Visualizations Continued & ggplot

STAT GU4206/GR5206 *Statistical Computing & Introduction to Data Science*

Gabriel Young
Columbia University

October 4, 2019

Course Notes

Last Time

Base R Graphics

Some More Plotting with Base R

Basics of Plotting

Recall,

- Visualization variation (of a single variable):
  - `hist()` – Histograms.
  - `barplot()` – Bargraphs.
- Visualizing covariation (of multiple variables):
  - `plot()` – Scatterplots.
  - `boxplot()` – Boxplots (box-and-whisker plots).

Basics of Plotting

The `plot()` function.

- The foundation of many of `R`'s graphics functions.
- Often one builds up the graph in stages with `plot()` as a base.
- Each call to `plot()` begins a new graph window.
- Takes arguments, called *graphical parameters*, to change various aspects of the plot. (`?par`)

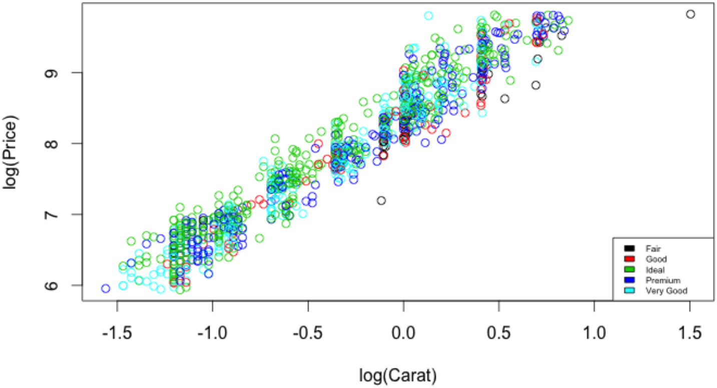
Diamonds Dataset

- Recall the diamonds data set. (diamonds.csv)
- Run diamonds <- read.csv("diamonds.csv", as.is = TRUE).

```
> diamonds          <- read.csv("diamonds.csv", as.is = T)
> diamonds$cut      <- factor(diamonds$cut)
> diamonds$color    <- factor(diamonds$color)
> diamonds$clarity <- factor(diamonds$clarity)
> set.seed(1)
> rows <- dim(diamonds)[1]
> diam <- diamonds[sample(1:rows, 1000), ]
```

Building a Visualization: An Example

```
> plot(log(diam$carat), log(diam$price), col = diam$cut)
> legend("bottomright", legend = levels(diam$cut),
+         fill = 1:length(levels(diam$cut)), cex = .5)
```

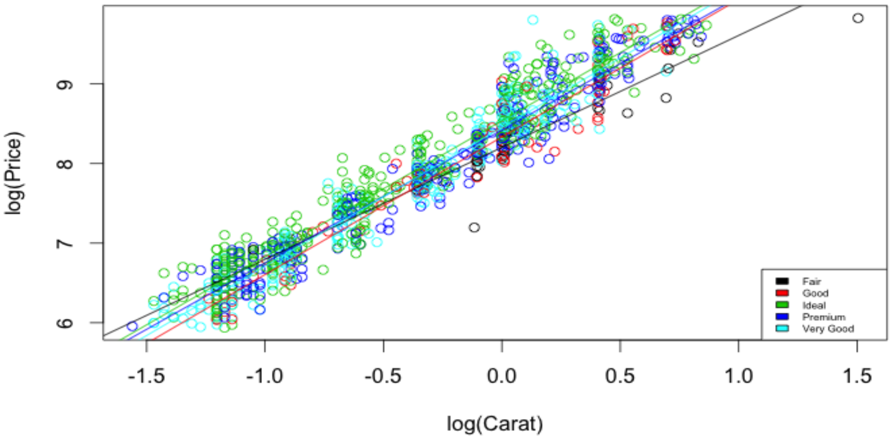


Building a Visualization: An Example

Let's instead plot a regression line for each cut separately.

```
> cuts        <- levels(diam$cut)
> col_counter <- 1
> for (i in cuts) {
+   this_cut    <- diam$cut == i
+   this_data   <- diam[this_cut, ]
+   this_lm     <- lm(log(this_data$price)
+                     ~ log(this_data$carat))
+   abline(this_lm, col = col_counter)
+   col_counter <- col_counter + 1
+ }
```

Building a Visualization: An Example



Check Yourself

Exercise:

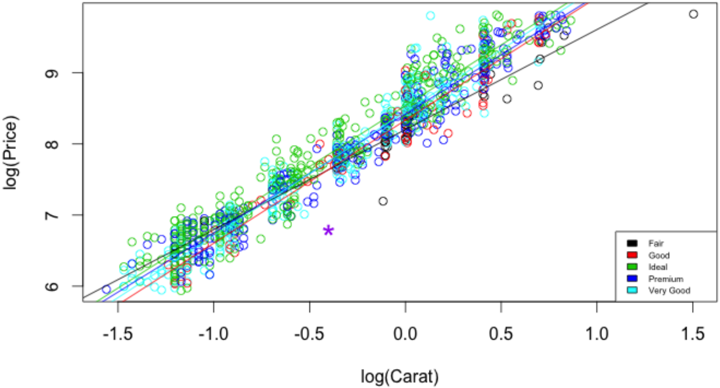
Use the built-in `iris` dataset.

- Create a new column `Setosa` that takes a 1 if the iris is a setosa and a 0 otherwise.
- Plot iris `Sepal.Width` on the x-axis and `Sepal.Length` on the y-axis. Color the points according to whether the iris is a setosa or not.
- Plot two regression lines on the plot, one for the setosa iris and one for non-setosa iris.

Building a Visualization: An Example

We add a new point for a diamond that is $898 and 0.67 carats.

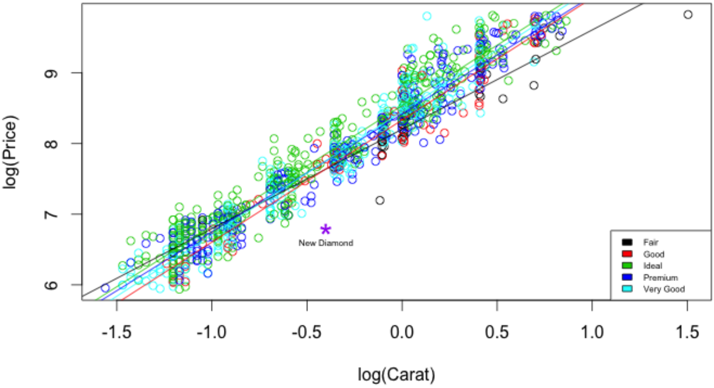
```
> points(-0.4, 6.8,  pch = "*", col = "purple")
```



Building a Visualization: An Example

We add text to the new point we just added.

```
> text(-0.4, 6.8 - .2, "New Diamond", cex = .5)
```



Useful Graphical Parameters

The table below lists a selection of R's graphical parameters. More info at `http://www.statmethods.net/advgraphs/parameters.html` or using `?par`.

Parameter	Description
`pch`	*Point Character*. Character of the points in the plot.
`main`	Title of the plot.
`xlab, ylab`	Axes labels.
`lty`	*Line Type*. E.g. 'dashed', 'dotted', etc.
`lwd`	*Line Width*. Line width relative to default = 1.
`cex`	*Character Expand*. Character size relative to default = 1.
`xlim, ylim`	The limits of the axes.
`mfrow`	Plot figures in an array (e.g. next to each other).
`col`	Plotting color.

Section II

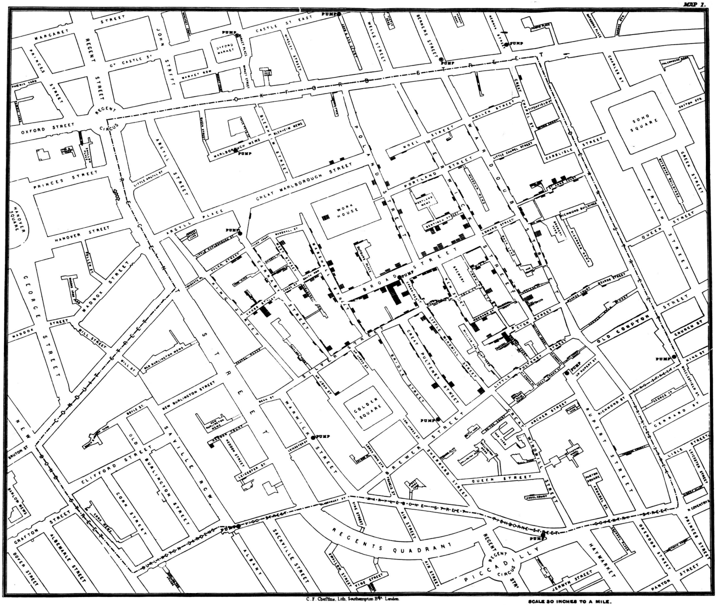
Data Visualization

Section II

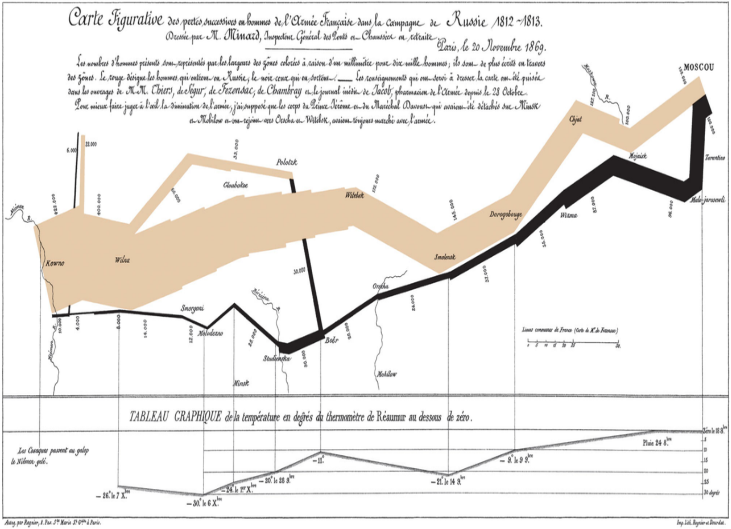
Good Visualizations

In data science, good visualizations should give you more information than you can see in just the data table itself.

Good Visualizations - John Snow 1854 (Wikipedia)



Good Visualizations - Charles Joseph Minard 1896 (Wikipedia)



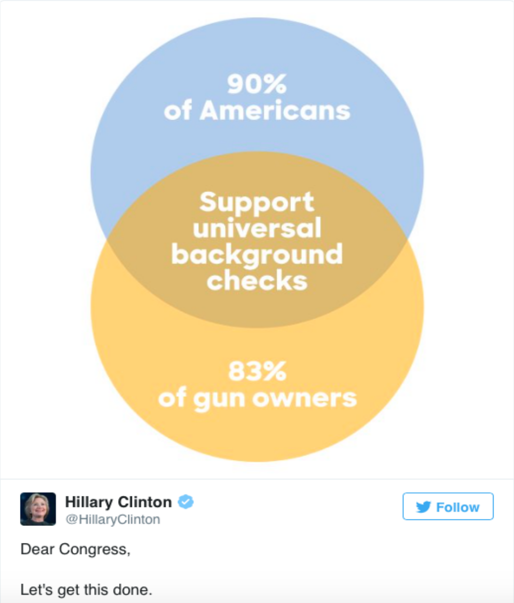
Good Visualizations - Charles Joseph Minard 1896 (Wikipedia)

Minard Graph

Minard shows six variables:

- Number of soldiers,
- Direction of the march,
- Location coordinates,
- Temperature on the return journey,
- Location on dates in November and December.

Bad Visualizations - Hillary Clinton



Hillary Clinton
@HillaryClinton

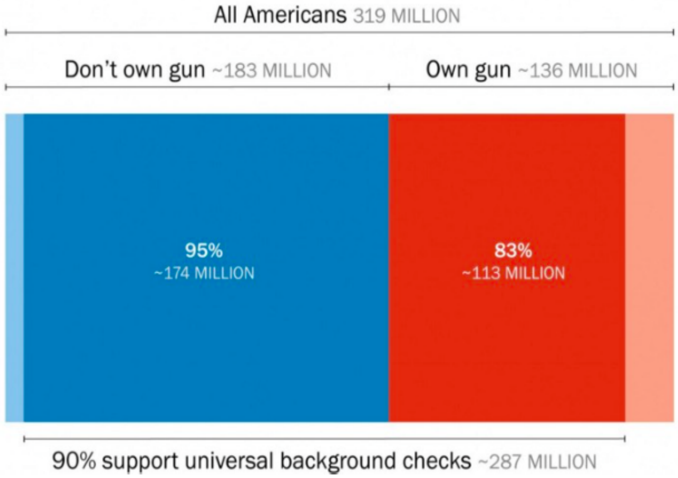
Dear Congress,

Let's get this done.

Good Visualizations - Washington Post

Improving the Clinton campaign's terrible graph

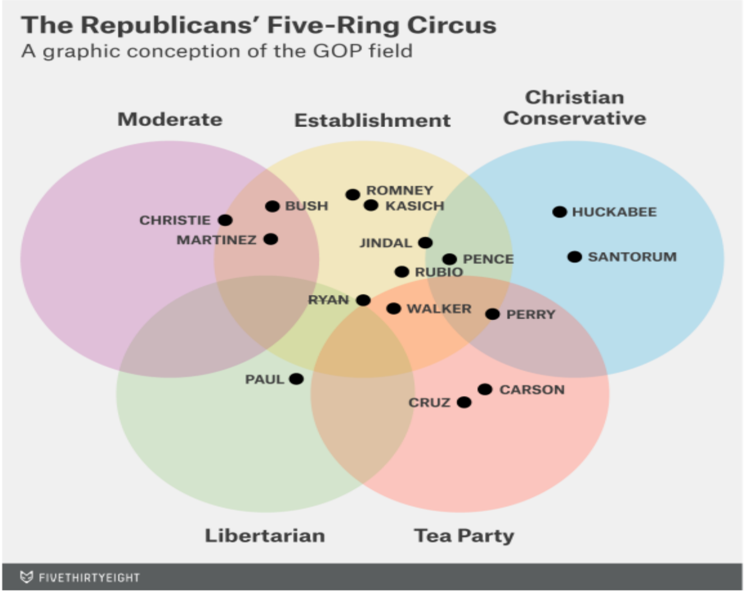
Population estimates from the Census Bureau. Gun ownership estimate based on calculations from Gallup compared with Census household data. Percentages based on Clinton campaign figures.



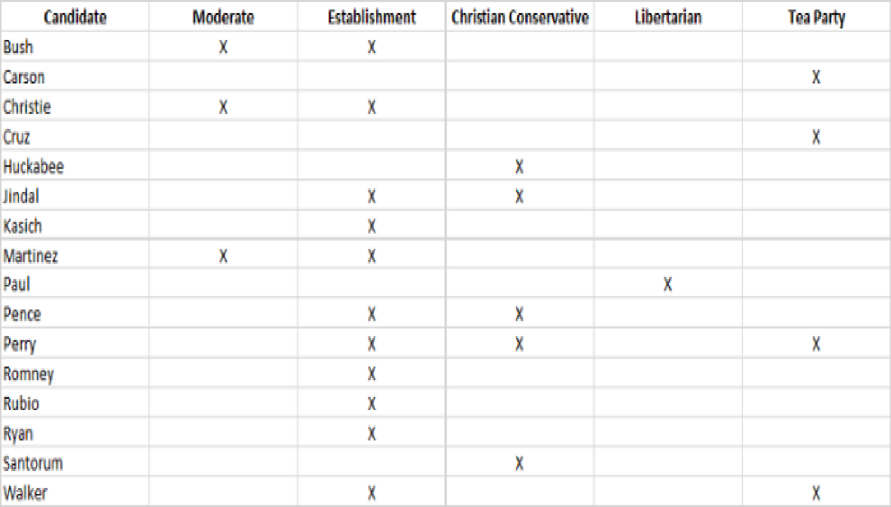
Bad Visualizations

Even statisticians are sometimes bad at making visualizations!

Bad Visualizations - Nate Silver



Bad Visualizations - Nate Silver

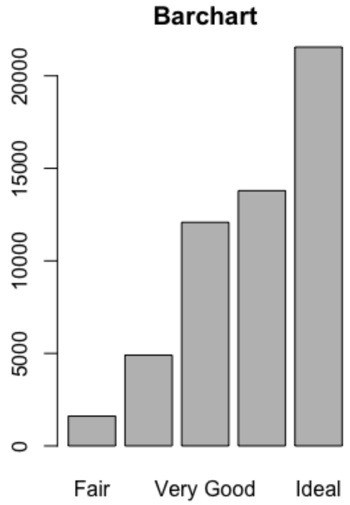


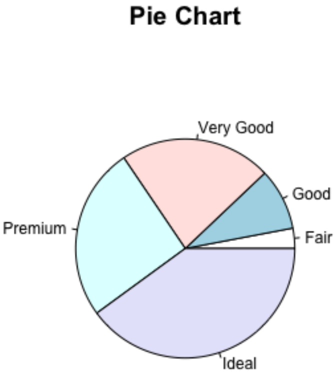
Candidate	Moderate	Establishment	Christian Conservative	Libertarian	Tea Party
Bush	X	X			
Carson					X
Christie	X	X			
Cruz					X
Huckabee			X		
Jindal		X	X		
Kasich		X			
Martinez	X	X			
Paul				X	
Pence		X	X		
Perry		X	X		X
Romney		X			
Rubio		X			
Ryan		X			
Santorum			X		
Walker		X			X

Bad Visualizations

- A piechart is a bad graphic for visualizing categorical data.
- This is a biased opinion from a statistician.
- Statisticians typically do not like pie charts.
- Barcharts display the same information as a piechart and the graphic is easier to interpret.

Barchart vs. Piechart


Barchart


Pie Chart

Task

Identify what the following function does.

Code

```
> pie.chart <- function(data) {
+ print("I suck")
+ }
```

Task

Identify what the following function does.

Code

```
> pie.chart <- function(data) {
+ print("I suck")
+ }
```

The pie.chart function prints ”I suck”

```
> pie.chart(c("Red","Red","Blue"))

[1] "I suck"
```

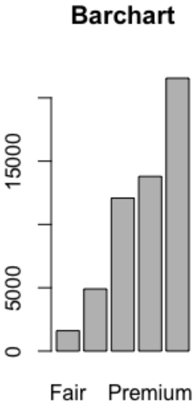
Side note: plots per window

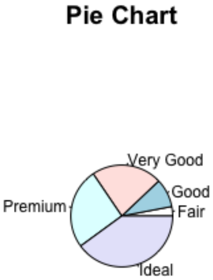
Change graphical parameters

- Use the `par()` function to change graphical parameters.
- Change plots per window with `mfrow`
- The default is `mfrow=c(1,1)`

```
> par(mfrow=c(1,2))
> barplot(height = table(diamonds$cut),
+           names.arg = names(table(diamonds$cut)),
+           main="Barchart")
> pie(table(diamonds$cut), labels = names(table(diamonds$cut))
+      main="Pie Chart",cex=.75)
```

Barchart vs. Piechart





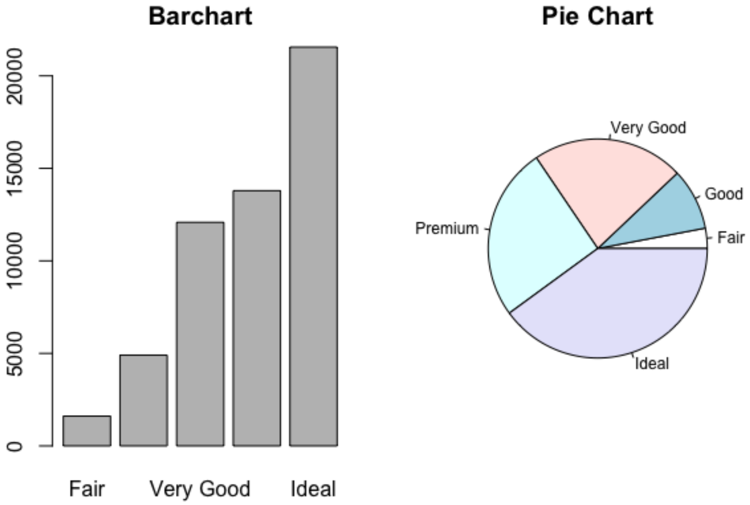
Side note: plot margins

Change graphical parameters

- Change margins with mar or mai
- Note: mar=c(bottom,left,top,right)
- The default is mar=c(5.1, 4.1, 4.1, 2.1))

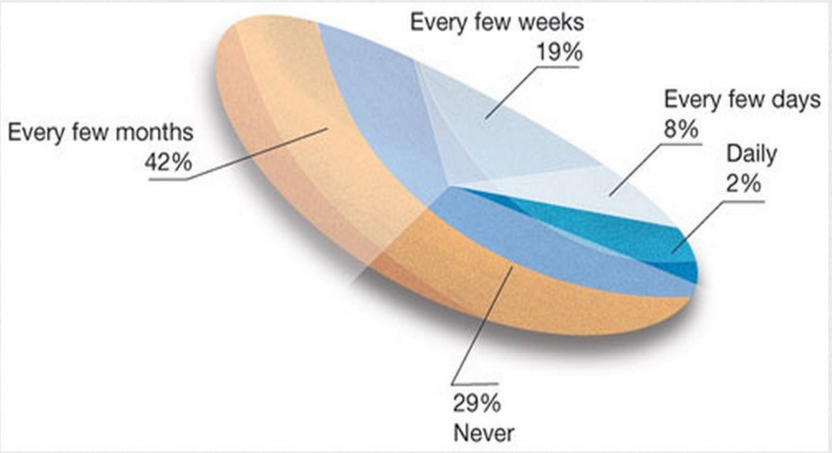
```
> par(mfrow=c(1,2),mai=c(.5,.4,.5,.4))
> barplot(height = table(diamonds$cut),
+          names.arg = names(table(diamonds$cut)),
+          main="Barchart")
> pie(table(diamonds$cut), labels = names(table(diamonds$cut))
+      main="Pie Chart",cex=.75)
```

Barchart vs. Piechart



Bad Visualizations

- Some more bad visualizations
- See the link: businessinsider



Good Visualizations

- Keep things simple in terms of color and presentation!
- Try not adding non-needed dimensions to a plot, i.e., 3D bar chart describing one categorical variable.
- Showing more dimensions on lower a dimensional plot is encouraged, i.e, diamond price versus carat split by cut.
- Barcharts are a better way to summarize categorical data compared to piecharts.

Section III

Advanced Visualization Techniques

ggplot2

- R has several systems for making graphs (we've looked at the base R functions).
- ggplot2 is one of the most elegant and flexible.
- ggplot2 uses a coherent system (or 'grammar') for describing and building graphs.

ggplot2

- R has several systems for making graphs (we've looked at the base R functions).
- ggplot2 is one of the most elegant and flexible.
- ggplot2 uses a coherent system (or 'grammar') for describing and building graphs.

Need to run `install.packages("ggplot2")` now and `library("ggplot2")` every time you want to use it!

ggplot2

We study ggplot2 using the mpg dataset. Let's try to answer the question: do cars with bigger engines use more fuel than cars with small engines?

ggplot2

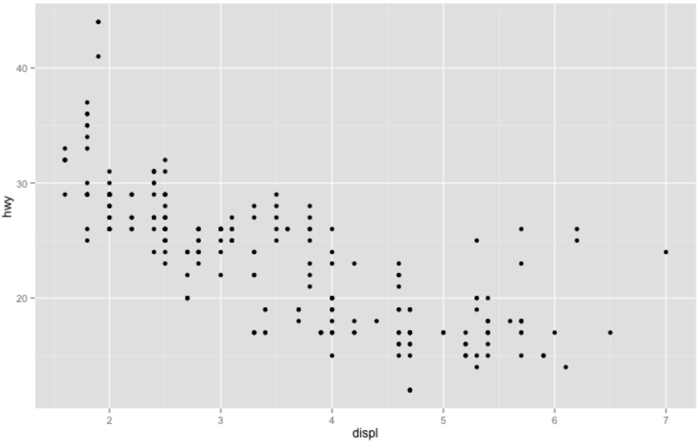
We study ggplot2 using the mpg dataset. Let's try to answer the question: do cars with bigger engines use more fuel than cars with small engines?

Read about the data using ?mpg.

```
> dim(mpg)

[1] 234  11

> head(mpg, 3)

# A tibble: 3 x 11
  manufacturer model displ  year   cyl trans    drv     cty
  <chr>        <chr> <dbl> <int> <int> <chr>    <chr> <int>
1 audi         a4     1.80  1999     4 auto(l5) f        18
2 audi         a4     1.80  1999     4 manual(âĂę f        21
3 audi         a4     2.00  2008     4 manual(âĂę f        20
# ... with 3 more variables: hwy <int>, fl <chr>,
#   class <chr>
```

A First Plot

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



A First Plot

Let's break apart the code:

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

A First Plot

Let's break apart the code:

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

- Begin a plot with `ggplot()`.
  - It creates the coordinate axis that you add to.
  - The first argument is the dataset

A First Plot

Let's break apart the code:

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

- Begin a plot with `ggplot()`.
  - It creates the coordinate axis that you add to.
  - The first argument is the dataset
- Next you want to add layers to the plot.
  - In our example: `geom_point()` adds a layer of points.
  - Lots of different `geom` functions doing different things.

A First Plot

Let's break apart the code:

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

- Begin a plot with `ggplot()`.
  - It creates the coordinate axis that you add to.
  - The first argument is the dataset
- Next you want to add layers to the plot.
  - In our example: `geom_point()` adds a layer of points.
  - Lots of different `geom` functions doing different things.
- `geom` functions take `mapping` arguments.
  - Defines how variables in your dataset are mapped to visual properties.
  - Always paired with `aes()`.
  - The `x` and `y` arguments specify which variables to map to the axes.

A First Plot

General structure:

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

To create a plot, replace the bracketed sections in the code above with a datatset, a `geom` function, and a set of mappings.

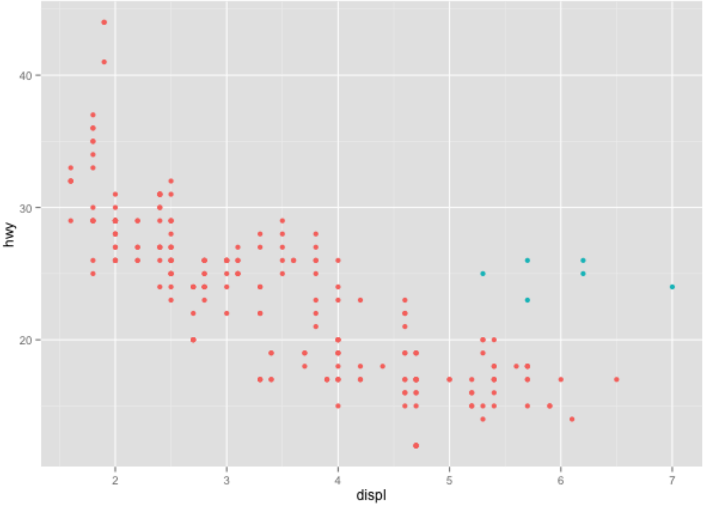
From this template, we can make many different kinds of graphs using `ggplot`.

Check Yourself

Tasks

- Plot just `ggplot(data = mpg)`. What do you get?
- Make a scatterplot of `hwy` vs. `cyl`.
- Make a scatterplot of `class` vs. `drv`. Why is this plot not useful?

Aesthetic Mappings



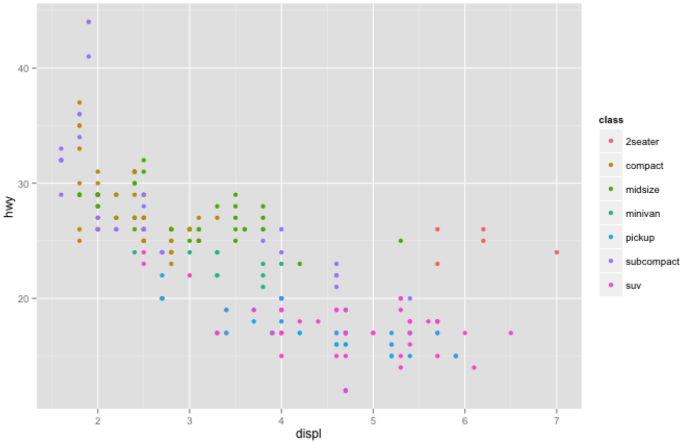
The blue points seem to have a different trend than the rest – possibly hybrids? We study car `class` to find out.

Aesthetic Mappings

- We can add a third variable to a scatterplot by mapping it to an **aesthetic**.
- An **aesthetic** is a visual property of the objects in the plot.
- Things like size, color, shape of points.

Mapping Aesthetics

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



Check Yourself

Tasks

- Instead of mapping `class` to the `color` aesthetic, map it to the `alpha` aesthetic or the `size` aesthetic.
- Instead of mapping `class` to the `color` aesthetic, map it to the `shape` aesthetic. Note that `ggplot()` will only use 6 shapes at a time. What does this mean for our plot?
- What does the following code do?

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

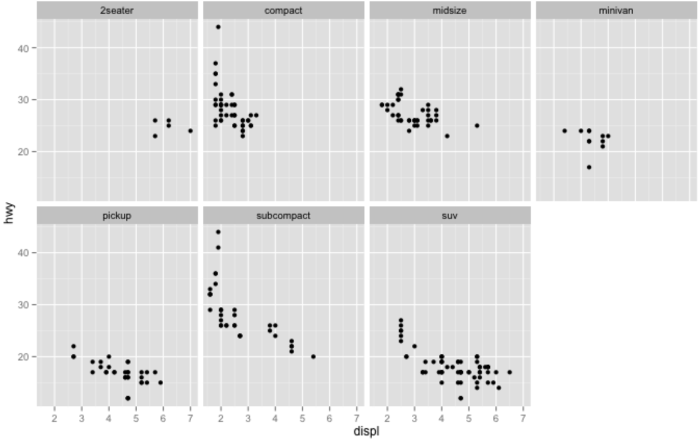
- Map a continuous variable in the `mpg` dataset, like `cty`, to the `alpha`, `shape`, and `size` aesthetics. What does this do?

Facets

- We saw we could add categorical variables to plots using aesthetics.
- Can also do this by splitting the plot into **facets**, which are subplots that each display one subset of the data.
- Use the `fact_wrap()` command to facet a plot by a single variable.
- The argument is a formula created with ˜ followed by a variable name.

Facets

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



Check Yourself

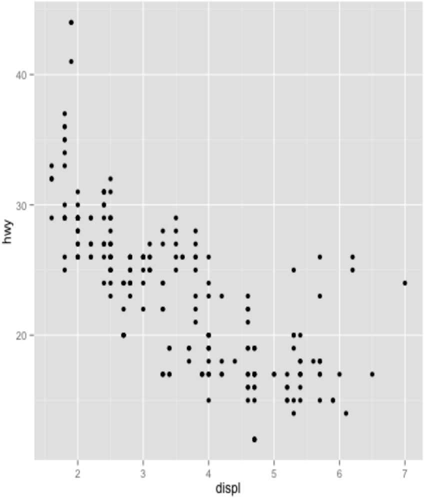
Tasks

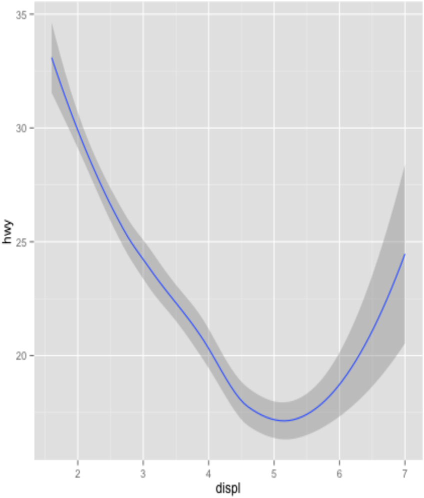
- Facet on two variables use the `facet_grid()` command. An example is the following:

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

  What do the empty cells mean?
- Look at `?facet_wrap`. What do `nrow` and `ncol` do? Why doesn't `facet_grid()` have `nrow` and `ncol` arguments?
- What happens if you facet on a continuous variable?

Geometric Objects



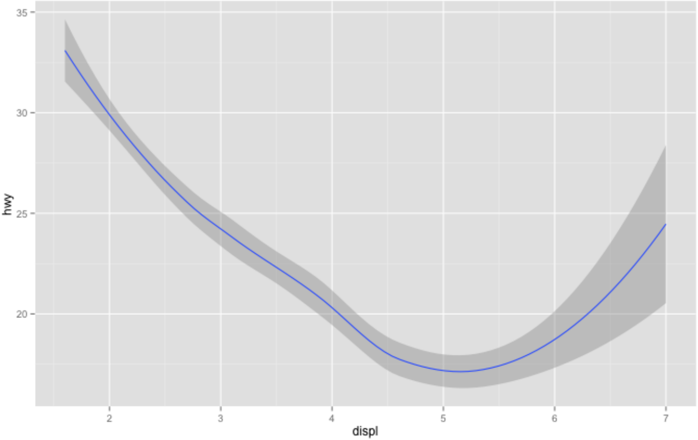


Geometric Objects

- In the previous slide, each plot used a different **visual object** to represent the data.
- Produce this by using different `geoms`.
- A `geom` is a geometrical object used to represent data in a plot.
- Often describe plots by the type of `geom` they use. For example, bar graphs use bar `geoms`.

Geometric Objects

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



Geometric Objects

- Every `geom` takes a `mapping` argument but not every aesthetic works with every `geom`.
  - E.g., you can set the `shape` of a point, but not a line. You can set the `linetype` of a line.
- `ggplot2` has around 30 different `geom`s.
- Can get help with `?geom_smooth`, for example.

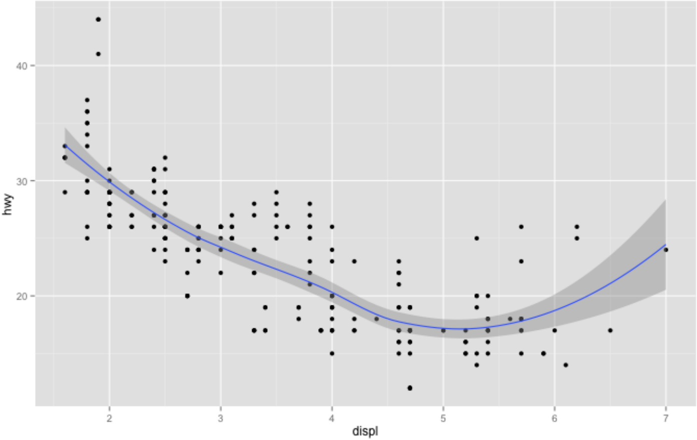
Geometric Objects

Some Commonly-used geoms

geom Name	Used to...	Aesthetics
geom_histogram	Visualize a Continuous Variable	x .
geom_bar	Visualize a Discrete Variable	x.
geom_point	Visualize a Two Continuous Variables	x, y.
geom_text	Add Labels to a Plot	x, y, label.
geom_boxplot	Visualize Continuous and Discrete Variables	x, y.
geom_jitter	Visualize a Two Variables	x, y.
many more ...		

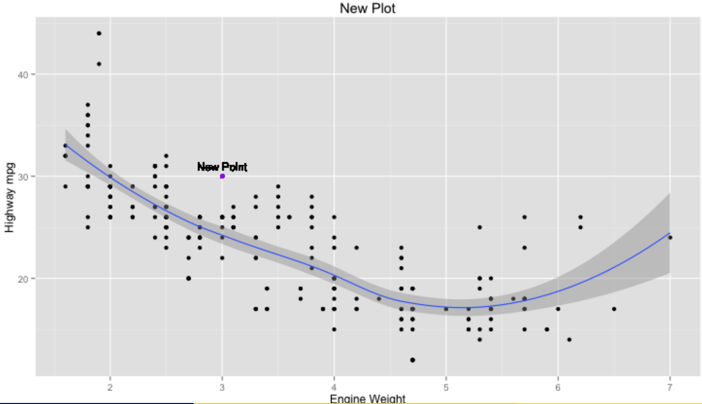
Layering geoms

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



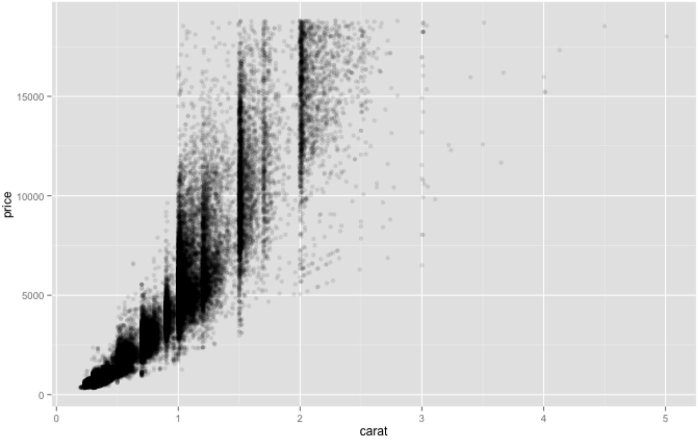
Adding Axis Labels

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(x=3, y=30), color = "purple") +
  geom_text(mapping = aes(x=3, y=31, label = "New Point"), size=4) -
  labs(title = "New Plot", x = "Engine Weight", y = "Highway mpg")
```



Layering geoms

```
> ggplot(data = diamonds) +
+   geom_point(mapping = aes(x = carat, y = price),
+              alpha = 1/10)
```



Check Yourself

Exercise:

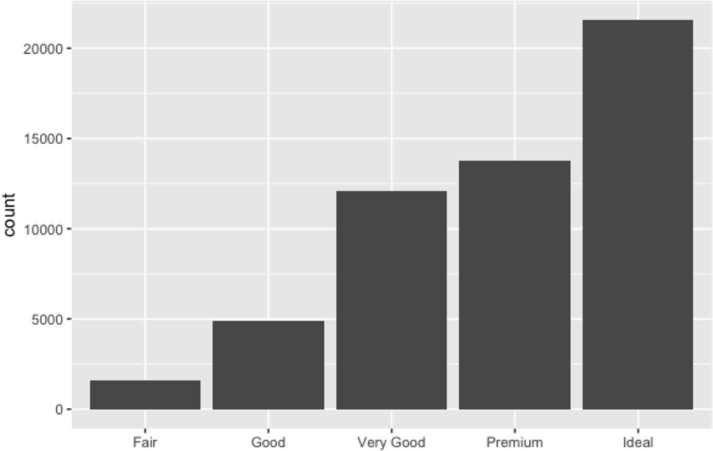
Use the built-in `iris` dataset.

- Plot iris `Sepal.Width` on the x-axis and `Sepal.Length` on the y-axis. Color the points according to whether the iris is a setosa or not.
- Plot two regression lines on the plot, one for the setosa iris and one for non-setosa iris. Hint: Use `geom_abline(intercept, slope)` or `geom_smooth()` with `method = "lm"`.

A few more examples

Barplot

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



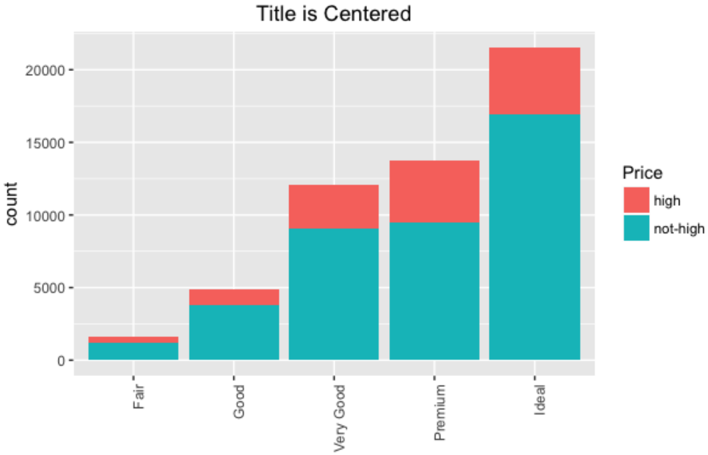
A few more examples

- Split `price` by the 75th percentile.
- Plot `cut` and `expensive` on one barchart.
- Center title
- Remove the word `cut` from xaxis label.

Barplot

```
> upper <- diamonds$price > quantile(diamonds$price,probs = .7
> diamonds$Expensive <- ifelse(upper,"high","not-high")
> theme_update(plot.title = element_text(hjust = 0.5))
> ggplot(data=diamonds) +
+   geom_bar(aes(x=cut,fill=factor(Expensive)))+
+   theme(axis.text.x = element_text(angle = 90, hjust = 1))+
+   labs(title = "Title is Centered",fill="Price",x="")
>
```

A few more examples

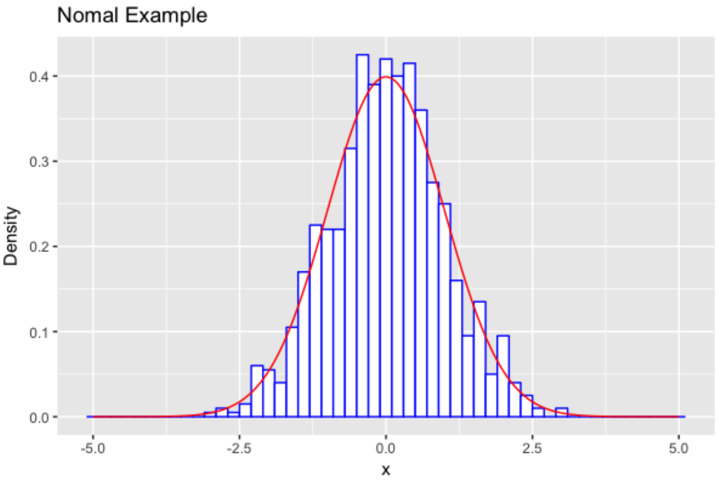


A few more examples

- Plot simulated standard normal and its density

```
> x <- seq(-5,5,by=.01)
> hist_data <- data.frame(x.var=rnorm(1000))
> plot_data <- data.frame(x=x,f=dnorm(x))
> ggplot(hist_data)+
+   geom_histogram(mapping=aes(x=x.var,y=..density..),
+                  col="blue",fill="white",binwidth=.2)+
+   geom_line(plot_data,mapping = aes(x = x, y = f),
+             col="red")+
+   labs(title = "Nomal Example",x="x",y="Density")
```

A few more examples: Plot simulated normal and its density

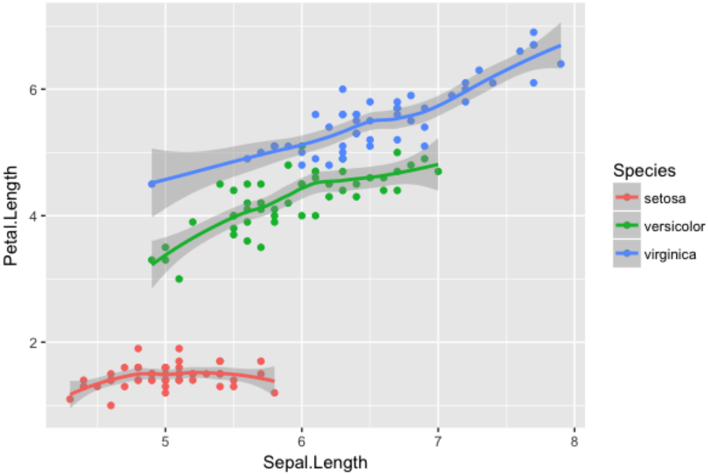


A few more examples: iris data

- Plot Petal.Length versus Sepal.Length split by Species
- Fit regression lines to each level of species.
- Match legend and colors accordingly.
- **A first attempt!**

```
> ggplot(data=iris)+
+   geom_point(mapping = aes(x=Sepal.Length,y=Petal.Length,
+                            color = Species))+
+   geom_smooth(mapping=aes(x=Sepal.Length,y=Petal.Length,
+                           color =Species))
```

A few more examples: iris data

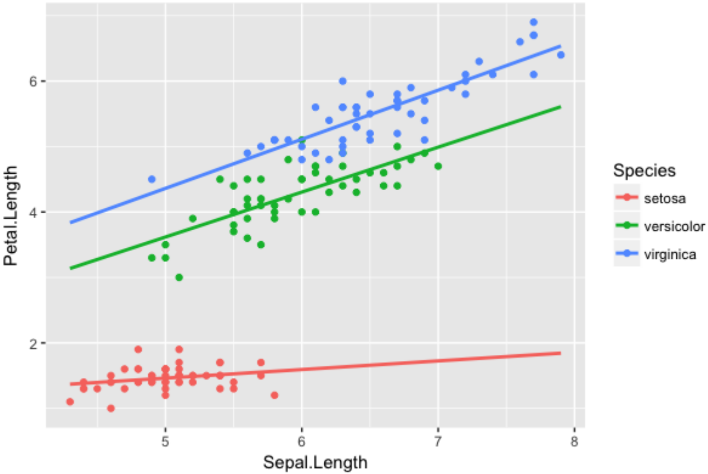


A few more examples: iris data

- Plot `Petal.Length` versus `Sepal.Length` split by `Species`
- Fit regression lines to each level of species.
- Match legend and colors accordingly.

```
> ggplot(data=iris)+
+   geom_point(mapping = aes(x=Sepal.Length,y=Petal.Length,
+                                color = Species))+
+   geom_smooth(mapping=aes(x=Sepal.Length,y=Petal.Length,
+                               color =Species),
+               method=lm,se=FALSE,fullrange=TRUE)
```

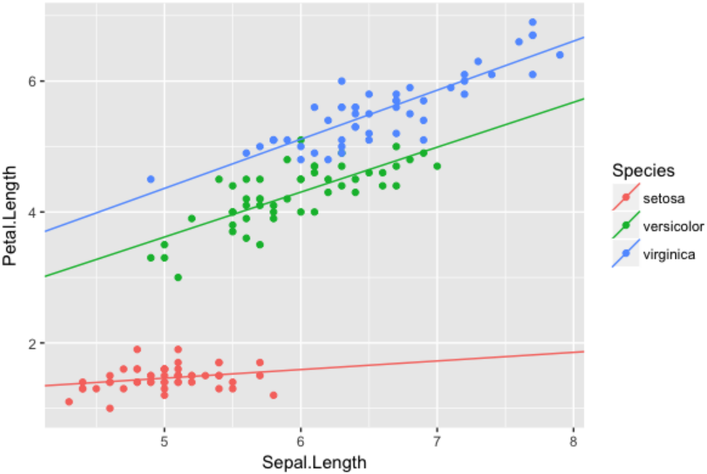
A few more examples: iris data



iris data: a different approach!

```
> # Define function that extracts intercept and slope
> slopes <- function(df) {
+   return(coef(lm(Petal.Length~Sepal.Length,data=df)))
+ }
> # Define dataframe of the slopes
> line_data <- data.frame(t(sapply(split(iris,iris$Species),
+                                    slopes)),
+                           Species=levels(iris$Species))
> # ggplot
> ggplot(data=iris)+
+   geom_point(mapping = aes(x=Sepal.Length,y=Petal.Length,
+                            colour = Species))+
+   geom_abline(data=line_data,aes(slope=Sepal.Length,
+                                  intercept=X.Intercept.,
+                                  colour= Species))
```

A few more examples: iris data



A few more examples: multiple time series graph

Data description

- We consider a data set containing information about the world's richest people. The data set us taken form the World Top Incomes Database (WTID) hosted by the Paris School of Economics [http://topincomes.g-mond.parisschoolofeconomics.eu]. This is derived from income tax reports, and compiles information about the very highest incomes in various countries over time, trying as hard as possible to produce numbers that are comparable across time and space.
- Open the file and make a new variable (dataframe) containing only the year, "P99", "P99.5" and "P99.9" variables; these are the income levels which put someone at the 99th, 99.5th, and 99.9th, percentile of income.

A few more examples: multiple time series graph

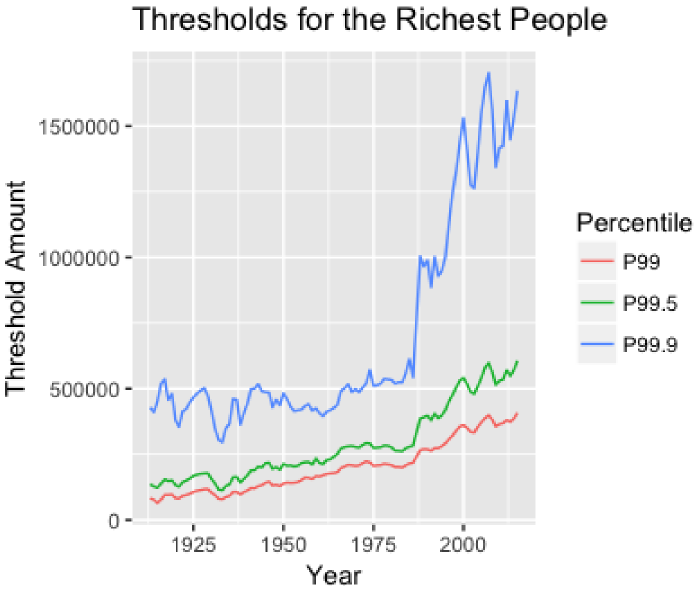
Data description

```
> wtid <- read.csv("wtid-report.csv", as.is = TRUE)
> wtid <- wtid[, c("Year", "P99.income.threshold","P99.5.incom
> names(wtid) <- c("Year", "P99", "P99.5", "P99.9")
> head(wtid)

  Year      P99   P99.5    P99.9
1 1913 82677.22 135583.5 428630.4
2 1914 76405.62 126910.5 410528.7
3 1915 64409.44 122555.7 451668.3
4 1916 77289.78 138102.3 518327.4
5 1917 95326.69 154537.8 536356.5
6 1918 95202.66 147850.1 457045.0
```

A few more examples: multiple time series graph

```
> n <- length(wtid$Year)
> wtid.new <- data.frame(Year=rep(wtid$Year,3),
+                        IncomeLevels=c(wtid$P99,
+                                       wtid$P99.5,
+                                       wtid$P99.9),
+                        Percentile=c(rep("P99",n),
+                                     rep("P99.5",n),
+                                     rep("P99.9",n))
+                         )
> ggplot(data = wtid.new) +
+   geom_line(mapping = aes(x = Year, y = IncomeLevels,
+                            color=Percentile))+
+   labs(title = "Thresholds for the Richest People",
+        x = "Year", y = "Threshold Amount",
+        color="Percentile")
```

A few more examples: iris data



Closing remarks

Base R versus ggplot

- ggplot is *arguably* easier to use than base R.
- The grammar used in ggplot synthesizes with the tidyverse.
- Many professionals prefer ggplot over base R graphics.
- Many professionals prefer base R graphics over ggplot.
- If you know the intricate details of base R graphics, you can construct very beautiful graphs.