.4))
```



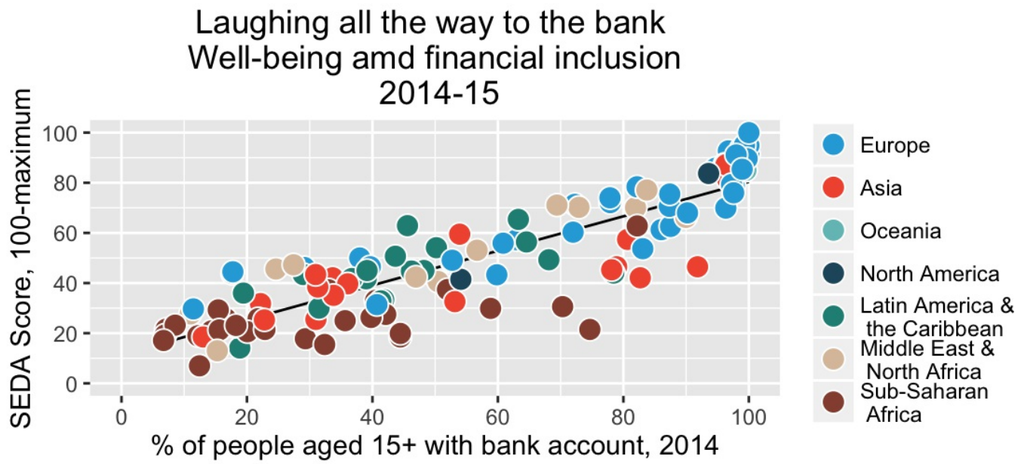
CHANGE THE COLOR SCHEME

```
colors <-  c("#28AADC","#F2583F", "#76C0C1","#24576D",
             "#248E84","#DCC3AA", "#96503F")
(pEc <- pEc + scale_fill_manual(name = "",
                                values = colors))
```



ADD A TITLE AND FORMAT THE AXES

```
(pEc <- pEc +
  scale_x_continuous(name = "% of people aged 15+ with bank account, 201
                     limits = c(0, 100),
                     breaks = seq(0, 100, by = 20)) +
  scale_y_continuous(name = "SEDA Score, 100-maximum",
                     limits = c(0, 100),
                     breaks = seq(0, 100, by = 20)) +
  ggtitle("Laughing all the way to the bank \n Well-being amd financial
```

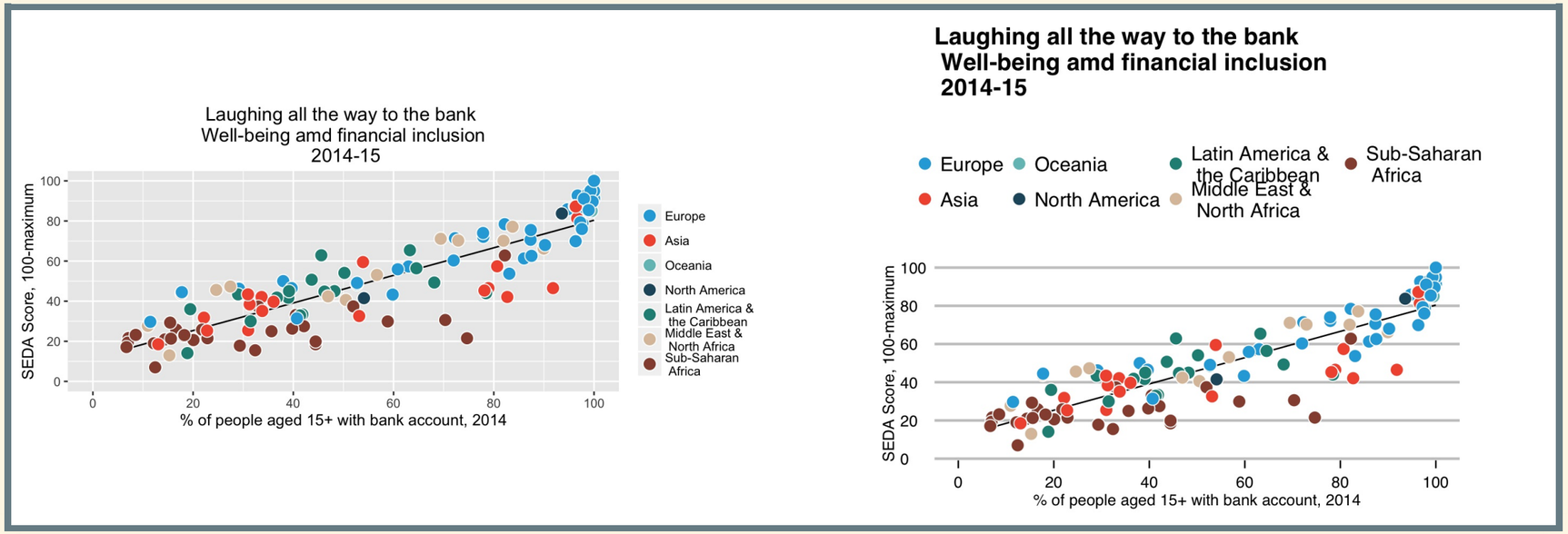


CHANGE THE BACKGROUND AND THEME

You can check out the `ggthemes` package which implement the themes that make your plots look like they came from:

- Base graphics
- Tableau
- Excel
- Stata
- Economist
- Wall Street Journal
- Edward Tufte
- Nate Silver's Fivethirtyeight
- etc.

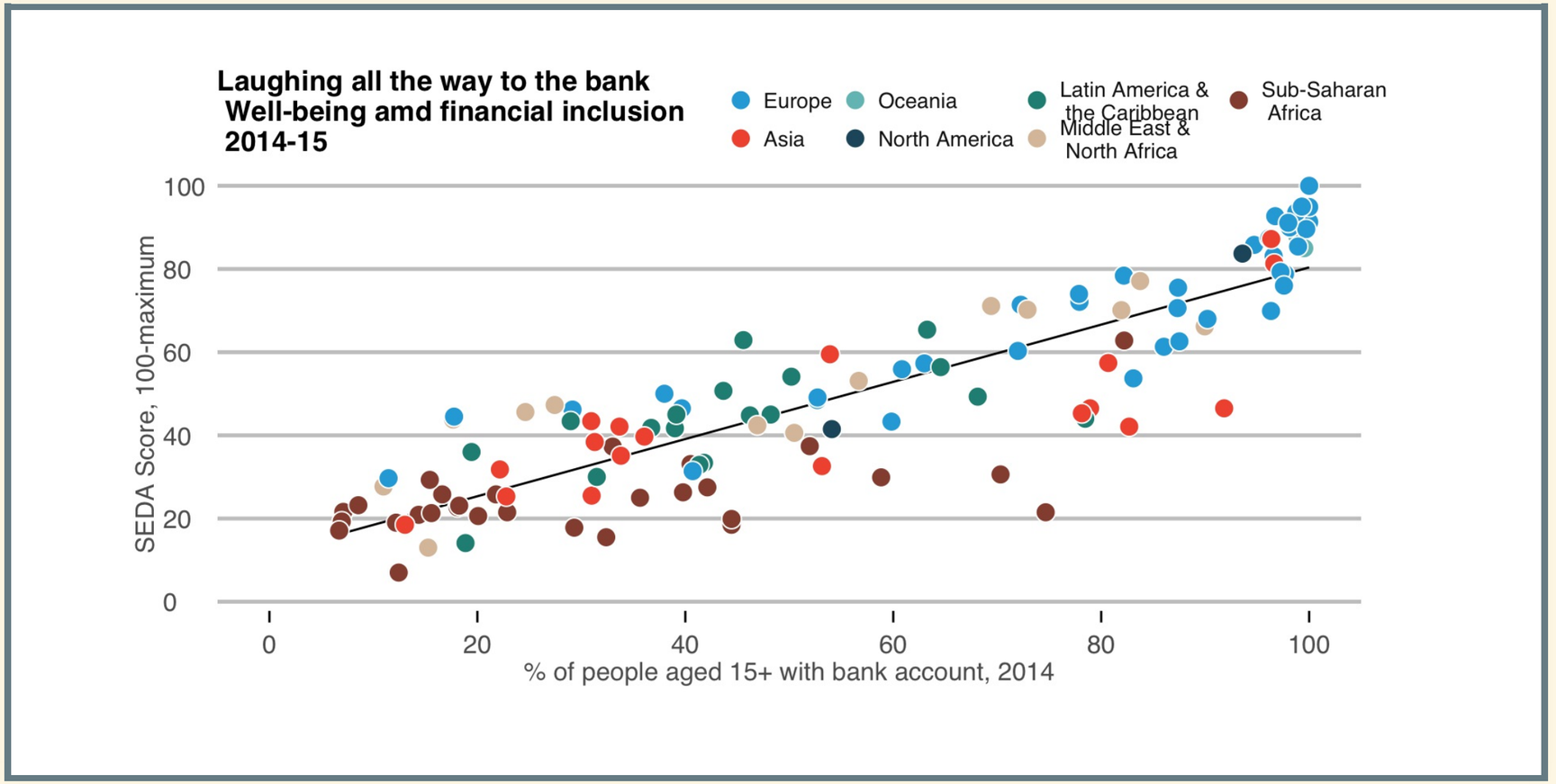
```r
# install.pcakages("ggthemes")
library(ggthemes)
pEc0 <- pEc
pEc <- pEc + theme_economist_white(gray_bg=FALSE)
grid.arrange(pEc0, pEc, ncol = 2)
```



FORMAT THE LEGEND

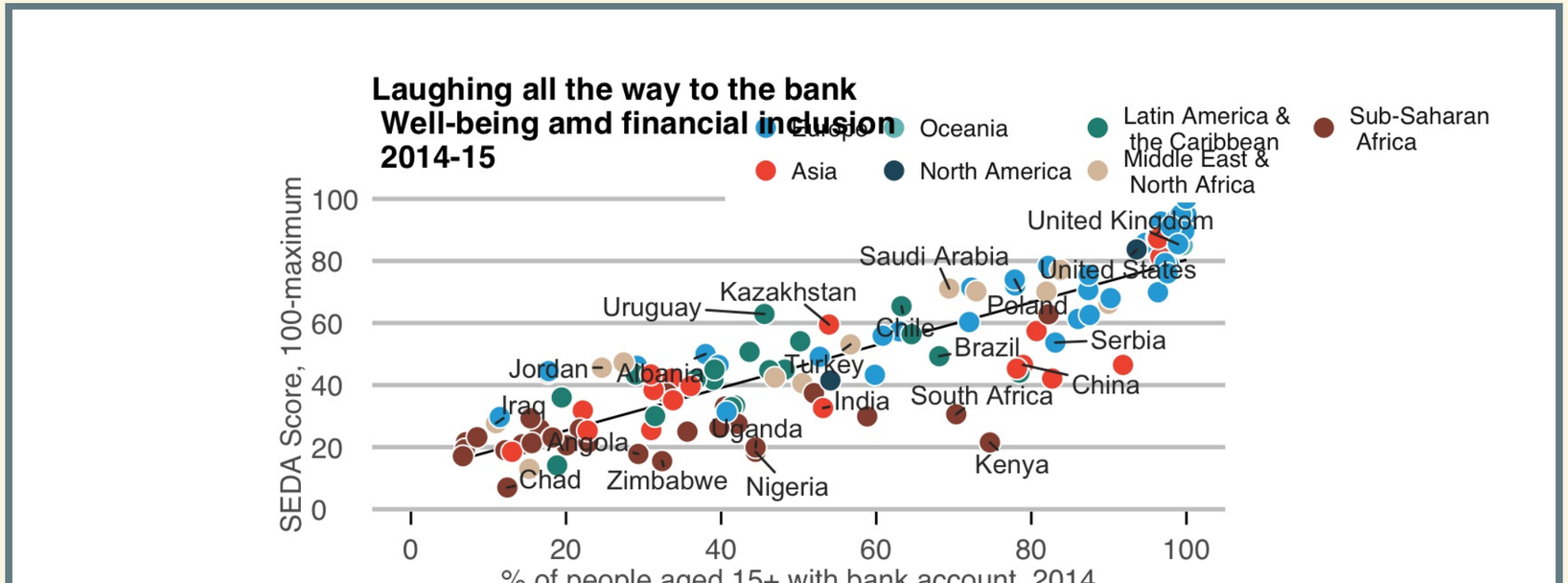
```
pEc <- pEc + coord_fixed(0.4) +
   theme(text = element_text(color = "grey37", size = 12),
         legend.position = c(0.45, 1.1), # position the legend in the upp
         legend.direction = "horizontal",
         legend.justification = 0.1, # anchor point for legend.position.
         legend.text = element_text(size = 10, color = "gray10"),
         plot.title = element_text(size = rel(1.1), color = "black"),
         plot.margin = unit(c(1, 1.5, 1.5, 0.5), "cm")) +
  guides(fill = guide_legend(ncol = 4, byrow = FALSE))
```

pEc


Laughing all the way to the bank
Well-being amd financial inclusion
2014-15
Europe
Asia
Oceania
North America
Latin America & the Caribbean
Middle East & North Africa
Sub-Saharan Africa
SEDA Score, 100-maximum
100
80
60
40
20
0
0
20
40
60
80
100
% of people aged 15+ with bank account, 2014

ADD POINT LABELS

```
library(ggrepel)
pointsToLabel <- c("Yemen", "Iraq", "Egypt", "Jordan", "Chad", "Congo",
                   "Angola", "Albania", "Zimbabwe", "Uganda", "Nigeria",
                   "Uruguay", "Kazakhstan", "India", "Turkey", "South Af
                   "Kenya", "Russia", "Brazil", "Chile", "Saudi Arabia",
                   "Poland", "China", "Serbia", "United States", "Unitec
(pEcText <-  pEc + geom_text_repel(aes(label = Country), color = "gray20
                                   data = subset(dat, Country %in% pointsTol
                                   force = 20))
```

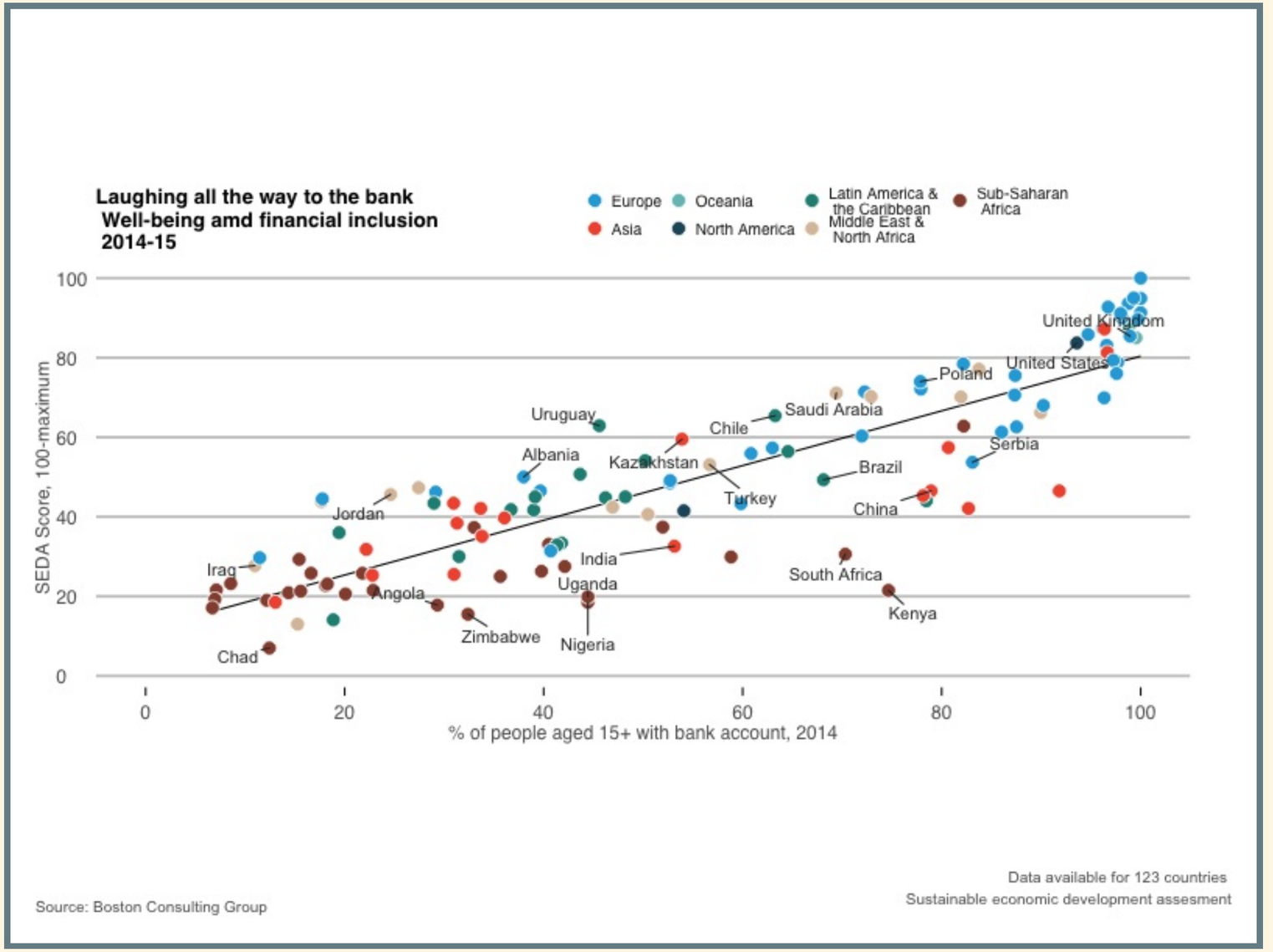


ADD NOTES TO THE BOTTOM AND SAVE THE PLOT

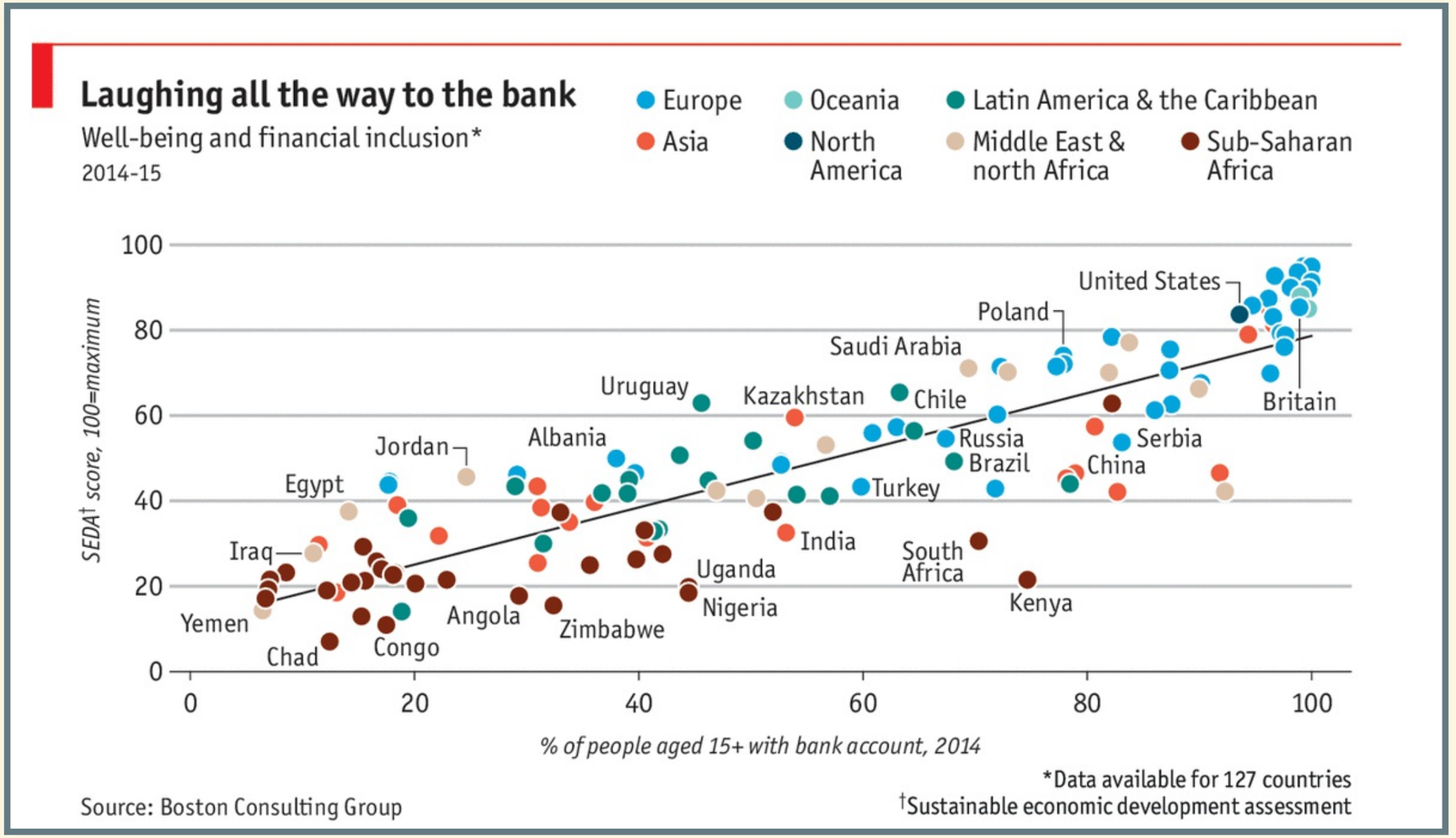
Use “grid.text()” to add notes

```
library(grid)
png(file = "./Lecture3_files/econScatter.png", width = 800, height = 600
pEcText
grid.text("Source: Boston Consulting Group",
          x = .02, y = .04, just = "left",
          draw = TRUE, gp=gpar(fontsize=10, col="grey37"))
grid.text("Data available for 123 countries \n Sustainable economic deve
          x = 0.98, y = .06, just = "right",
           draw = TRUE, gp=gpar(fontsize=10, col="grey37"))
dev.off()
```

```
## quartz_off_screen
##                 2
```


Laughing all the way to the bank
Well-being amd financial inclusion
2014-15
Europe
Oceania
Latin America & the Caribbean
Sub-Saharan Africa
Asia
North America
Middle East & North Africa
SEDA Score, 100-maximum
% of people aged 15+ with bank account, 2014
United Kingdom
United States
Poland
Saudi Arabia
Uruguay
Chile
Serbia
Albania
Kazakhstan
Brazil
Turkey
China
Jordan
India
South Africa
Iraq
Uganda
Angola
Kenya
Zimbabwe
Nigeria
Chad
Source: Boston Consulting Group
Data available for 123 countries
Sustainable economic development assesment

Similar to the original:



ADDITIONAL RESOURCES

- Hadley Wickham's R for Data Science: Chapter 3. Data Visualization
- Hadley Wickham's ggplot2: Elegant Graphics for Data Analysis(http://ggplot2.org/book/)
- Wiki: https://github.com/hadley/ggplot2/wiki
- `plotly` github

1. https://www.bcgperspectives.com/content/articles/growth-globalization-private-sector-opportunity-improve-well-being-2016-economic-development-assessment/↩