LECTURE 3: GENERATING 2D GRAPHICS WITH R

OCTOBER 6, 2016

WHAT WE WILL LEARN TODAY

BASE GRAPHICS AND GGPLOT2 FOR 2D PLOTTING

- How to generate:
  - line plots, scatter plots
  - histograms, bar plots, boxplots
  - prediction/trend lines or smoothers
- How to modify the aesthetics settings:
  - coloring scheme
  - shapes
- How to use themes to automatically set the plot style
- How to use facetting to display information for different subsets of data.

SOURCES

Some of the material presented today is based on the following material:

- workshop notes by Data Science Services at Harvard IQSS
- and Chapter 2 from Hadley Wickham's book on ggplot2.

INSTALL AND LOAD PACKAGES

```
# Clear the workspace
rm(list = ls())
# List of package needed for this workshop
reqpkg <- c("ggplot2", "ggrepel", "ggthemes", "grid", "gridExtra",
            "RColorBrewer")
# Check if the packages are installed:
inpkg <- installed.packages()[, "Package"] #installed packages
neededpkg <- reqpkg[!reqpkg %in% inpkg]
if(length(neededpkg) > 0){
  stop(paste("\n Need to install the following package:", neededpkg))
}
# If any errors then type the following in the console
#lapply(neededpkg, install.packages, dependencies = TRUE)
```

WHAT IS GGPLOT2?

`ggplot2` is a plotting system for R, based on the grammar of graphics. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics. [1]

ADVANTAGES OF GGPLOT2:

- plots are defined at **a high level of abstraction**,
- plots are broken down into **modules/layers**,
- a great flexibility when **customizing** your plot,
- good **documentation**,
- a large user base - **easy access to help**.

WEAKNESSES OF GGPLOT2 / WHAT THE PACKAGE SHOULD NOT BE USED FOR:

- 3D graphics: use `rgl` or `ggplot2 + plotly` instead,
- graph/network plots with nodes and edges: use `igraph`
- interactive graphics: use `ggvis`, `plotly`

WHAT IS THE GRAMMAR OF GRAPHICS?

It is a concept coined by Leland Wilkinson in *2005*.

The basic idea: a plot is defined by independent building blocks, which combined create just about any kind of visualization you want.

THE BUILDING BLOCKS OF A GRAPH INCLUDE:

- data
- aesthetic mapping
- geometric objects
- statistical transformations
- scales
- coordinate system
- positioning adjustments
- faceting

GGPLOT() FUNCTION

- The `ggplot()` is used to initialize the basic graph structure.
- It **cannot produce the plot** by itself.
- Instead, we need to **add extra building blocks** it.
- **The basic idea** is that you **specify different parts** of the plot, and *add them together* using the + operator.

THE STRUCTURE OF GGPLOT OBJECT

```
ggplot(data = <default data set>,
       aes(x = <default x axis variable>,
           y = <default y axis variable>,
           ... <other default aesthetic mappings>),
       ... <other plot defaults>) +

  geom_<geom type>(aes(size = <size variable for this geom>,
                       ... <other aesthetic mappings>),
                   data = <data for this point geom>,
                   stat = <statistic string or function>,
                   position = <position string or function>,
                   color = <"fixed color specification">,
                  ... <other arguments, possibly passed to the _stat_ function) +

  scale_<aesthetic>_<type>(name = <"scale label">,
                           breaks = <where to put tick marks>,
                           labels = <labels for tick marks>,
                           ... <other options for the scale>) +

  theme(plot.background = element_rect(fill = "gray"),
        ... <other theme elements>)
```

GGPLOT2 VS BASE GRAPHICS

GGPLOT2 COMPARED TO BASE GRAPHICS IS:

- more verbose for **simple / out of the box** graphics
- less verbose for **complex / custom** graphics
- uses a different system for **adding plot elements** instead of calling new functions.
- more details on why to use `ggplot2` over base plot can be found in this blog.

EXAMPLE 1: HISTORY OF UNEMPLOYMENT

Economics Dataset is built in `ggplot2` package

```
library(ggplot2)
data("economics")
str(economics)
```

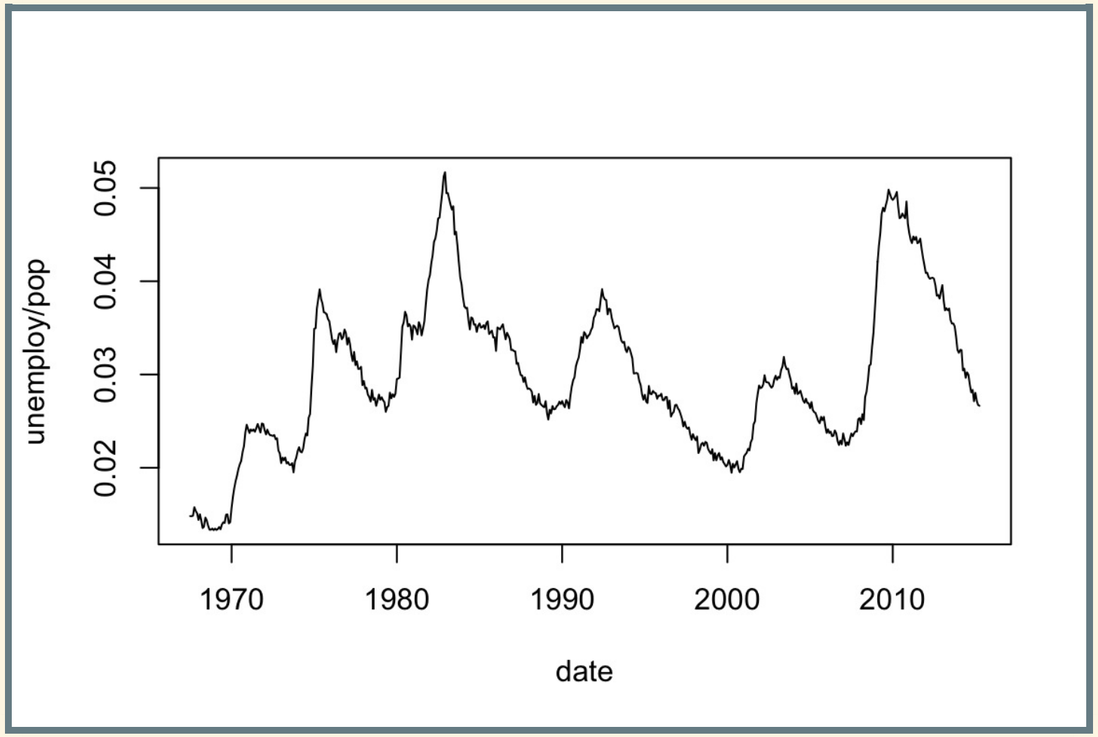
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    574 obs. of  6 variables
##  $ date    : Date, format: "1967-07-01" "1967-08-01" ...
##  $ pce     : num  507 510 516 513 518 ...
##  $ pop     : int  198712 198911 199113 199311 199498 199657 199808 19
##  $ psavert : num  12.5 12.5 11.7 12.5 12.5 12.1 11.7 12.2 11.6 12.2 .
##  $ uempmed : num  4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
##  $ unemploy: int  2944 2945 2958 3143 3066 3018 2878 3001 2877 2709 .
```

```
head(economics)
```

```
##         date   pce    pop psavert uempmed unemploy
## 1 1967-07-01 507.4 198712    12.5     4.5     2944
## 2 1967-08-01 510.5 198911    12.5     4.7     2945
## 3 1967-09-01 516.3 199113    11.7     4.6     2958
## 4 1967-10-01 512.9 199311    12.5     4.9     3143
```

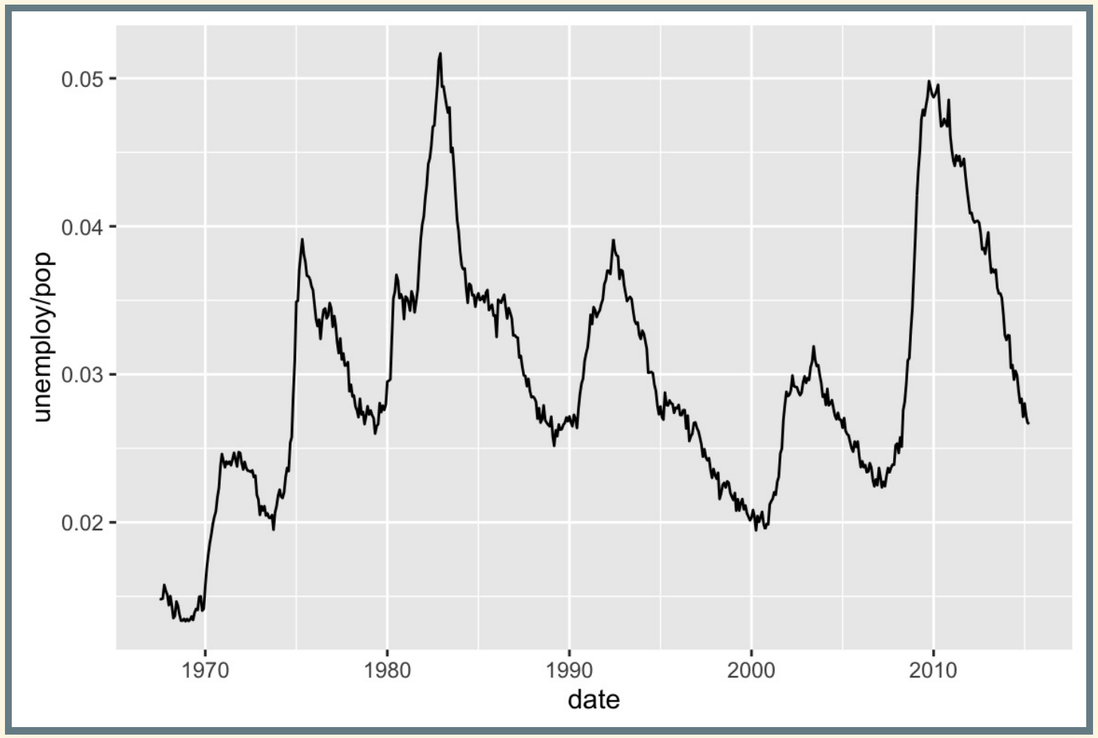
WITH PLOT() FROM BASE GRAPHICS

```
plot(unemploy/pop ~ date, data = economics,  type = "l")
```



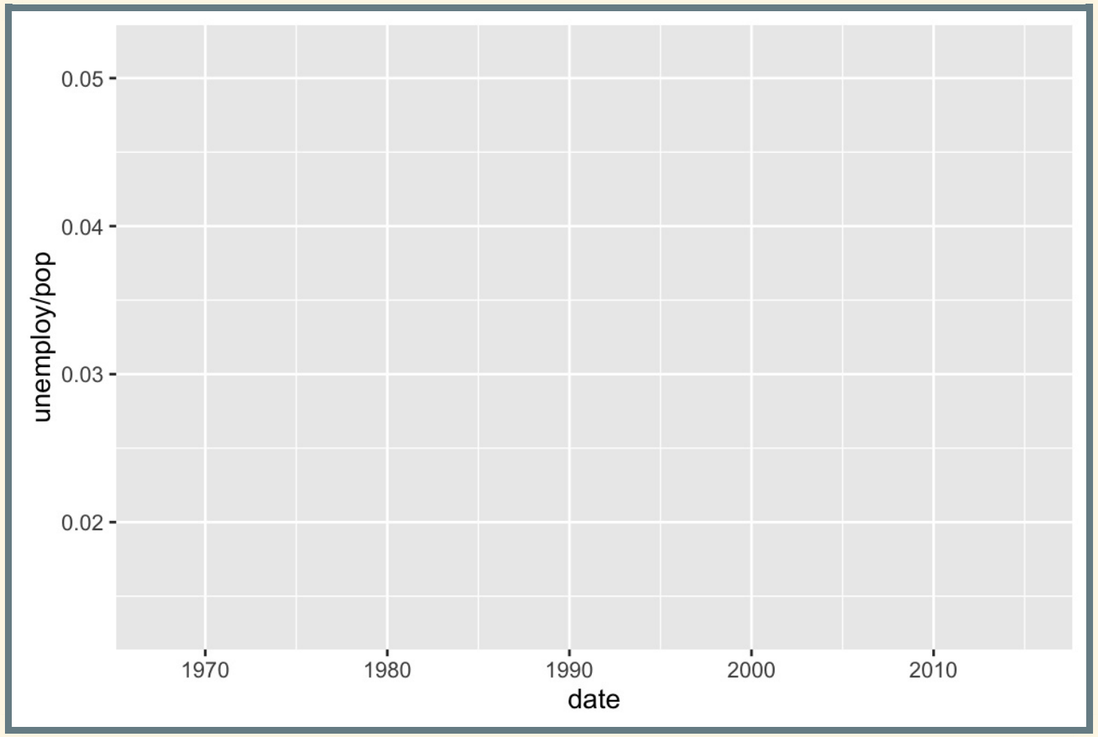
WITH GGPLOT() FROM GGPLOT2 PACKAGE

```
library(ggplot2)
ggplot(data = economics, aes(x = date, y = unemploy/pop)) + geom_line()
```



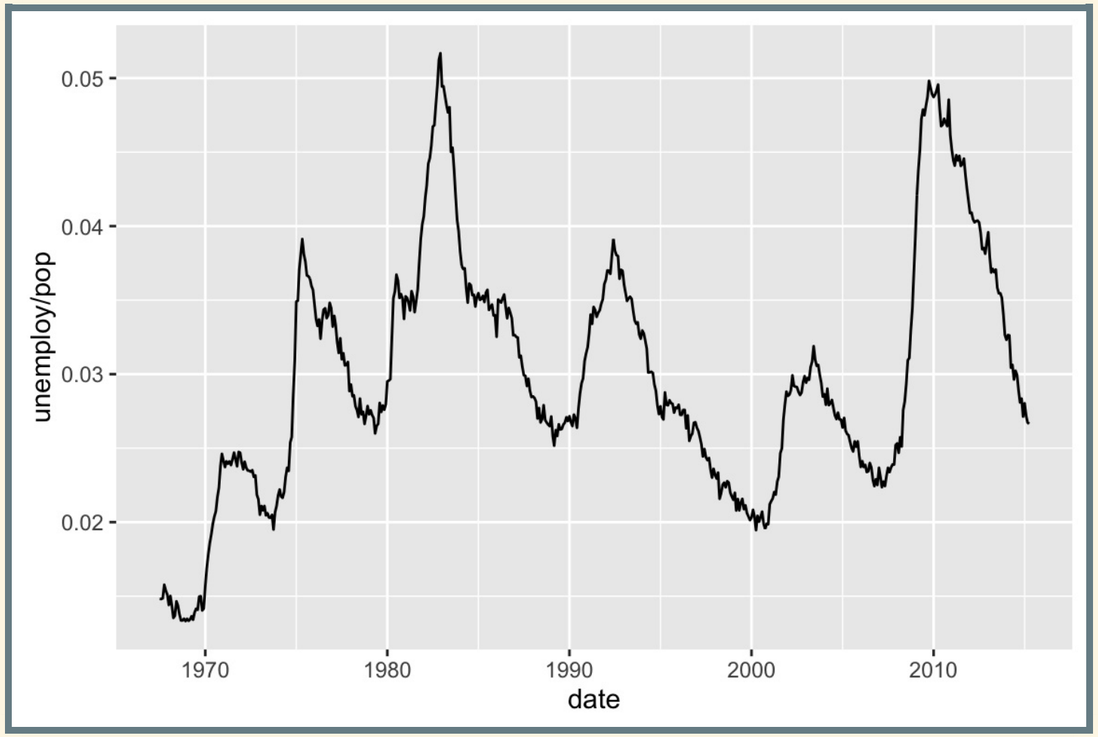
GGPLOT() BY ITSELF DOES NOT PLOT THE DATA

```
ggplot(data = economics, aes(x = date, y = unemploy/pop))
```



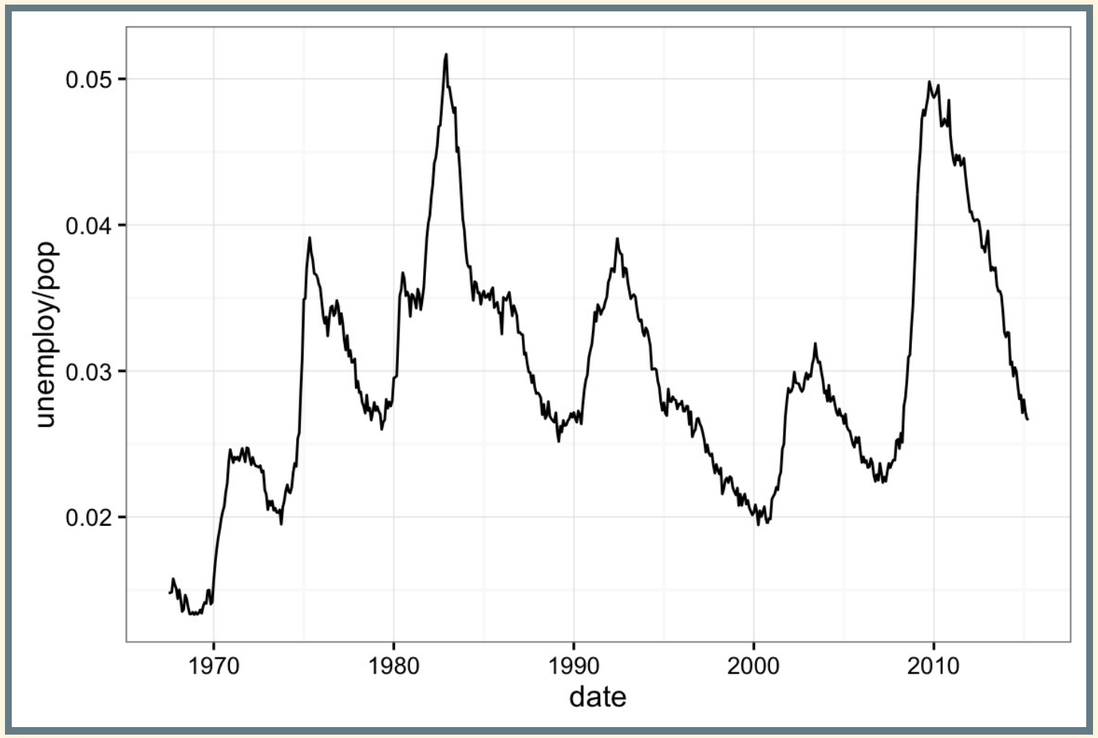
YOU NEED TO ADD THE LINES LAYER

```
ggplot(data = economics, aes(x = date, y = unemploy/pop)) + geom_line()
```



CHANGE THE BACKGROUND COLOR TO WHITE

```
ggplot(data = economics, aes(x = date, y = unemploy/pop)) +
  geom_line() + theme_bw()
```



WHAT IF WE WANT TO COMPARE THE TREND FROM YEAR 2009 TO 2014?

Add two variables one for year and one for day of the year:

```
economics$month <-months(economics$date)
economics$year <- format(economics$date, format="%Y")
head(economics)
```

```
##         date   pce    pop psavert uempmed unemploy     month year
## 1 1967-07-01 507.4 198712    12.5     4.5     2944      July 1967
## 2 1967-08-01 510.5 198911    12.5     4.7     2945    August 1967
## 3 1967-09-01 516.3 199113    11.7     4.6     2958 September 1967
## 4 1967-10-01 512.9 199311    12.5     4.9     3143   October 1967
## 5 1967-11-01 518.1 199498    12.5     4.7     3066  November 1967
## 6 1967-12-01 525.8 199657    12.1     4.8     3018  December 1967
```

```r
cat("Data type of economics$month:", class(economics$month), "\n")
```

```
## Data type of economics$month: character
```

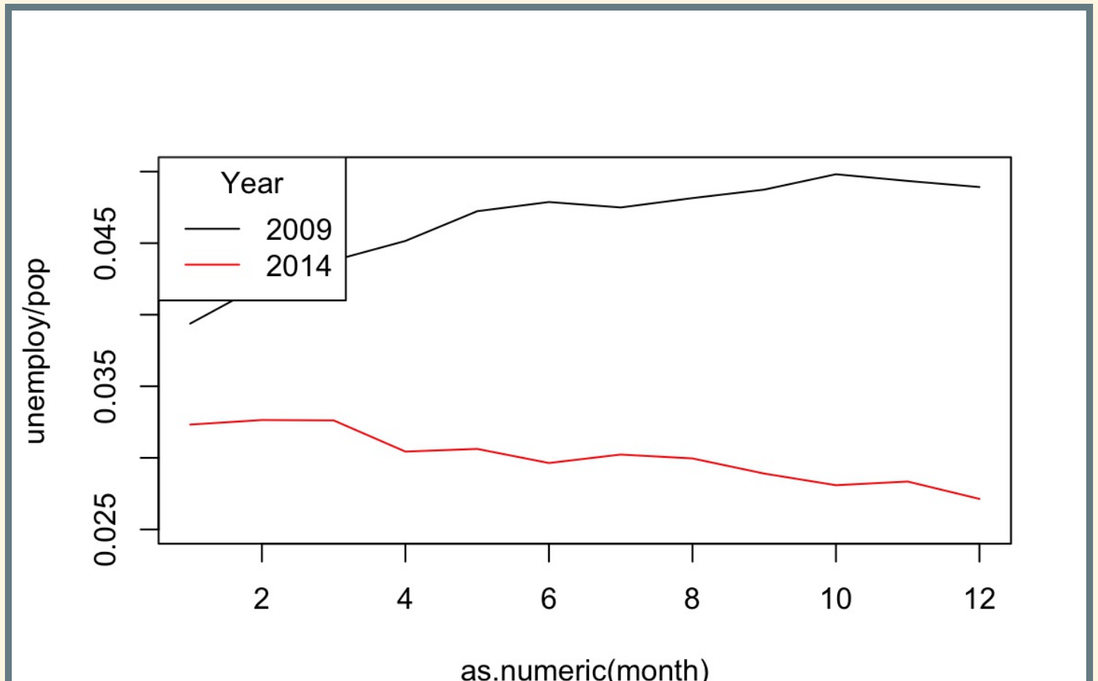
```r
# Convert the character vector to a ordered factor vector:
economics$month <- factor(economics$month, levels = month.name)
head(economics$month)
```

```
## [1] July      August    September October   November  December
## 12 Levels: January February March April May June July August ... Dece
```

```r
# further to numeric values for plotting
#economics$month <- as.numeric(economics$month)
#head(economics$month)
```

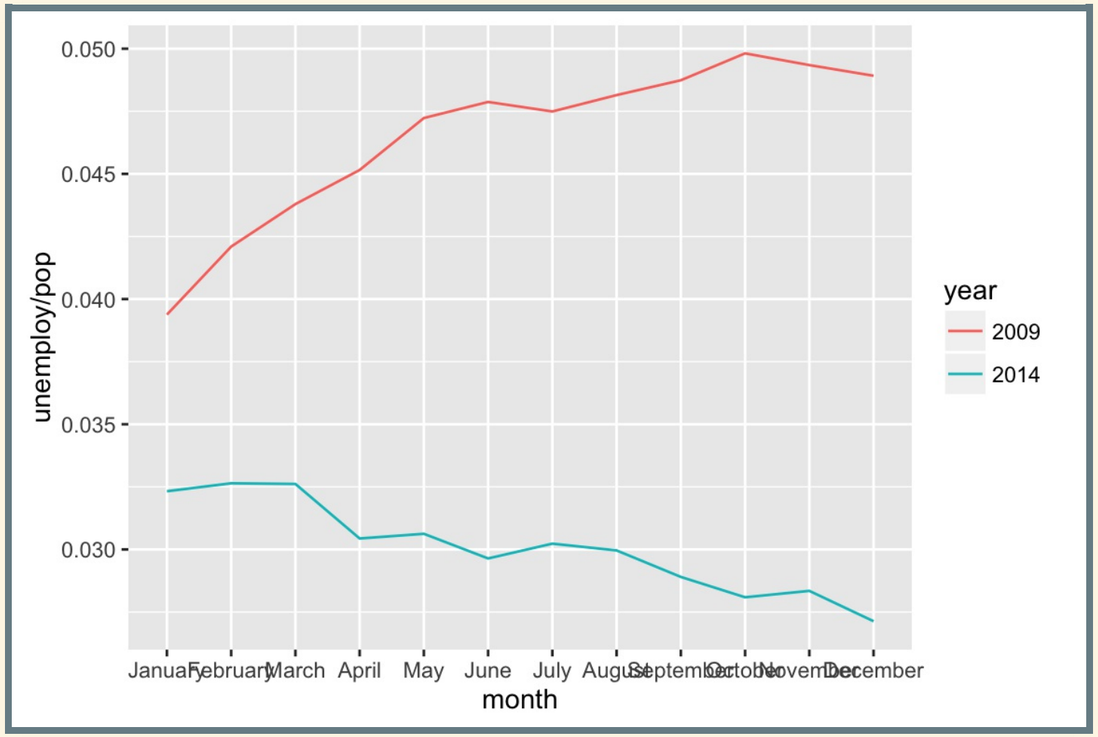
USING BASE GRAPHICS

```
data2009 <- subset(economics, year == 2009)
data2014 <- subset(economics, year == 2014)
plot(unemploy/pop ~ as.numeric(month), data = data2009,
     ylim = c(0.025, 0.05), type = "l")
lines(unemploy/pop ~ as.numeric(month), data = data2014,
      col = "red")
legend("topleft", c("2009", "2014"), title="Year",
       col=c("black", "red"), lty = c(1,1))
```



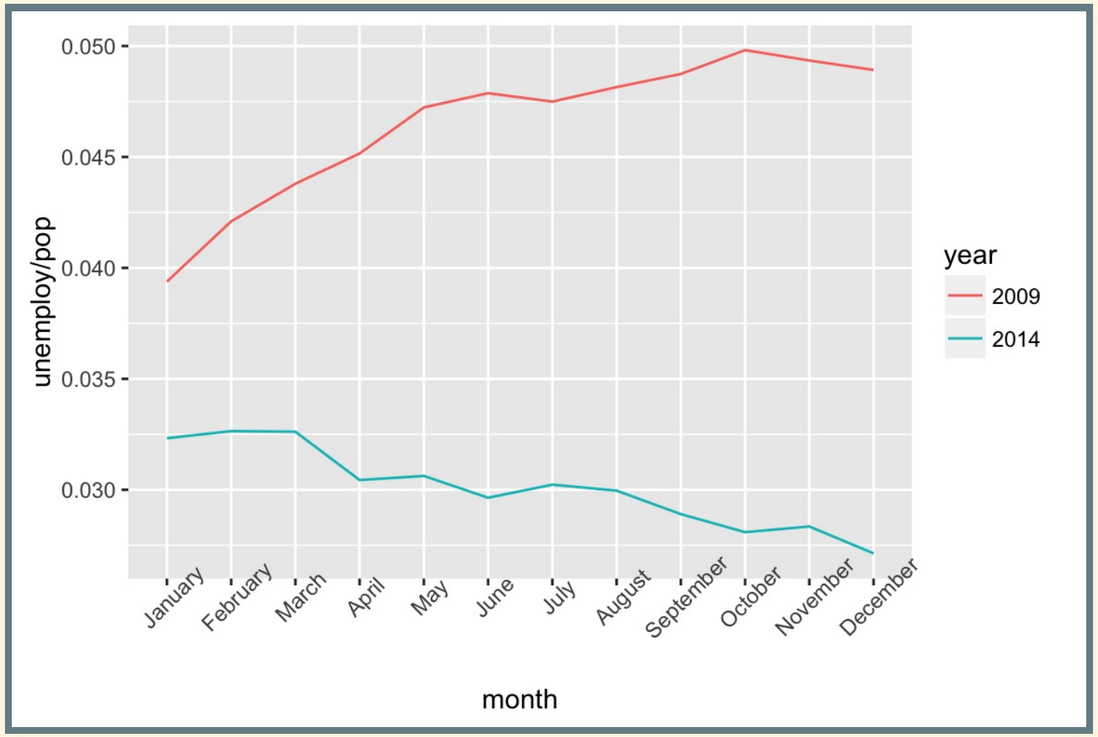
USING GGPLOT2

```
data2009_2014 <- subset(economics, year %in% c(2014, 2009))
# No need to specify a legend, it is produced automatically
ggplot(data = data2009_2014, aes(x = month, y = unemploy/pop)) +
  geom_line(aes(group = year, color = year))
```



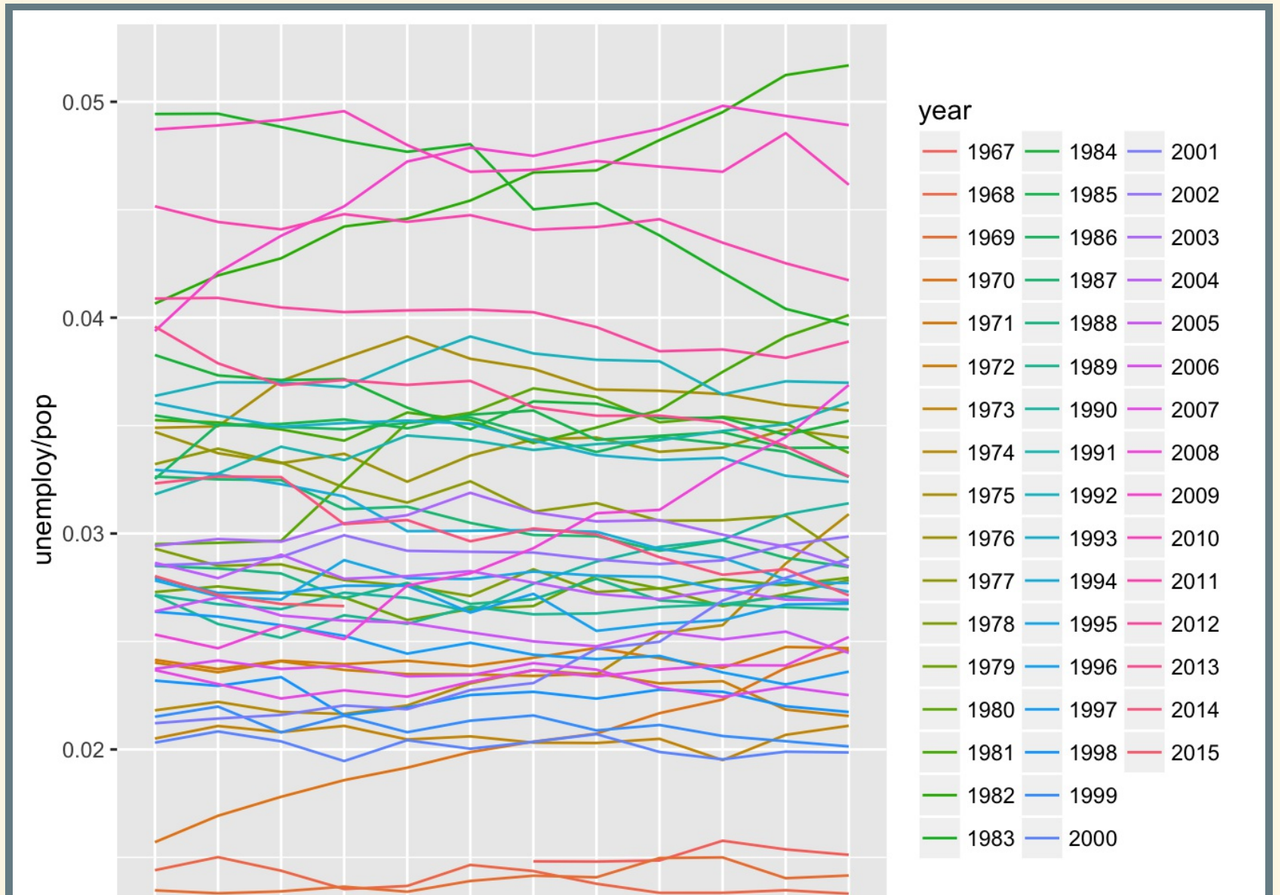
EASY FIX

```
data2009_2014 <- subset(economics, year %in% c(2014, 2009))
# No need to specify a legend, it is produced automatically
ggplot(data = data2009_2014, aes(x = month, y = unemploy/pop)) +
  geom_line(aes(group = year, color = year)) +
  theme(axis.text.x = element_text(angle = 45))
```



PLOTTING ALL THE YEARS TOGETHER IS EASY

```
ggplot(data = economics, aes(x = month, y = unemploy/pop)) +
  geom_line(aes(group = year, color = year)) +
    theme(axis.text.x = element_text(angle = 45))
```



EXAMPLE 2: DIAMONDS DATASET

Diamonds Dataset: is included in the `ggplot2` package.

The dataset contains the prices and other attributes of almost 54,000 diamonds. You can call `?diamonds` to learn more about the available attributes.

```
data("diamonds")
str(diamonds)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    53940 obs. of  10 variab
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ..
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ..
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ..
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ..
```

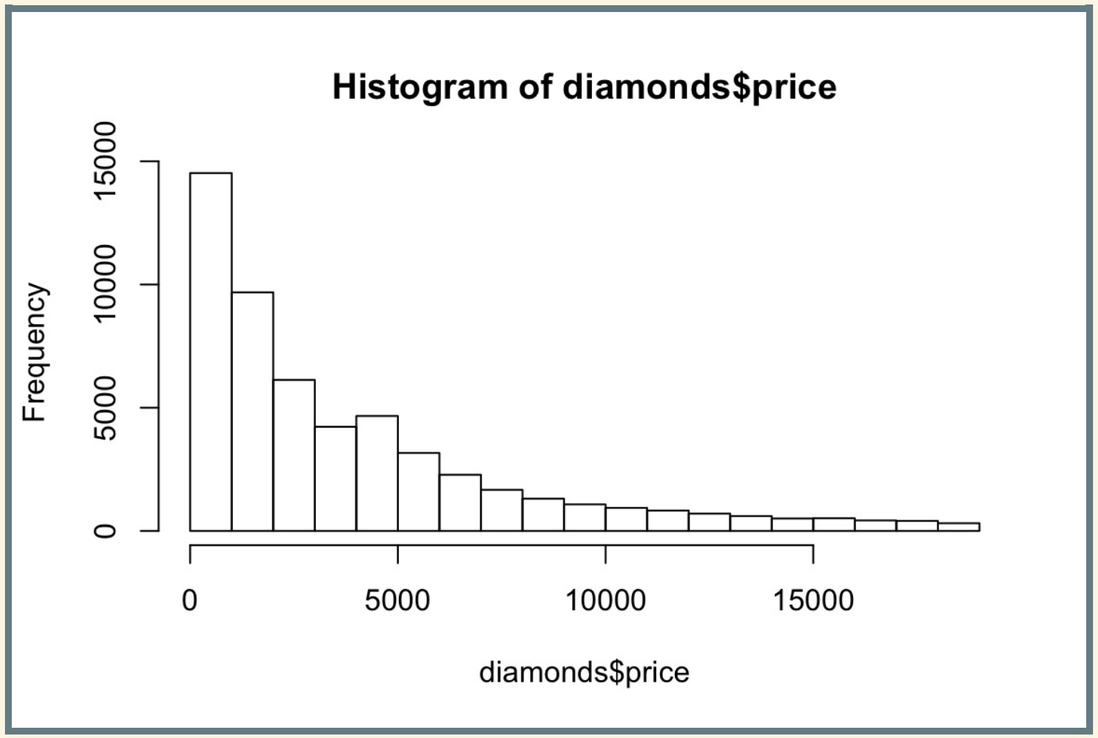
FIRST FEW ROWS

```
head(diamonds)
```

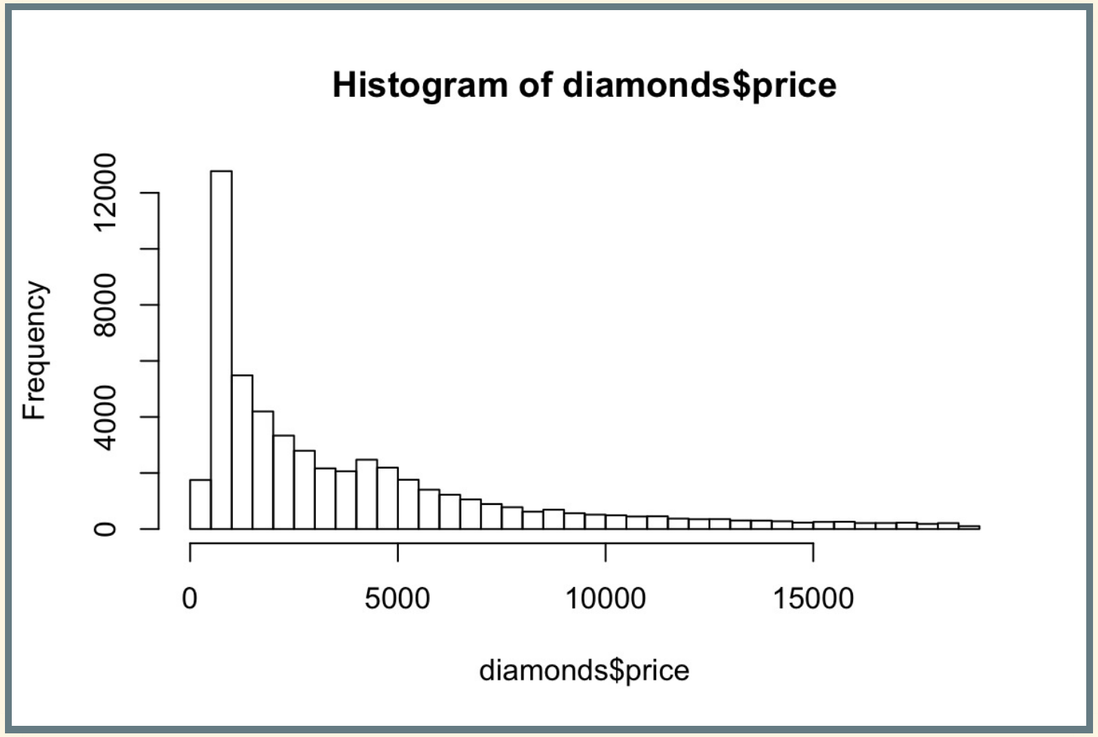
```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 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
```

DISTRIBUTION OF THE DIAMONDS PRICES WITH BASE GRAPHICS

```
hist(diamonds$price)
```



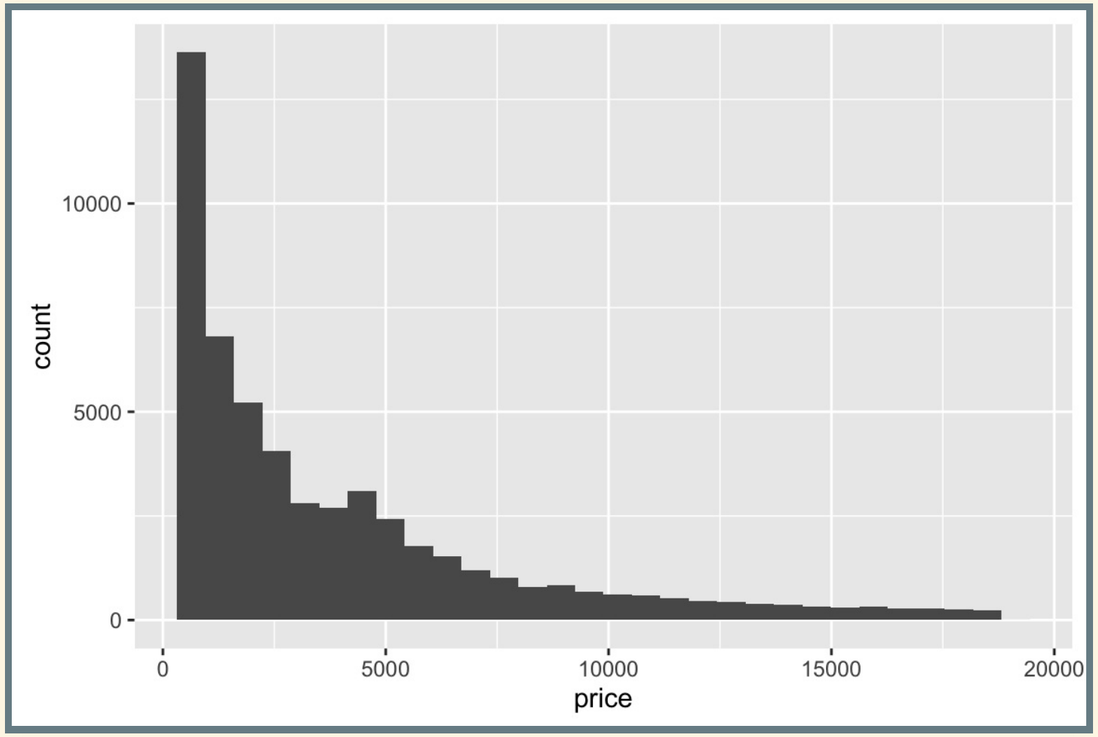
```r
# breaks can be a vector, a function or a single number
hist(diamonds$price, breaks = 50)
```



HISTOGRAMS WITH GGPLOT2

```
ggplot(diamonds, aes(x = price)) + geom_histogram()
```

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




```
# you can set bins, or binwidths
```

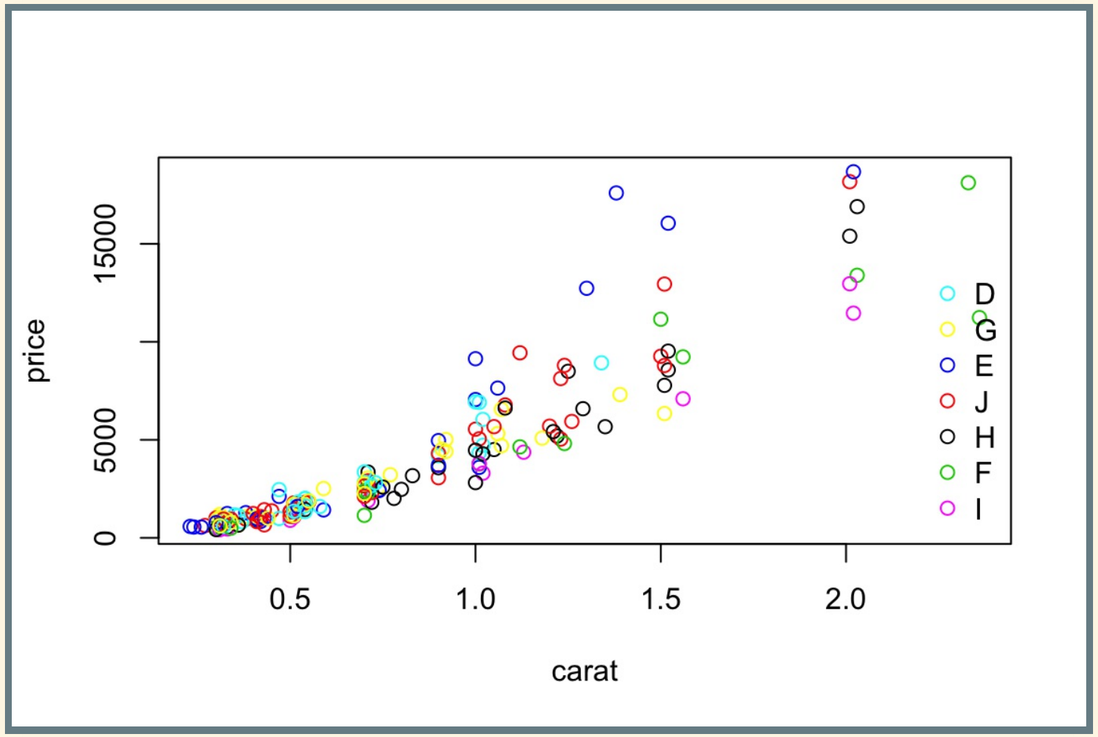
SUBSET OF THE DATA

- We select a random subset of the data.
- Show the relationship between the diamonds weights (carat = 200 mg) and their prices ($):

```
set.seed(12345) # Make the sample reproducible
dsmall <- diamonds[sample(nrow(diamonds), 200), ]
```

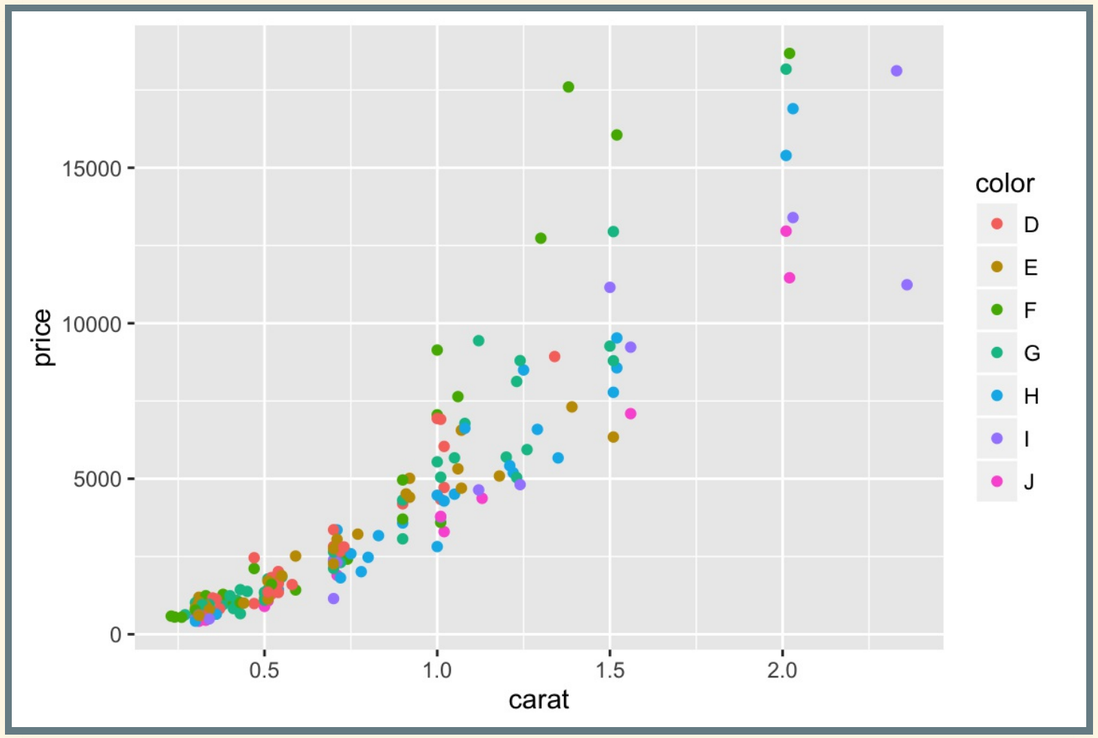
SCATTER PLOT WITH BASE GRAPHICS

```
colorMap <- data.frame(color = rainbow(length(unique(dsmall$color))))
rownames(colorMap) <- unique(dsmall$color)
plot(price ~ carat, data = dsmall, col = colorMap[dsmall$color, "color"]
legend(x = 'bottomright', legend = rownames(colorMap),
       col = colorMap$color, pch = par("pch"), bty = 'n', xjust = 1)
```



SCATTER PLOT WITH GGPLOT2

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



GEOMETRIC OBJECTS AND AESTHETICS

GEOMETIC OBJECTS:

Geometric objects are the actual items we put on a plot:

- points (geom_point, for scatter plots, dot plots, etc)
- lines (geom_line, for time series, trend lines, etc)
- boxplot (geom_boxplot, for, well, boxplots!)

A plot must have **at least** one `geom()`. There is no upper limit. You can add a geom to a plot using the + operator.

You can get a list of available geometric objects using the code below:

```
help.search("geom_", package = "ggplot2")
```

AESTHETIC MAPPING

In ggplot an aesthetic mapping, defined with aes(), describes how variables are mapped to visual properties or aesthetics.

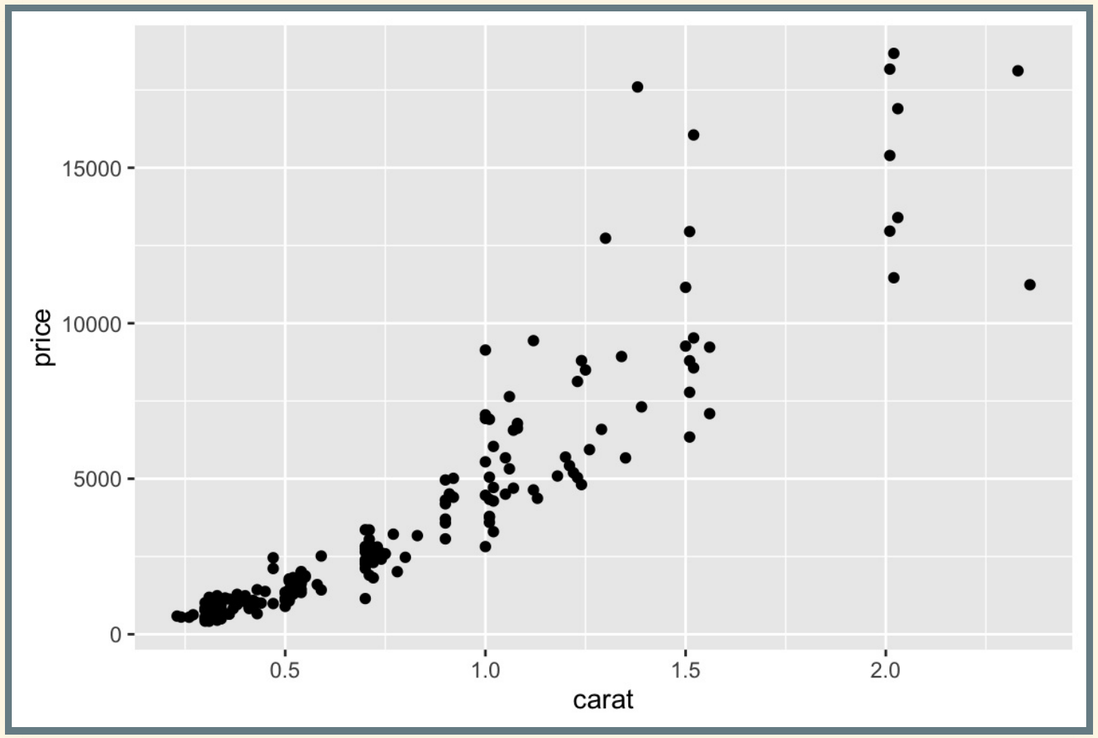
EXAMPLES OF AESTHETICS ARE:

- position (i.e., on the x and y axes)
- shape (of points)
- linetype
- size
- color (“outside” color)
- fill (“inside” color)

Each type of geom objects accepts only a subset of all aesthetics. Refer to the geom help pages to see what mappings each geom accepts.

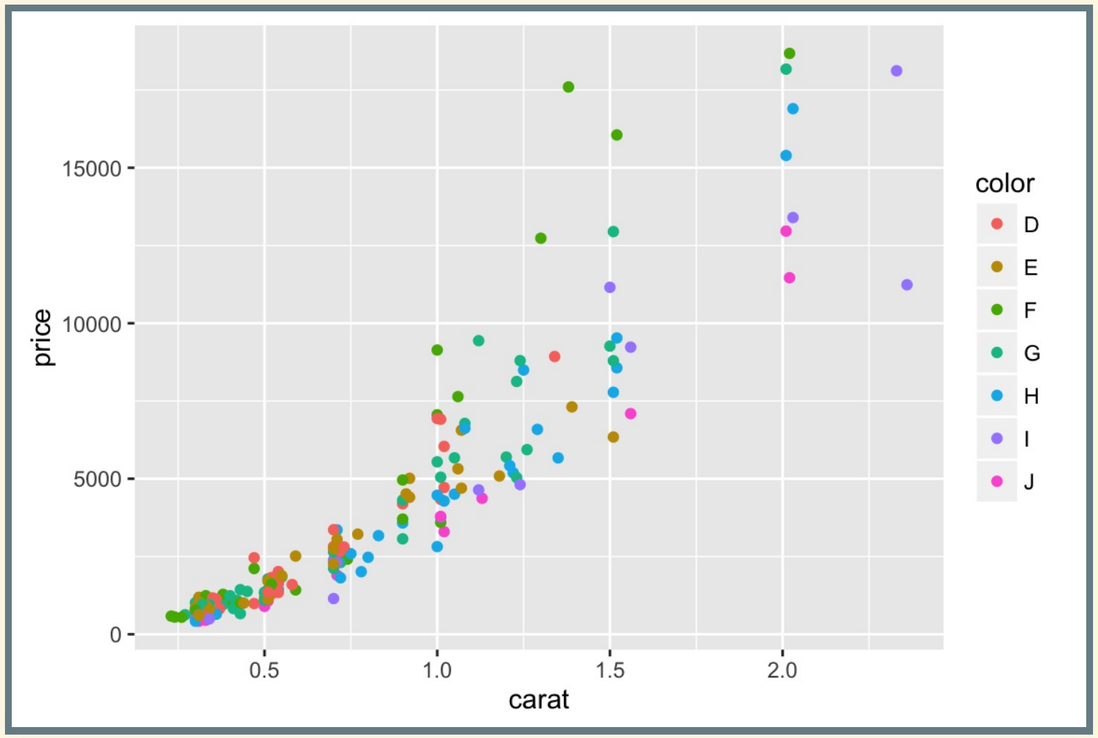
SCATTER PLOTS WITH GEOM_POINTS

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



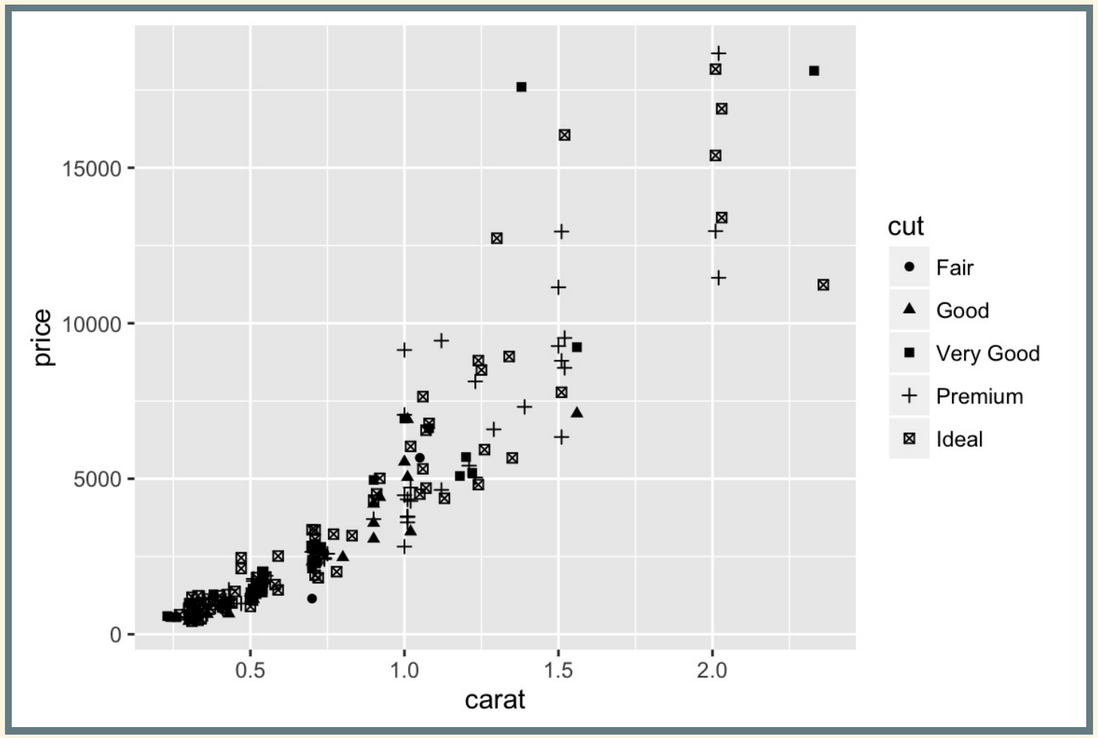
COLOR POINTS

```
# color by diamonds color
p1 + geom_point(aes(color = color))
```



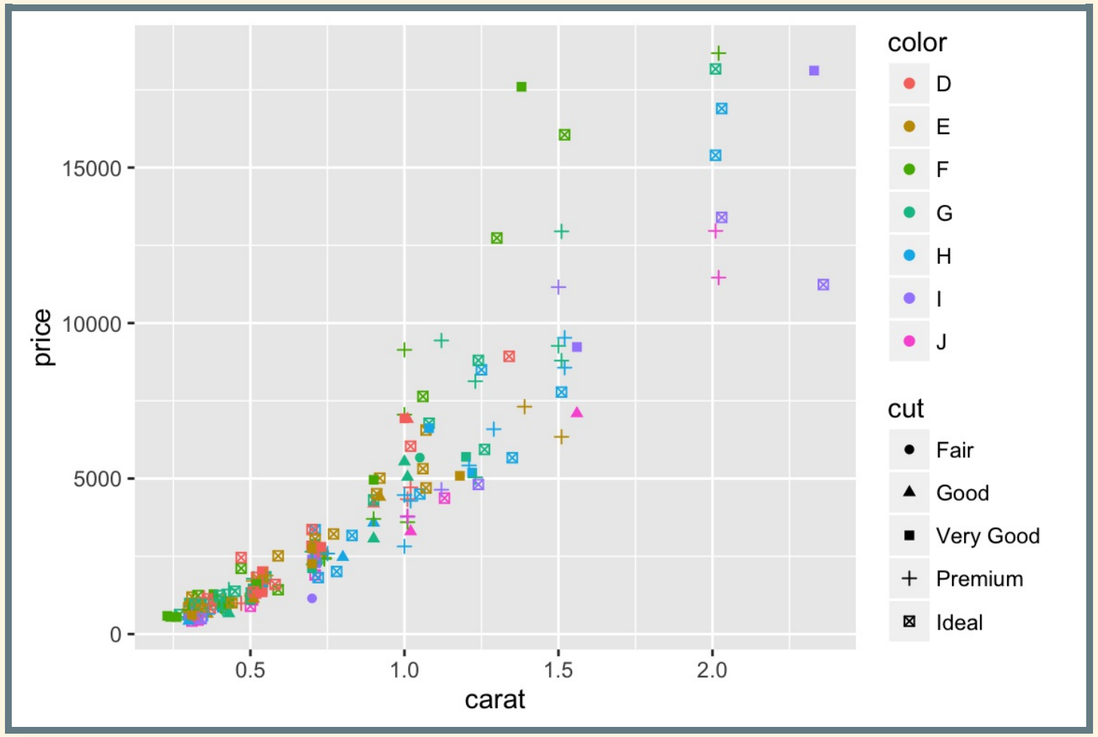
SET THE SHAPE OF THE POINTS

```
# set shape by diamond cut
p1 + geom_point(aes(shape = cut))
```



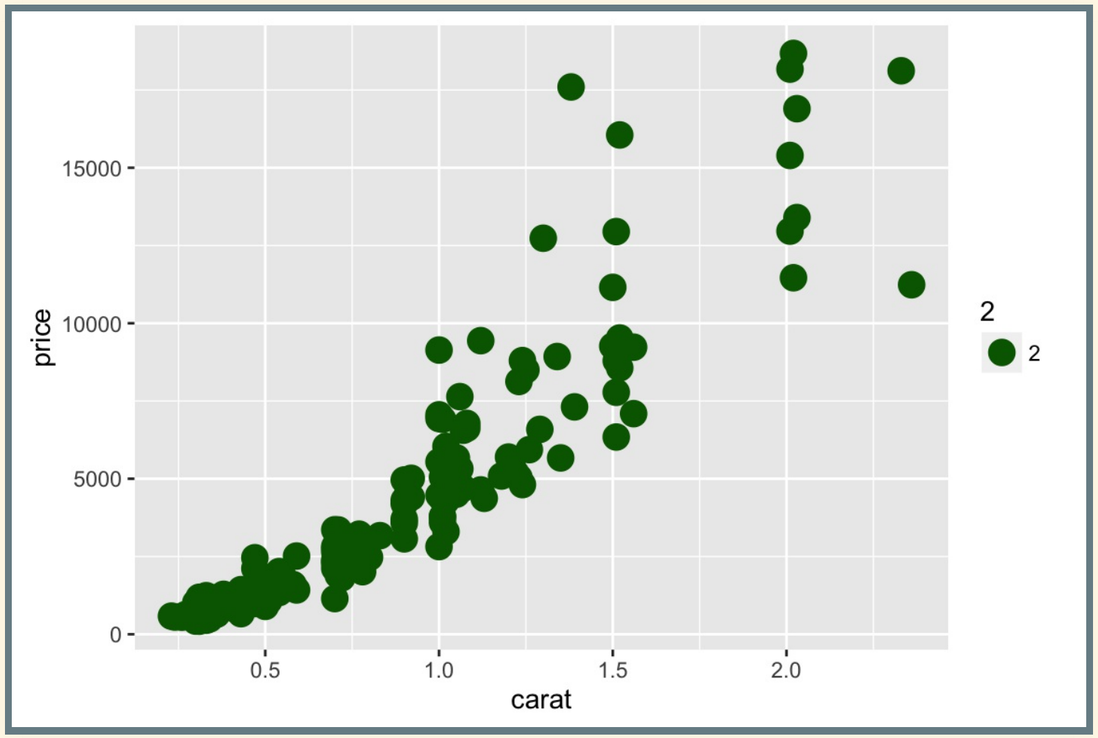
SET COLOR AND SHAPE

```
p1 + geom_point(aes(shape = cut, color = color))
```

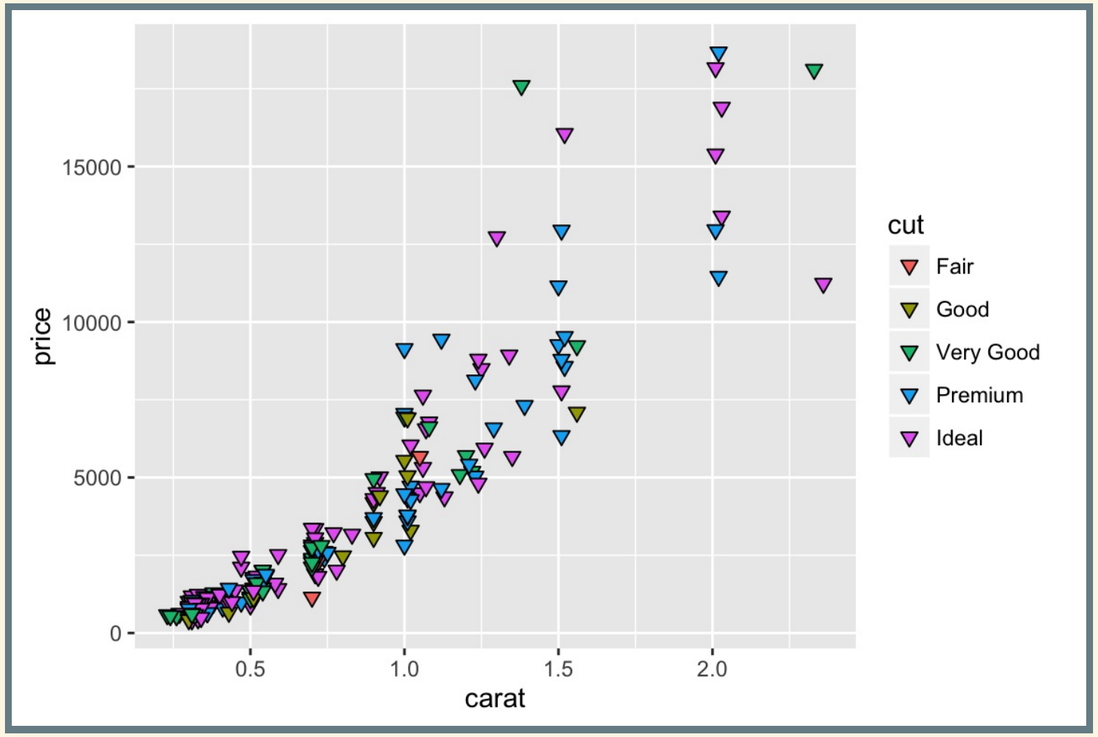


VARIABLE VS FIXED AESTHETICS

```
ggplot(data = dsmall, aes(x = carat, y = price)) +
  geom_point(aes(size = 2), color = "darkgreen")
```

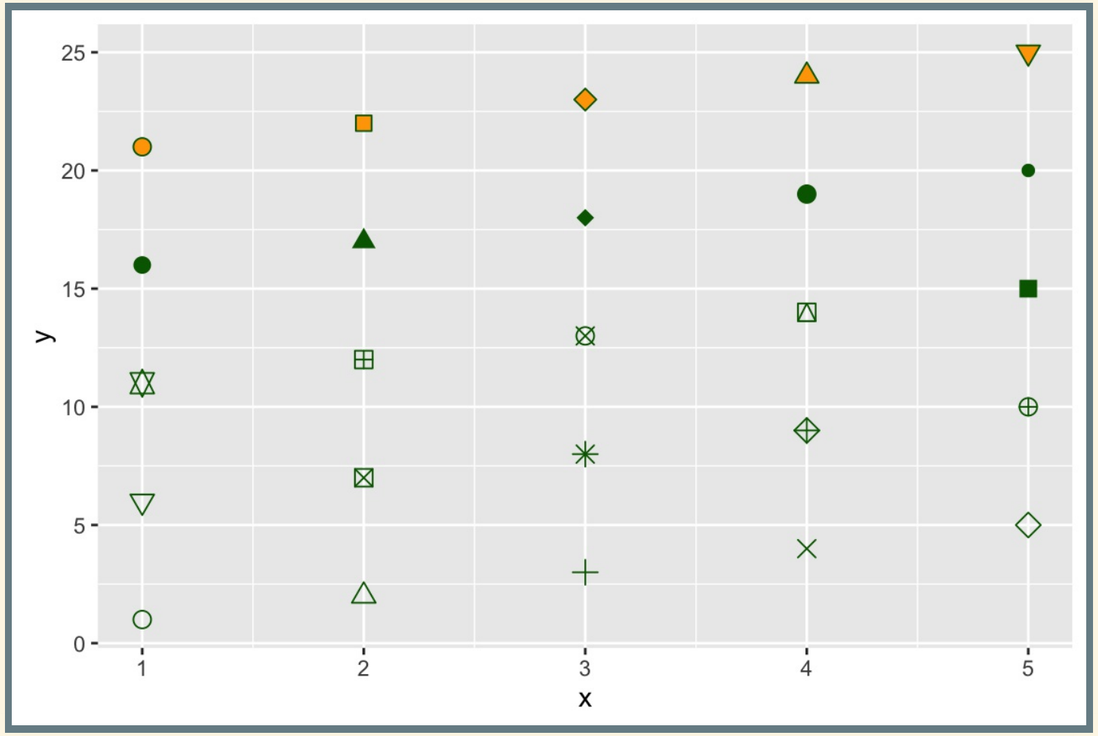


```
ggplot(data = dsmall, aes(x = carat, y = price)) +
  geom_point(aes(fill = cut), size = 2, color = "black", shape = 25)
```



AVAILABLE SHAPE CONFIGURATIONS

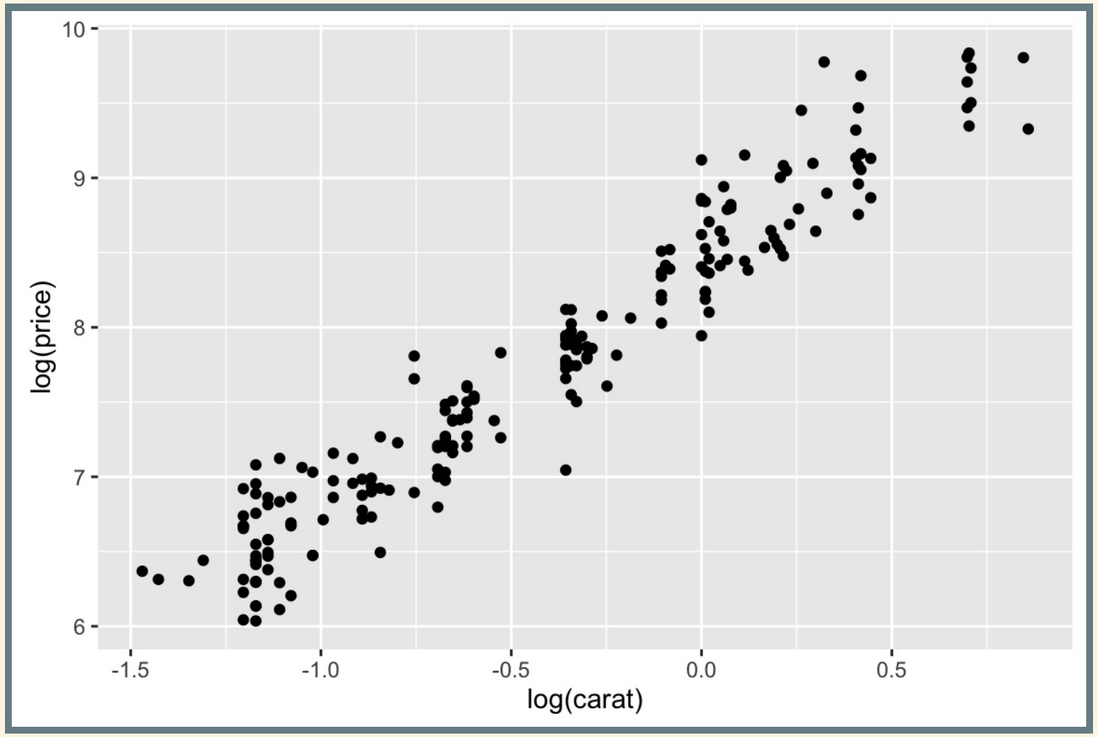
```
## See all 25 available symbols
df2 <- data.frame(x = 1:5 , y = 1:25, z = 1:25)
ggplot(df2, aes(x = x, y = y)) +
  geom_point(aes(shape = z), size = 3,
             colour = "darkgreen", fill = "orange") +
  scale_shape_identity()
```



DATA TRANSFORMATIONS

Transformation the variables directly.

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



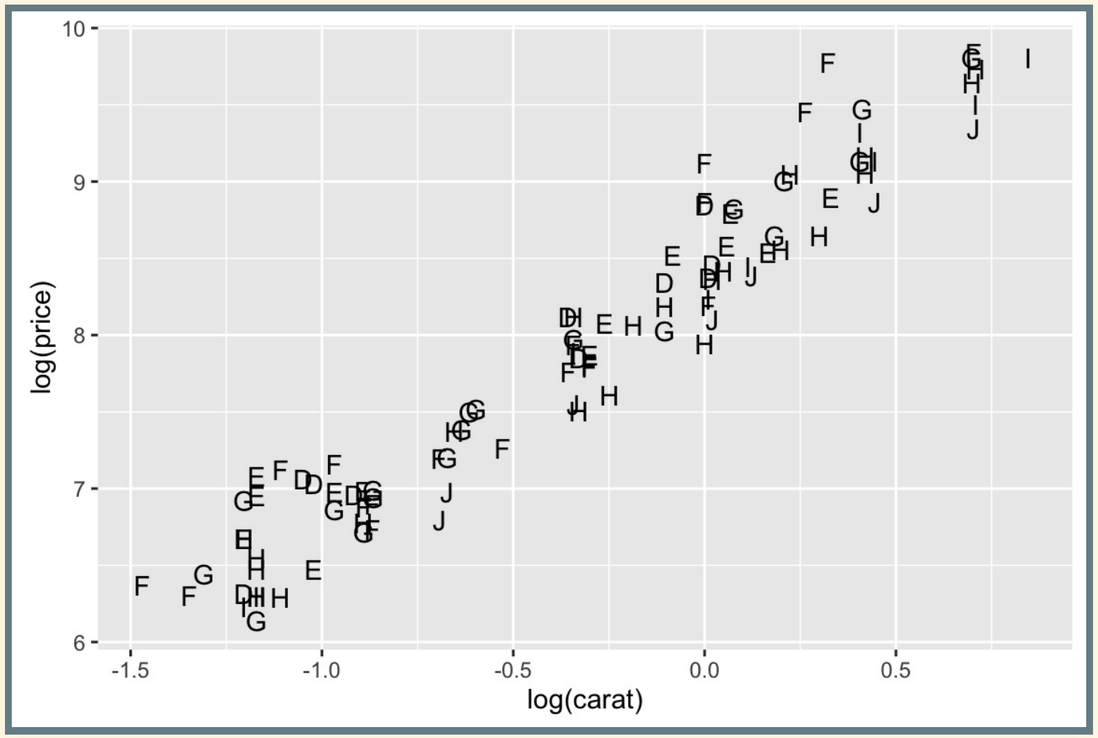
TEXT LABELS

We make an even smaller subset to show labelling:

```
set.seed(12345) # Make the sample reproducible
dsmall2 <- diamonds[sample(nrow(diamonds), 100), ]
```

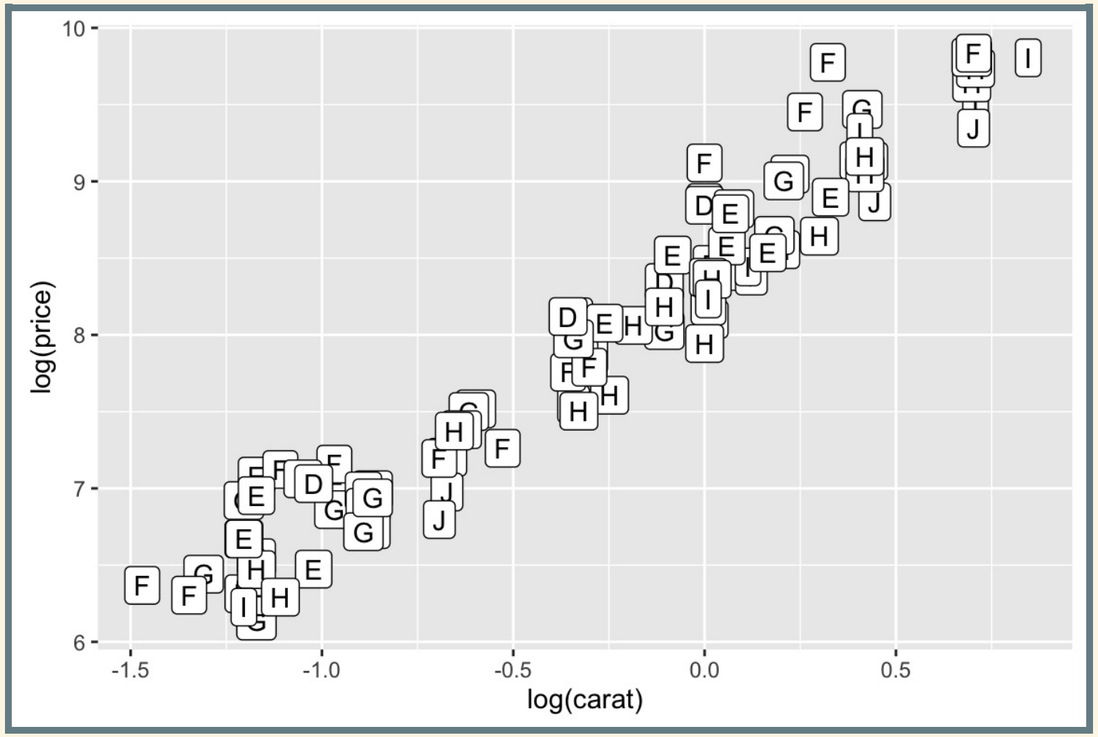
TEXT ONLY

```
p2 <- ggplot(dsmall2, aes(x = log(carat), y = log(price)))
p2 + geom_text(aes(label = color))
```



TEXT WITH RECTANGLE PLATES

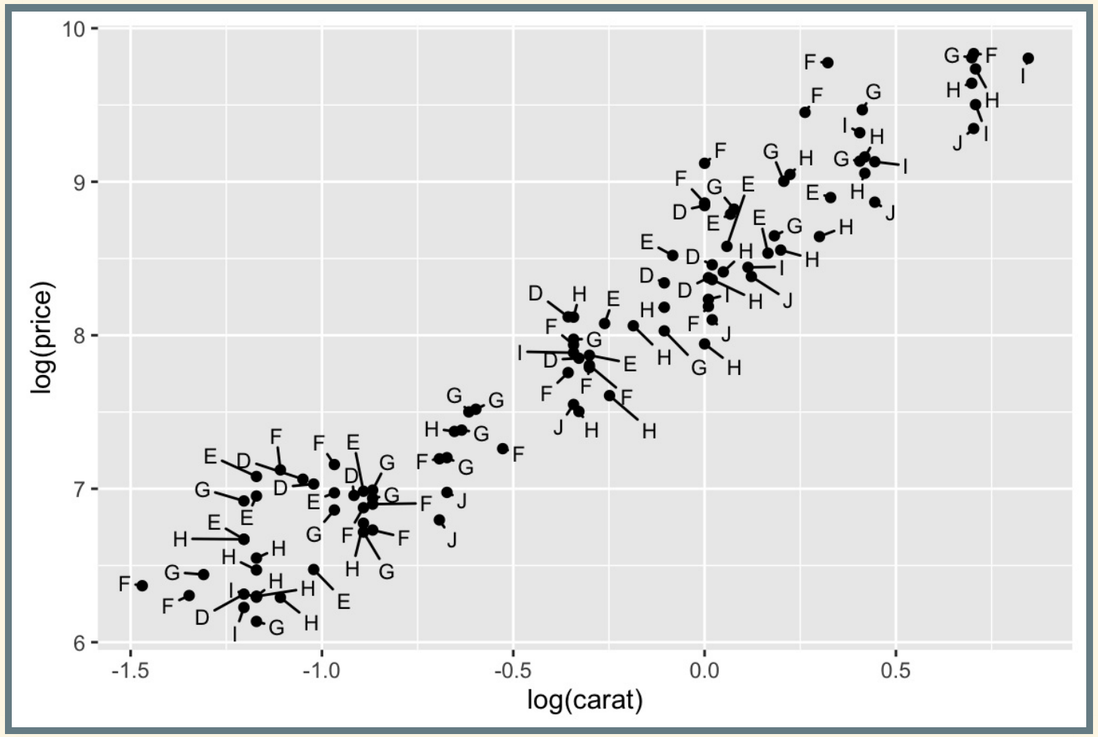
```
p2 + geom_label(aes(label = color))
```



GGREPEL PACKAGE FOR ANNOTATION

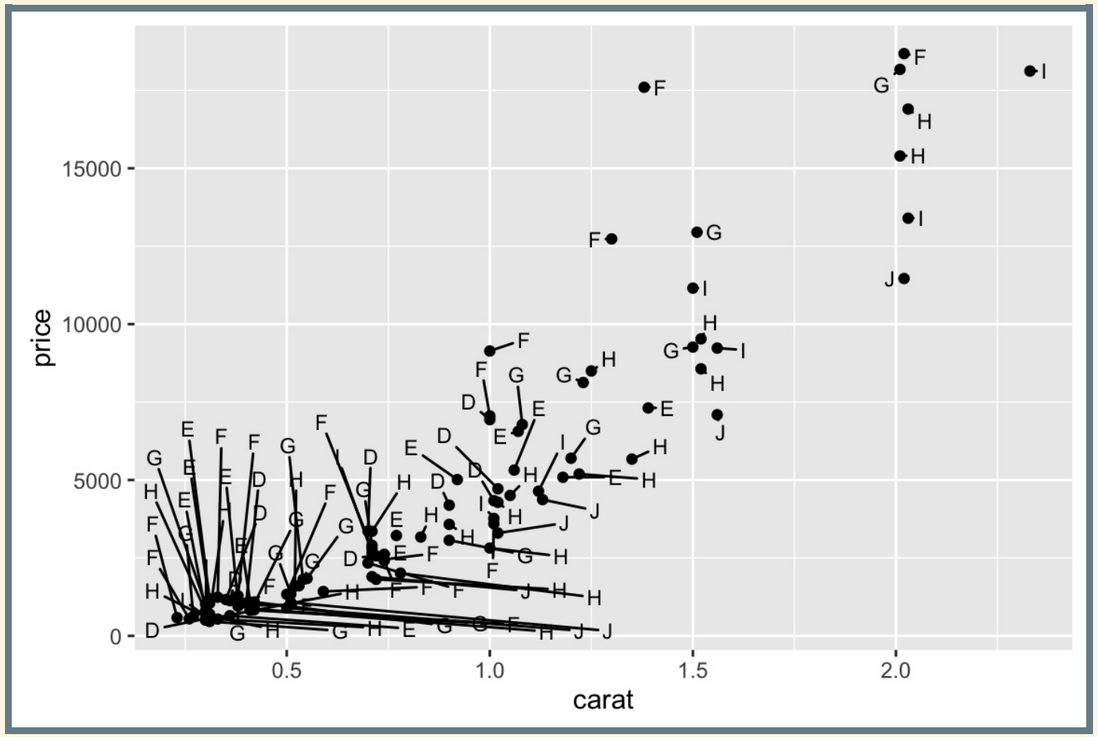
ggrepel helps annotating **overlapping labels.**

```
library(ggrepel)
p2 + geom_point() + geom_text_repel(aes(label=color), size = 3)
```



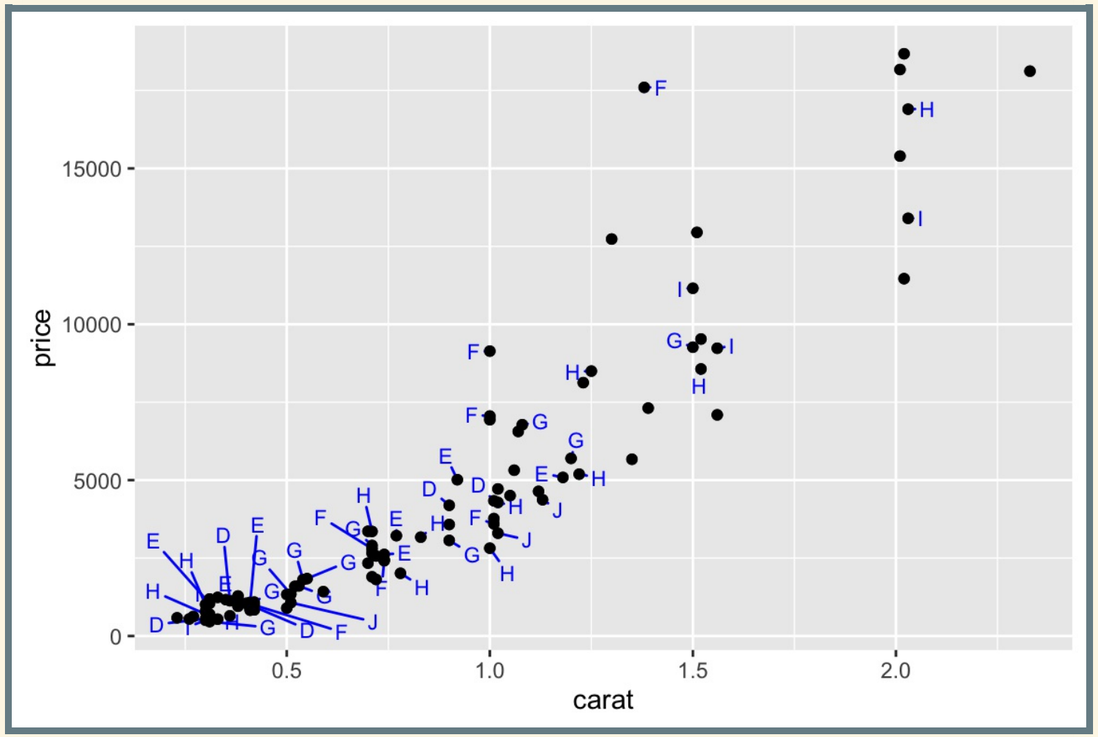
BUT it doesn't work well if points are densely clustered, then the lines pointing to the points will extend too far way, to make room for all the labels.

```
# Here we plot data NOT log transformed
ggplot(dsmall2, aes(x = carat, y = price)) + geom_point() +
  geom_text_repel(aes(label=color), size = 3)
```



THEN LABEL ONLY A SUBSET OF POINTS.

```
# sample indices of points to label
set.seed(123456)
idx <- sample(c(TRUE, FALSE), nrow(dsmall2), replace = TRUE, prob = c(0.
ggplot(dsmall2, aes(x = carat, y = price)) + geom_point() +
  geom_text_repel(data = subset(dsmall2, idx), aes(label=color),
                  size = 3, col = "Blue") + geom_point()
```



1. http://ggplot2.org/↩