

# Interactive Data Visualization

David Li

# The goals of today

- The key ideas behind ggplot2
- The basic steps of data visualization
- Several important plots
- Learn the popular cross-language cross-platform plotting package plotly
- Gain insight and practical skills for creating interactive and dynamic web graphics for data analysis

# Basic Plotting: ggplot2

- Advantages of ggplot2
  - consistent underlying grammar of graphics (Wilkinson, 2005)
  - plot specification at a high level of abstraction
  - very flexible
  - theme system for polishing plot appearance
  - mature and complete graphics system
  - many users, active mailing list

# Basic Plotting: ggplot2

- Some things you cannot (or should not) do with ggplot2:
  - 3-dimensional graphics (see the rgl package)
  - Graph-theory type graphs (nodes/edges layout, Spark GraphX)
  - Interactive graphics (We will learn plotly today)

# Grammar Of Graphics in ggplot2

- Building blocks of a graph include:
  - data
  - aesthetic mapping
  - geometric object
  - statistical transformations
  - scales
  - coordinate system
  - position adjustments
  - faceting

# Setup: install the tidyverse package

- Why tidyverse? Because this package includes a popular collection of packages

- ggplot2, for data visualisation.
- dplyr, for data manipulation.
- tidyr, for data tidying.
- readr, for data import.
- purrr, for functional programming.
- tibble, for tibbles, a modern re-imagining of data frames.

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

```
installing the source packages 'tinytex', 'modelr'
```

```
-- Attaching packages ----- tidyverse 1.3.0 --
```

```
v ggplot2 3.2.1      v purrr  0.3.3  
v tibble  2.1.3      v dplyr  0.8.4  
v tidyr   1.0.2      v stringr 1.4.0  
v readr   1.3.1      v forcats 0.4.0
```

# The first example

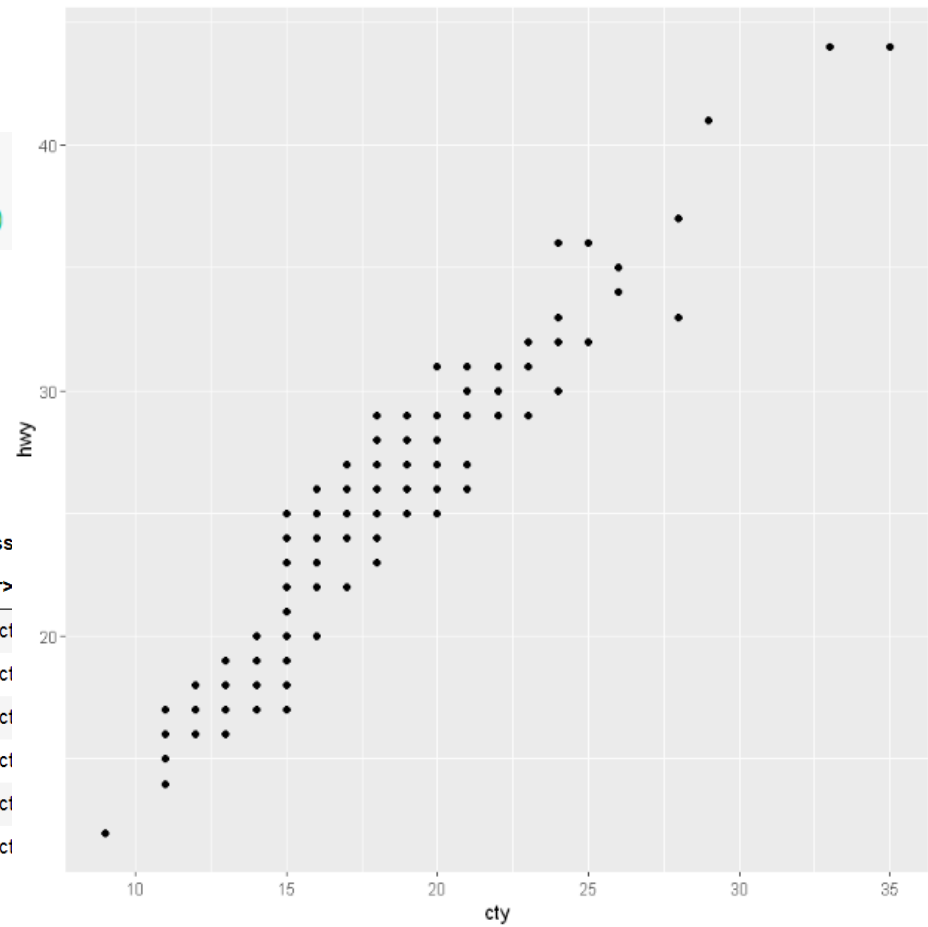
Three key components of a graph:  
data, aesthetics, geometries

```
head(mpg)
ggplot(mpg, aes(cty, hwy)) + geom_point()
```

The full version is

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

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact



# Types of Statistical Data

Most data fall into one of two groups: numerical or categorical

- Numerical data

- *Discrete data*

- represent items that can be counted;
    - they take on possible values that can be listed out.
    - The list of possible values may be fixed (also called *finite*);
    - or it may go from 0, 1, 2, on to infinity (making it *countably infinite*).

- *Continuous data*

- represent measurements;
    - their possible values cannot be counted and can only be described using intervals on the real number line

- Categorical data

- represent characteristics such as a person's gender, marital status, hometown
  - can take on numerical values (such as "1" indicating male and "2" indicating female), but those numbers don't have mathematical meaning



# Most basic barplot

- Always start by calling the `ggplot()` function.
- Then specify the `data` object. It has to be a data frame. And it needs one numeric and one categorical variable.
- Then aesthetics, set in the `aes()` function: set the categorical variable for the X axis, use the numeric for the Y axis
- Finally call `geom_bar()`. You have to specify `stat="identity"` for this kind of dataset, which is sum of y for each category of x and is the height of bar.

```
data=data.frame(  
  name=c("A", "A", "C", "C", "E") ,  
  value=c(3, 17, 4, 6, 40)  
)
```

```
ggplot(data, aes(x=name, y=value)) + geom_bar(stat = "identity")
```

↑  
data

↑  
aesthetics

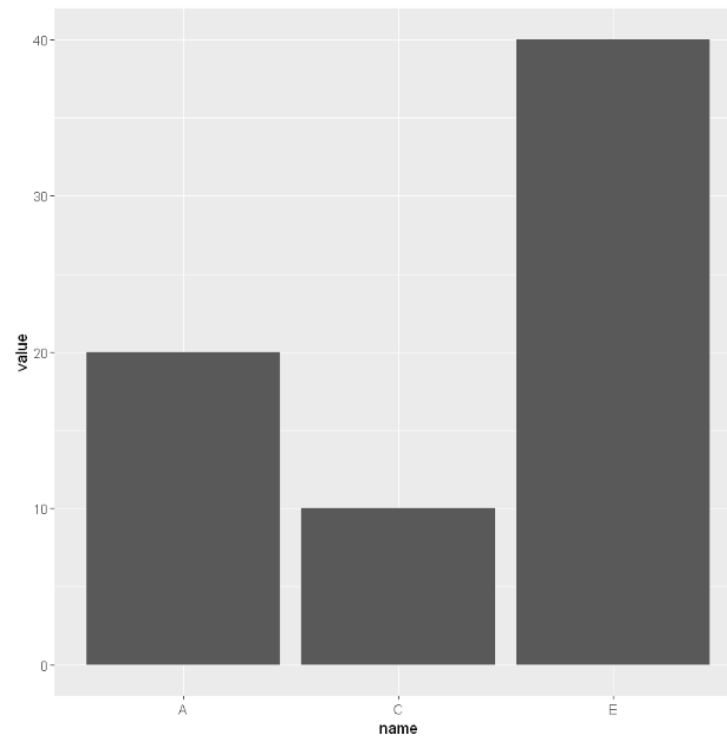
↑  
geometries

↑  
statistical transformations

# Most basic barplot

```
data=data.frame(  
  name=c("A", "A", "C", "C", "E") ,  
  value=c(3, 17, 4, 6, 40)  
)  
  
ggplot(data, aes(x=name, y=value)) + geom_bar(stat = "identity")
```

name	value
<fct>	<dbl>
A	3
A	17
C	4
C	6
E	40



# Basic barplot without y variable

- Start by calling the `ggplot()` function.
- Then specify the `data` object. It has to be a data frame. And it needs one categorical variable.
- Then aesthetics, set in the `aes()` function: set the categorical variable for the x axis
- Finally call `geom_bar()`. Because you don't have y, so you can't specify `stat="identity"`. The default is `stat="count"`, which is count of rows for each category of x and is the height of bar.

```
data=data.frame(name=c("A", "A", "C", "C", "E"))  
  
ggplot(data, aes(x=name)) + geom_bar()  
ggplot(data, aes(x=name)) + geom_bar(stat="count")
```



data



aesthetics



geometries

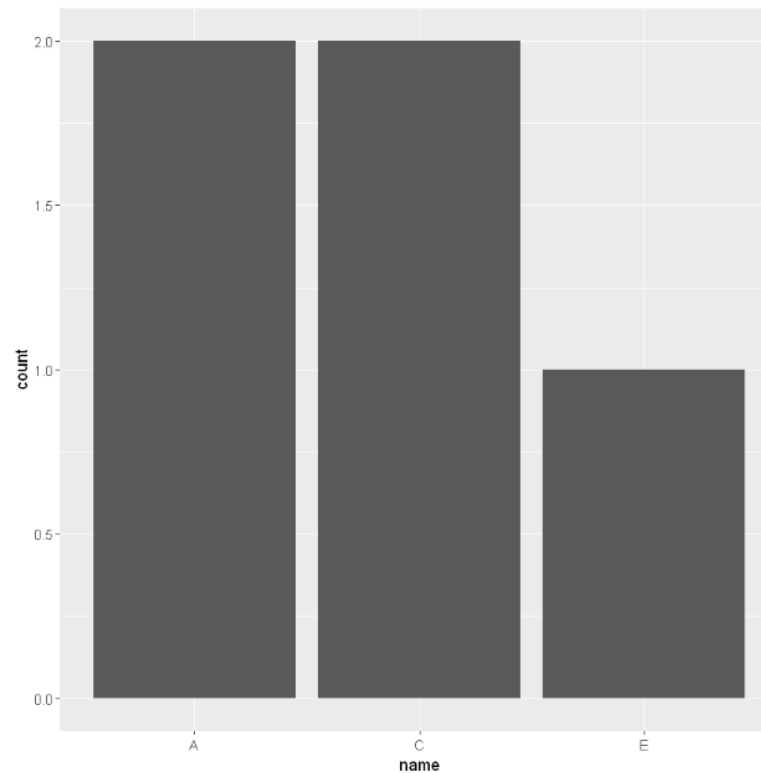


statistical transformations

# Basic barplot without y variable

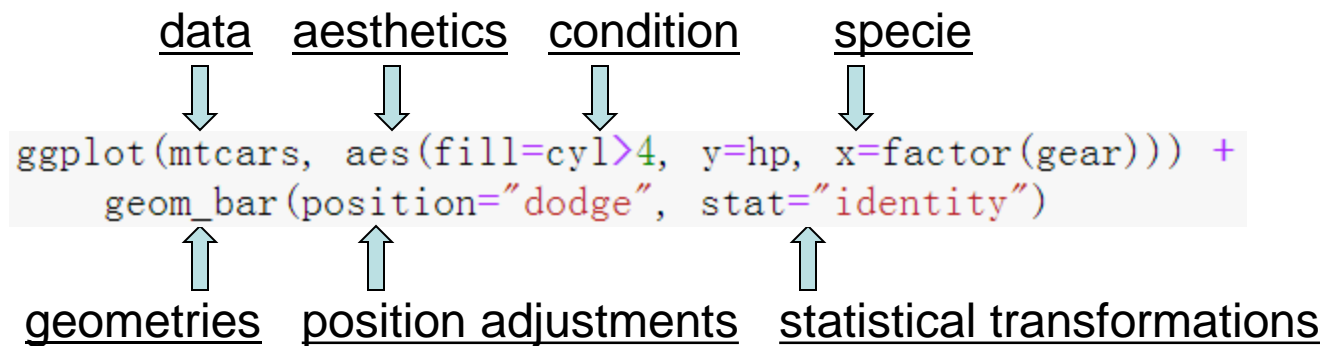
```
data=data.frame(name=c("A", "A", "C", "C", "E"))  
  
ggplot(data, aes(x=name)) + geom_bar()  
ggplot(data, aes(x=name)) + geom_bar(stat="count")
```

name
<fct>
A
A
C
C
E



# Grouped barplot

- A grouped barplot display a numeric value for a set of entities split in groups and subgroups.
- Data must have 3 columns: the numeric value (**value**), and 2 categorical variables for the group (**specie**) and the subgroup (**condition**) levels.
- In **aes()**, is the group (**specie**), and the subgroup (**condition**) is given to the **fill** argument.
- In **geom\_bar()**, **position="dodge"** must be specified to have the bars one beside each other. Here we specify **stat="identity"** to use y values as bar heights.

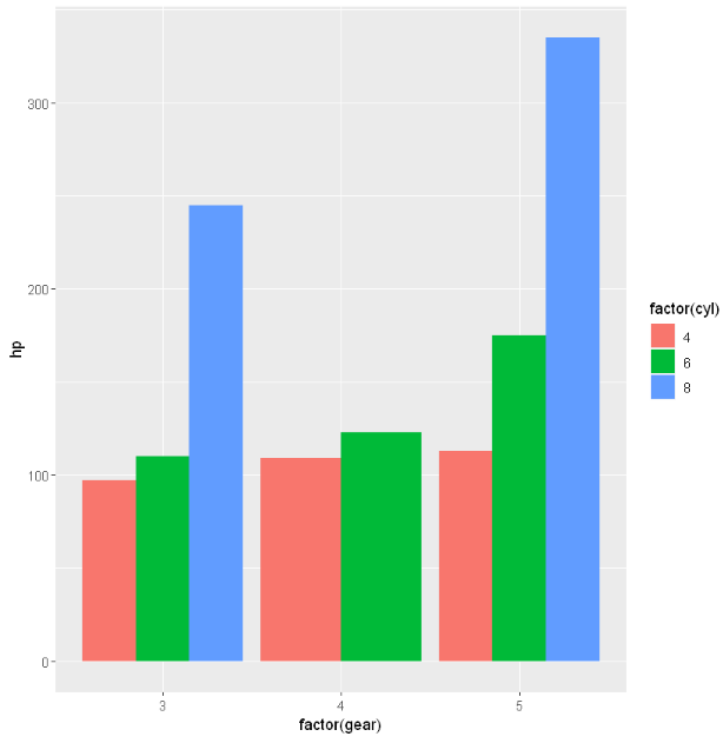
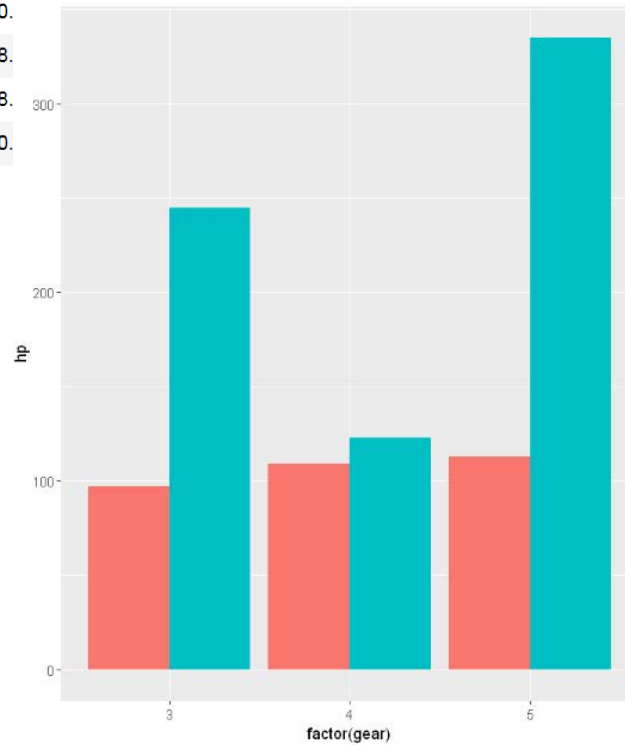


# Grouped barplot examples

```
ggplot(mtcars, aes(fill=cyl>4, y=hp, x=factor(gear))) +  
  geom_bar(position="dodge", stat="identity")  
  
ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) +  
  geom_bar(position="dodge", stat="identity")
```

factor() converts discrete to categorical

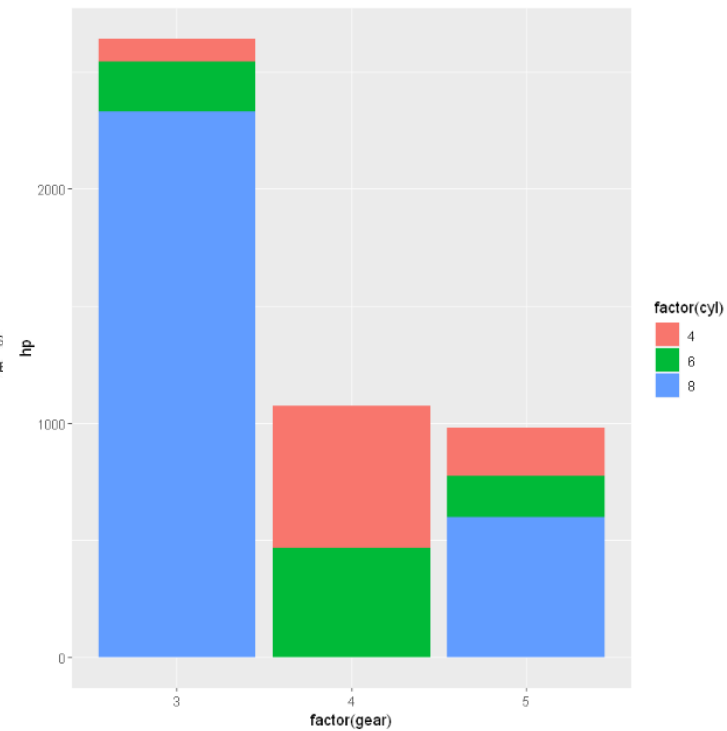
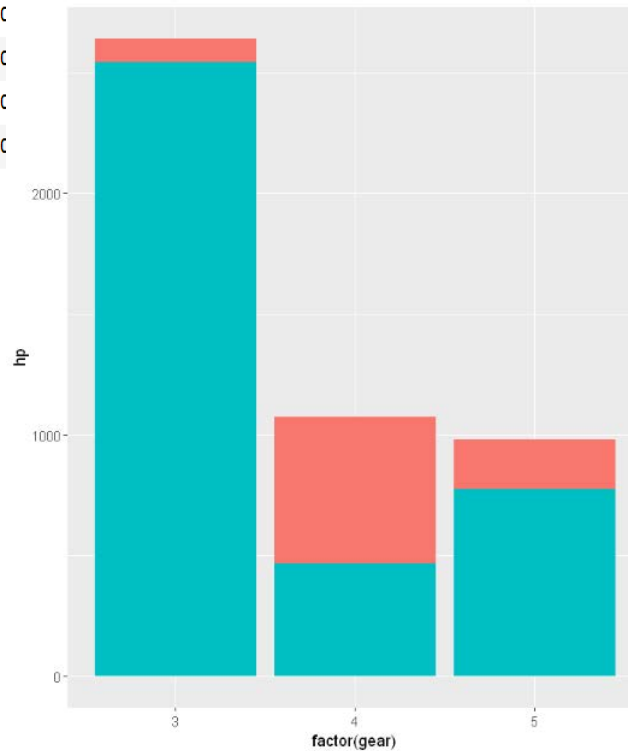
	mpg	cyl	displacement	horsepower	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.								
Datsun 710	22.8	4	108.								
Hornet 4 Drive	21.4	6	258.								
Hornet Sportabout	18.7	8	360.								



# Stacked barplot examples

```
ggplot(mtcars, aes(fill=cyl>4, y=hp, x=factor(gear))) +  
  geom_bar(position="stack", stat="identity")  
ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) +  
  geom_bar(position="stack", stat="identity")
```

	mpg	cyl	displacement	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	16.99	1	0	3	1
Hornet 4 Drive	21.4	6	258.0	151	3.70	3.440	17.02	0	1	4	4
Hornet Sportabout	18.7	8	360.0	228	3.70	5.250	17.05	0	1	4	4

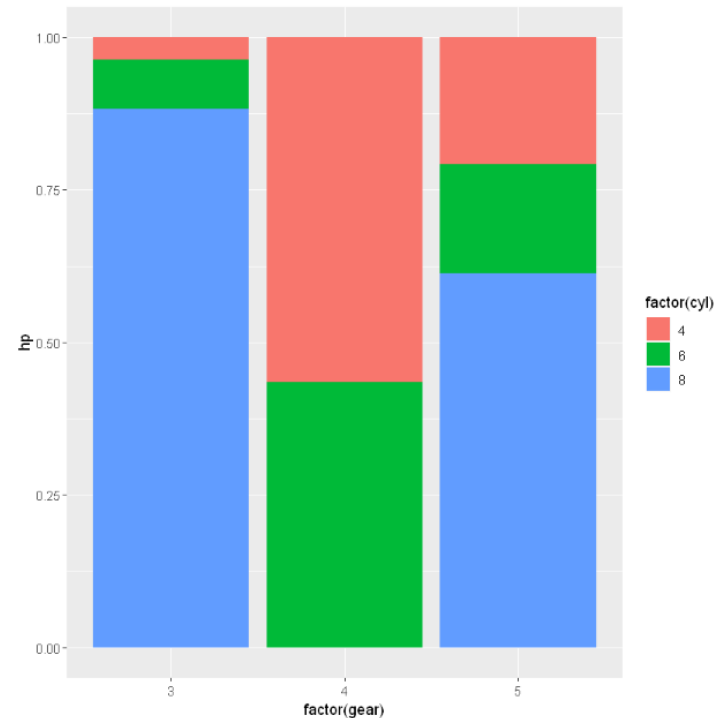
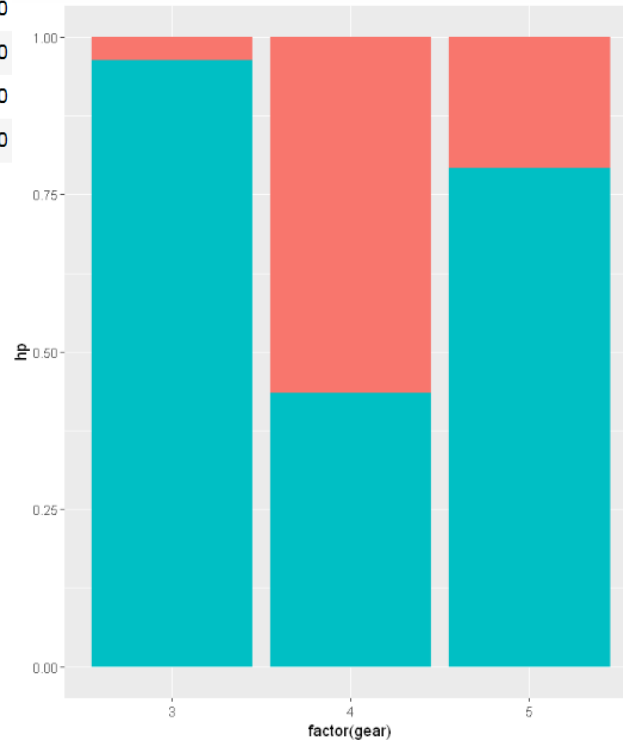


# Percent Stacked barplot examples

```
ggplot(mtcars, aes(fill=cyl>4, y=hp, x=factor(gear))) +  
  geom_bar(position="fill", stat="identity")  
ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) +  
  geom_bar(position="fill", stat="identity")
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>

Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0								
Datsun 710	22.8	4	108.0								
Hornet 4 Drive	21.4	6	258.0								
Hornet Sportabout	18.7	8	360.0								





# Small multiple barplot

- Small multiple can be used as an alternative of stacking or grouping
- Use `facet_wrap()` to specify the first level grouping

The diagram illustrates the components of a ggplot2 plot and how they map to a specific code snippet. The code snippet is: `ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) + geom_bar(position="dodge", stat="identity") + facet_wrap(~mpg>20)`. Arrows point from the following labels to the corresponding parts of the code: 

- data points to `mtcars`
- aesthetics points to `aes(fill=factor(cyl), y=hp, x=factor(gear))`
- condition points to `geom_bar`
- specie points to `facet_wrap`
- geometries points to `geom_bar`
- position points to `position="dodge"`
- statistical transformations points to `stat="identity"`
- faceting points to `facet_wrap`

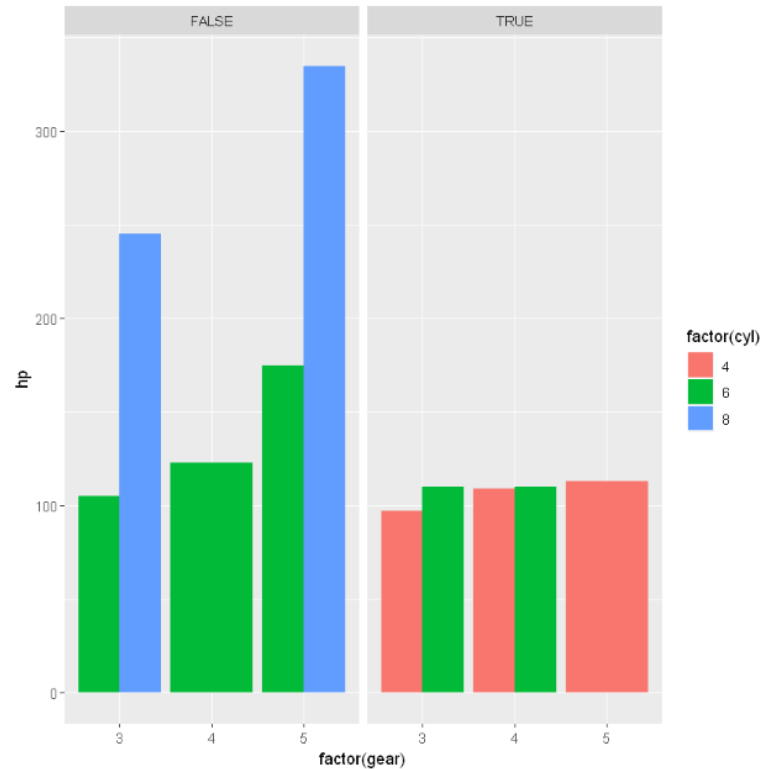
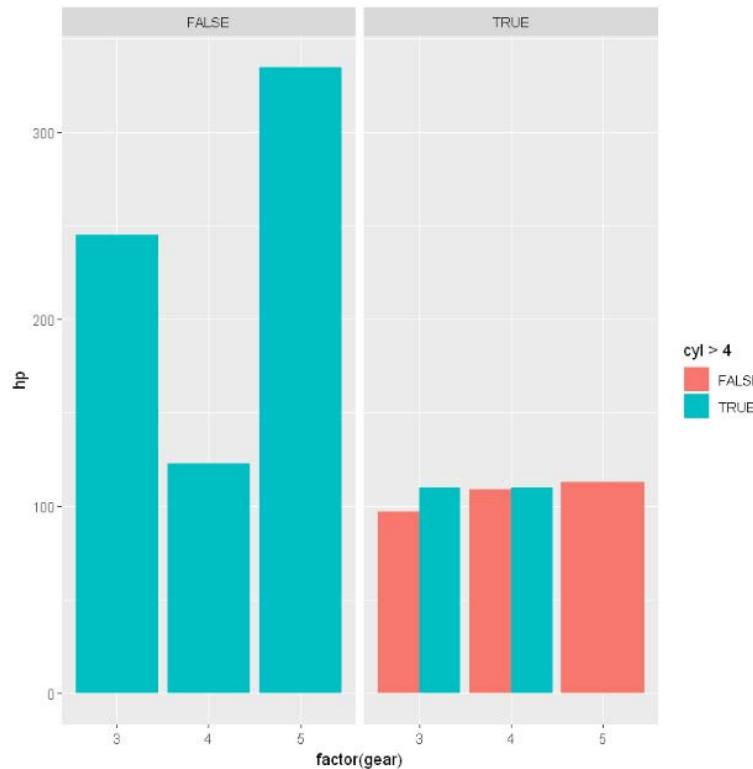
```
ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) +  
  geom_bar(position="dodge", stat="identity") + facet_wrap(~mpg>20)
```

# Small multiple barplot examples

```
ggplot(mtcars, aes(fill=cyl>4, y=hp, x=factor(gear))) +  
  geom_bar(position="dodge", stat="identity") + facet_wrap(~mpg>20)  
ggplot(mtcars, aes(fill=factor(cyl), y=hp, x=factor(gear))) +  
  geom_bar(position="dodge", stat="identity") + facet_wrap(~mpg>20)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>

Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6									
Datsun 710	22.8	4									
Hornet 4 Drive	21.4	6									
Hornet Sportabout	18.7	8									

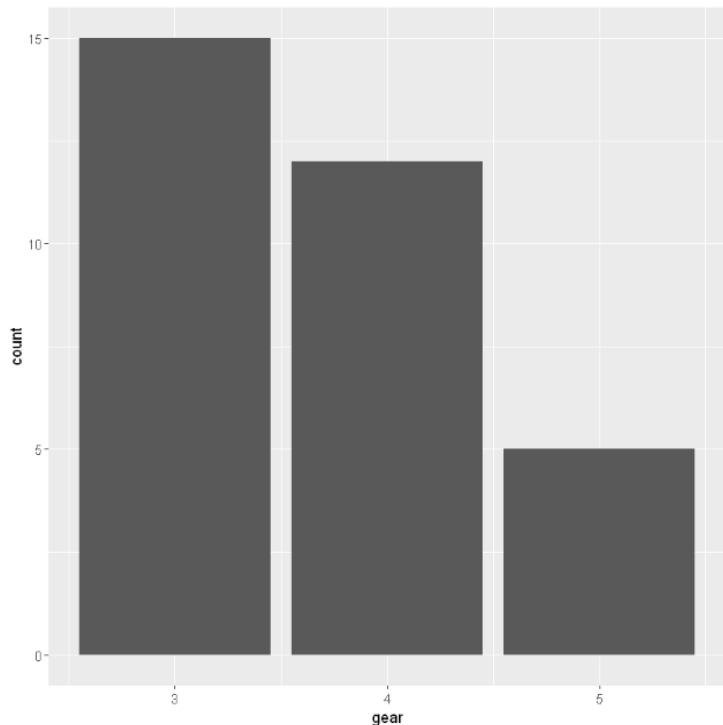


# Difference between bar and histogram

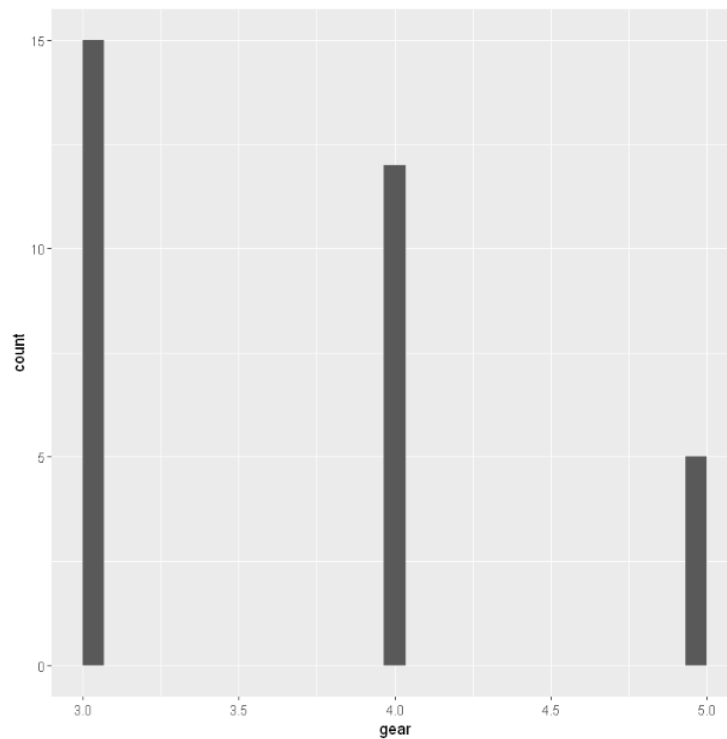
Count the rows for each value of “gear”

- To quickly see a distribution

```
ggplot(mtcars, aes(gear)) + geom_bar()  
ggplot(mtcars, aes(gear)) + geom_histogram()
```



	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

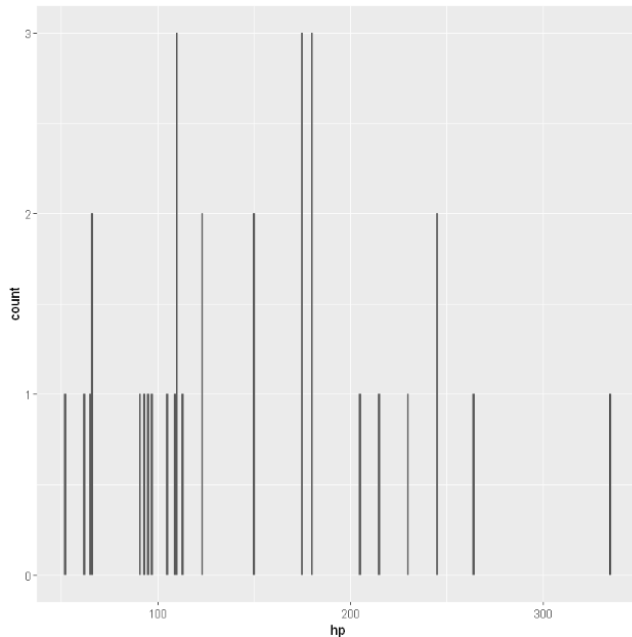


# Difference between bar and histogram

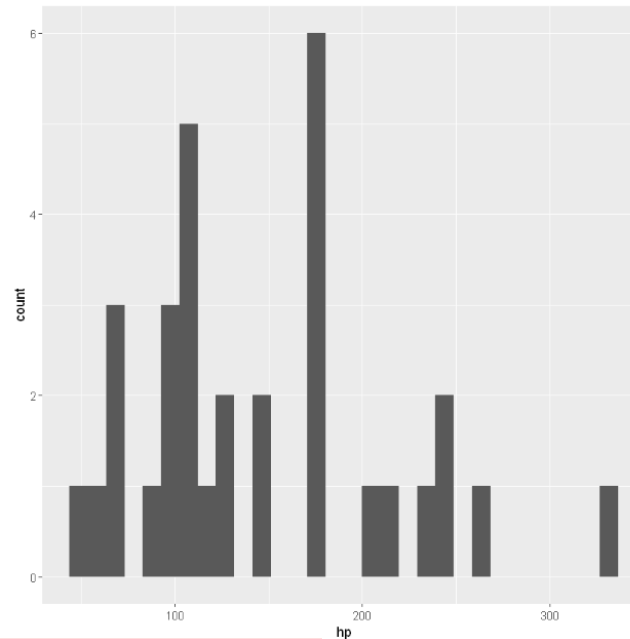
## Difference between bar and histogram:

- Bar shows each individual x values
- Histogram splits the range of x values into bins

```
ggplot(mtcars, aes(hp)) + geom_bar()  
ggplot(mtcars, aes(hp)) + geom_histogram()
```



	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1



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

# Line Chart

- A line chart or line graph displays the evolution of one or several numeric variables.
- Data points are usually connected by straight line segments.
- Start by calling the **ggplot()** function.
- Then specify the **data** object. The input data frame requires at least 2 columns:
  - An **ordered** numeric variable for the X axis
  - Another numeric variable for the Y axis
- Then aesthetics, set x and y in the **aes()** function
- Finally call **geom\_line()**.

```
data=data.frame(xValue=1:3, yValue=4:6)  
ggplot(data, aes(x=xValue, y=yValue)) + geom_line()
```



data



aesthetics

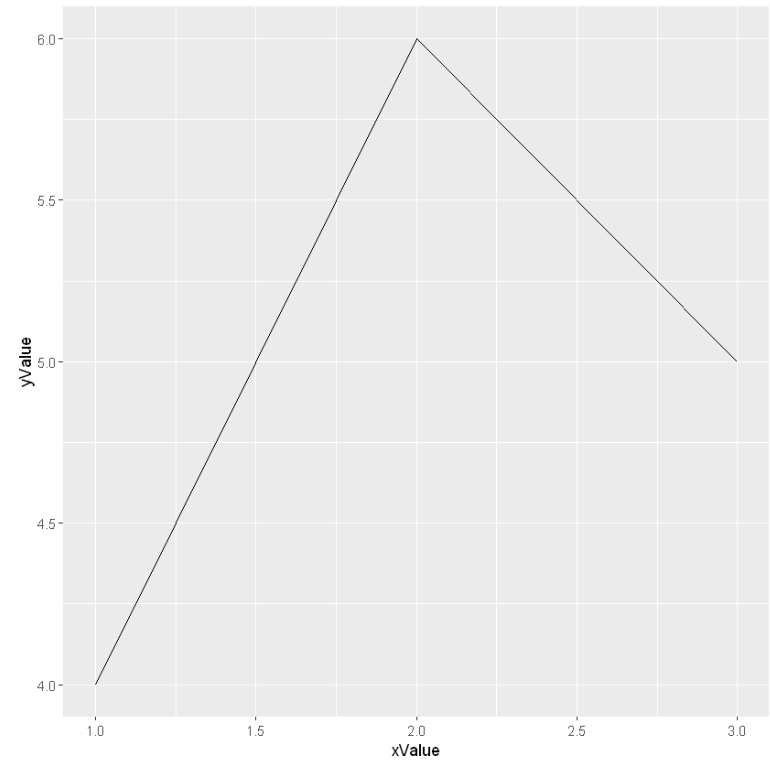
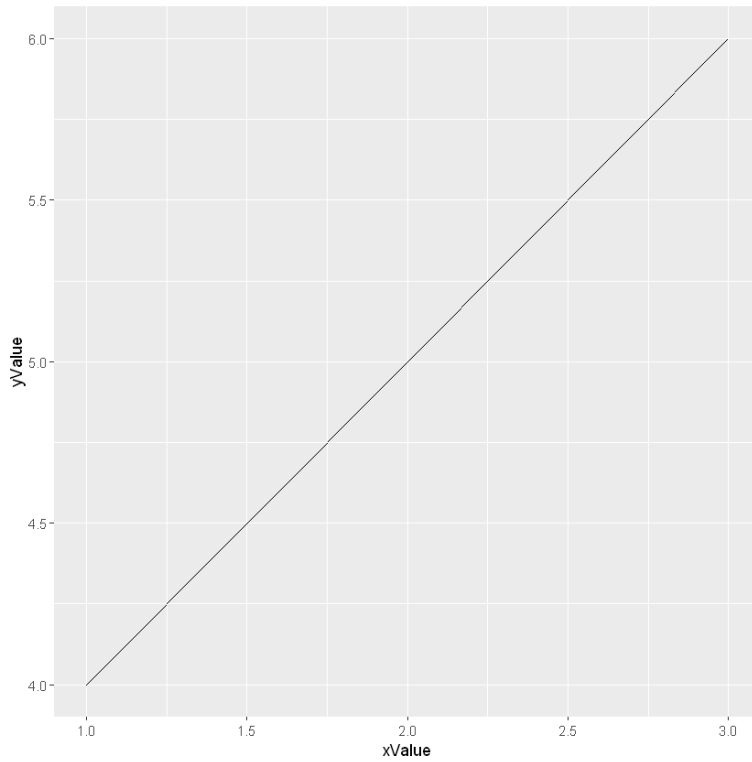


geometries

# Line Chart examples

If x is not sorted, R will sort it before connecting points

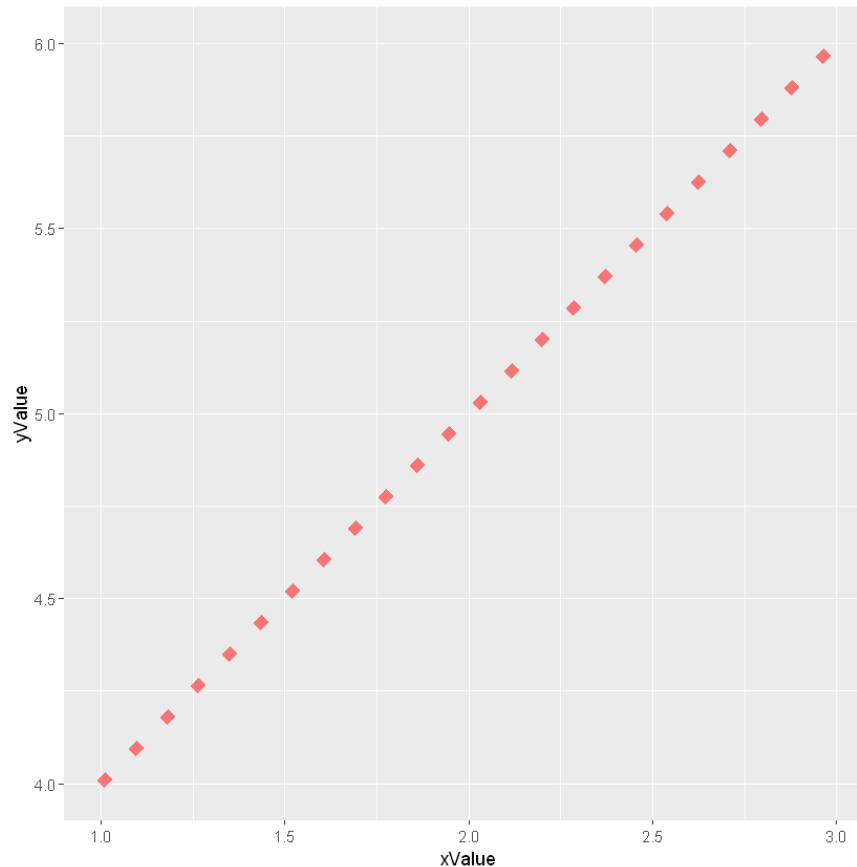
```
data=data.frame(xValue=1:3, yValue=4:6)
ggplot(data, aes(x=xValue, y=yValue)) + geom_line()
data=data.frame(xValue=c(1,3,2), yValue=4:6)
ggplot(data, aes(x=xValue, y=yValue)) + geom_line()
```



# Line Chart examples

Line styles can be added in `geom_line()`

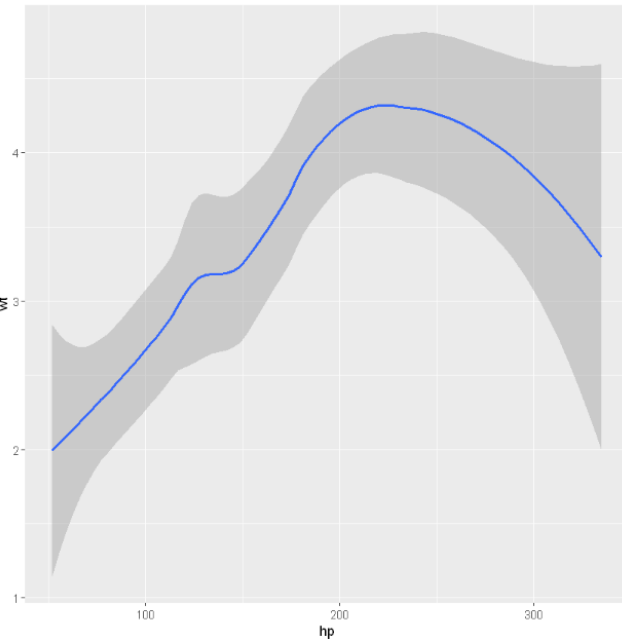
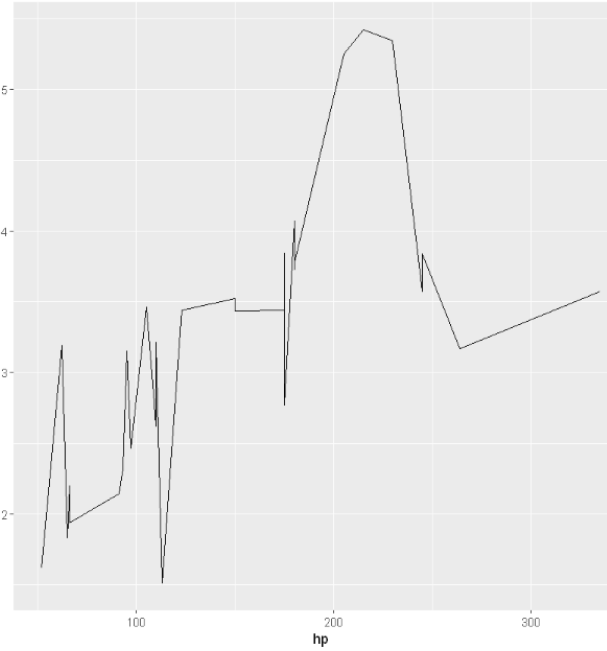
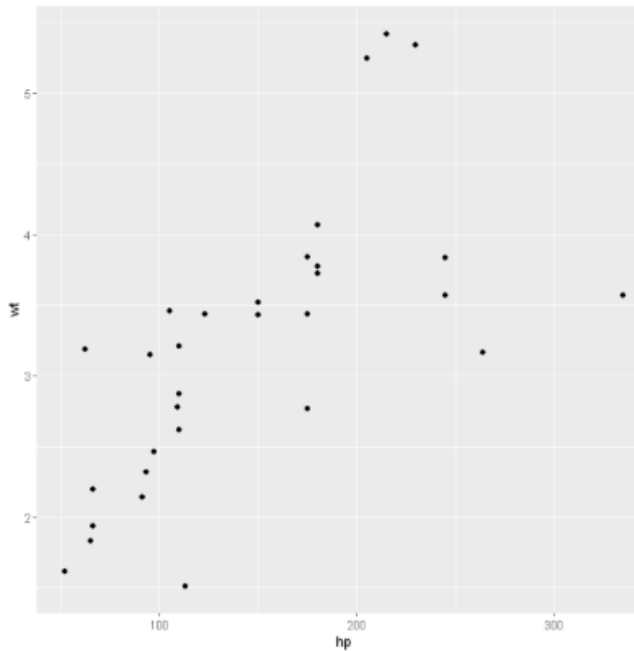
```
data=data.frame(xValue=1:3, yValue=4:6)  
ggplot(data, aes(x=xValue, y=yValue)) + geom_line(color="red", size=3, alpha=0.5, linetype=3)
```



# More examples of plotting (x, y)

```
ggplot(mtcars, aes(hp, wt)) + geom_point()
ggplot(mtcars, aes(hp, wt)) + geom_line()
ggplot(mtcars, aes(hp, wt)) + geom_smooth()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
<b>Mazda RX4</b>	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
<b>Mazda RX4 Wag</b>	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
<b>Datsun 710</b>	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
<b>Hornet 4 Drive</b>	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
<b>Hornet Sportabout</b>	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
<b>Valiant</b>	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1





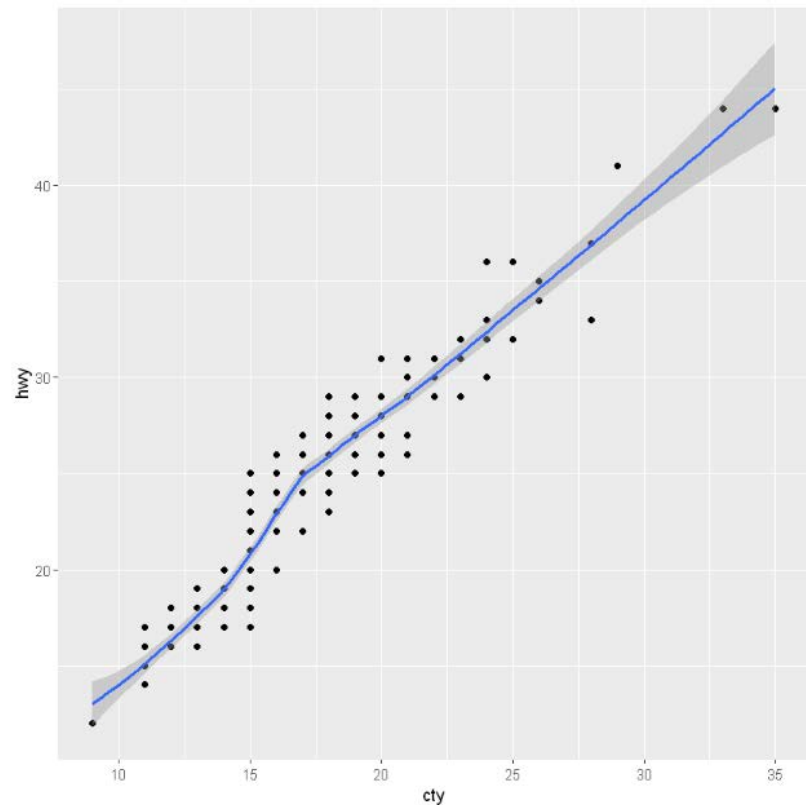
# Overlap plotting

We can add more geometries so that the plots will be on one graph

```
ggplot(DATAFRAME, aes(X_COLUMN, Y_COLUMN)) + GEOM_TYPE1 + GEOM_TYPE2
```

Example,

```
ggplot(mpg, aes(cty, hwy))  
+ geom_point()  
+ geom_smooth()
```



# Line Chart for Time Series

- The ggplot2 package recognizes the date format and automatically uses a specific type of X axis.
  - If the time variable isn't at the date format, this won't work.
  - Check with `str(data)` how variables are understood by R.
  - If not read as a date, use `lubridate` or `anytime` to convert it
- Then the rest is the same as line chart

```
library(anytime)
IBM=read.csv('IBM.csv')
IBM$Date = anytime(IBM$Date)
ggplot(IBM, aes(x=Date, y=Adj.Close)) + geom_line() +
  scale_x_date(limit=c(as.Date("2018-01-01"), as.Date("2018-02-11")))
```

scales

data

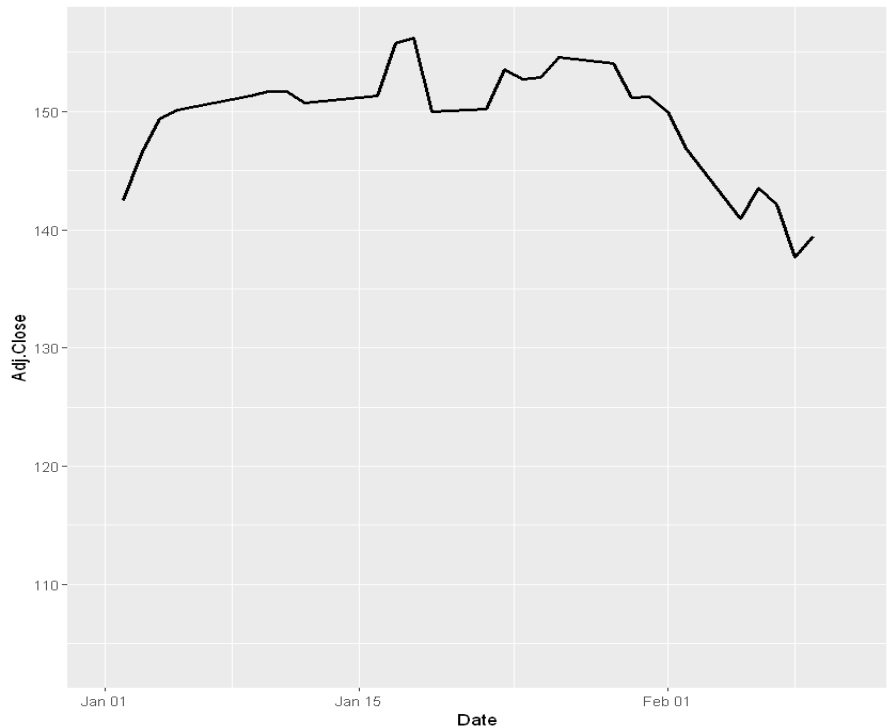
aesthetics

geometries

# Time Series examples

Assume time series has sorted date column

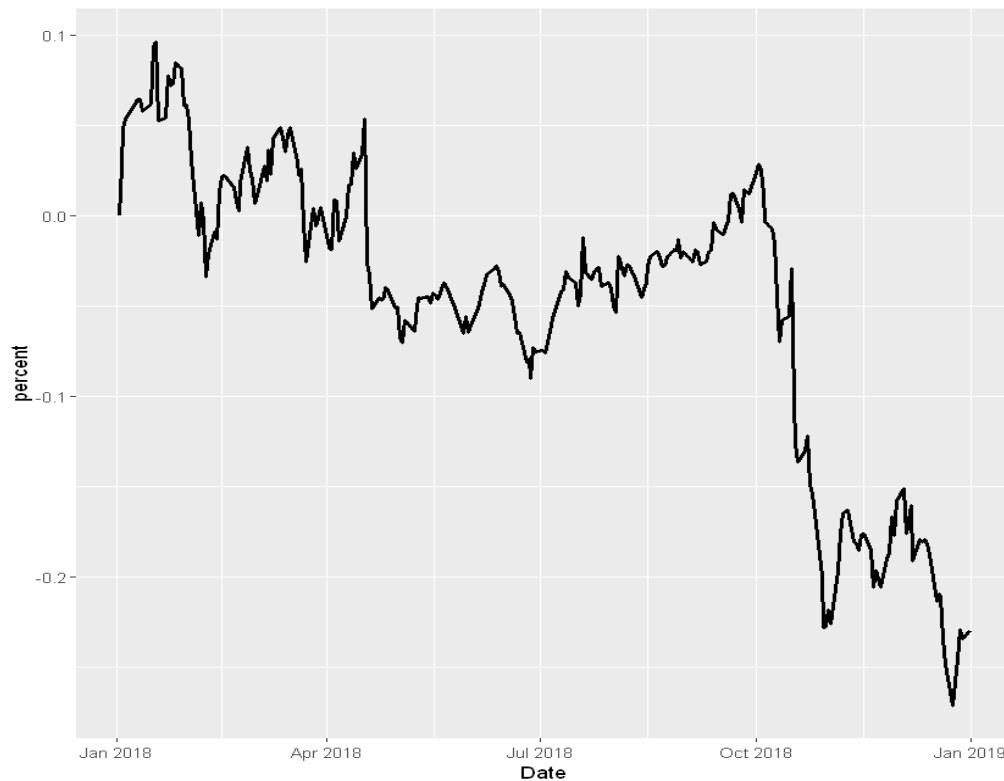
```
library(anytime)
IBM=read.csv('IBM.csv')
IBM$Date = anytime(IBM$Date)
ggplot(IBM, aes(x=Date, y=Adj.Close)) + geom_line(size=1)
ggplot(IBM, aes(x=Date, y=Adj.Close)) + geom_line(size=1) +
  scale_x_date(limit=c(as.Date("2018-01-01"), as.Date("2018-02-11"))))
```



# Time Series examples

Convert y to the percent change from the first day

```
startprice=IBM[1, 6]  
IBM %>%  
mutate(percent=(Adj.Close-startprice)/startprice) %>%  
ggplot(aes(x=Date, y=percent)) + geom_line(size=1)
```



# Multi groups line chart for Time Series

- The input data frame is composed by 3 columns:
  - An **ordered** numeric variable for the X axis. For time series, it is the Date column.
  - Another numeric variable for the Y axis
  - A categorical variable that specify the group of the observation
- The idea is to draw one line per group

```
data.frame(Date=IBM$Date, IBM=IBM$Adj.Close, GOOG=GOOG$Adj.Close) %>%  
reshape2::melt(id.vars="Date") %>%  
ggplot(aes(x=Date, y=value, group=variable, color=variable)) + geom_line(size=1)
```

data

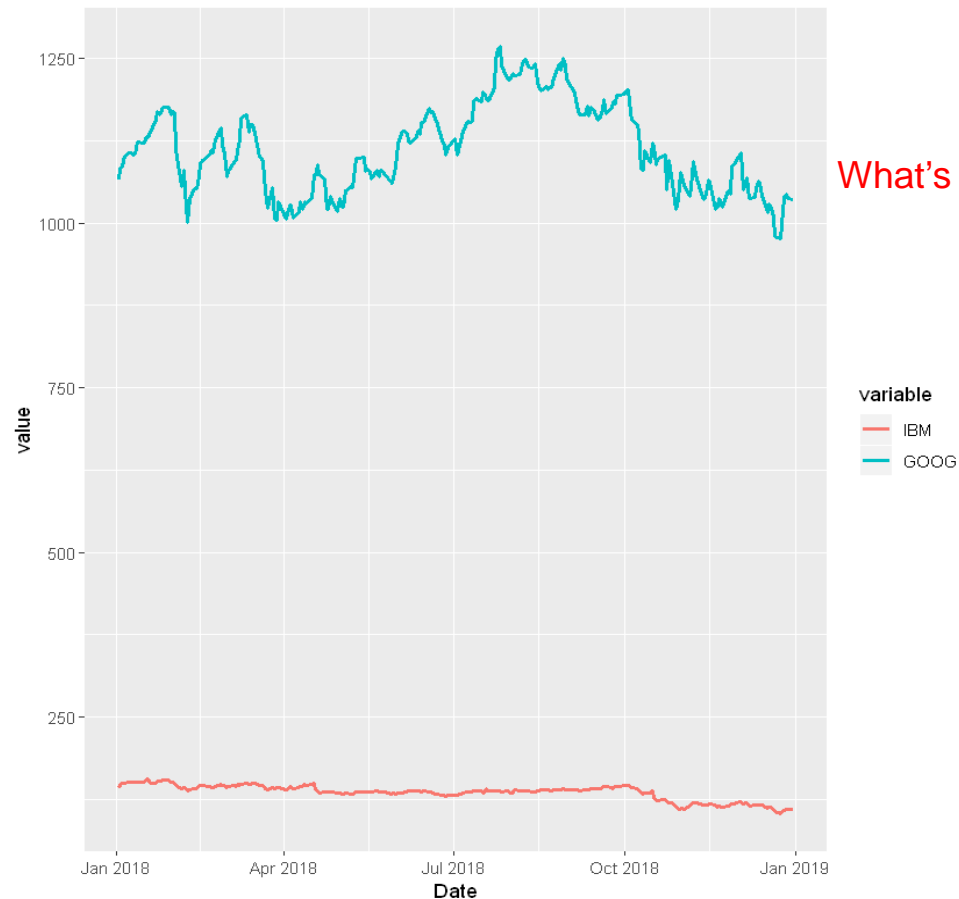
aesthetics

geometries

# Multiple Time Series examples

```
data.frame(Date=IBM$Date, IBM=IBM$Adj.Close, GOOG=GOOG$Adj.Close) %>%
  reshape2::melt(id.vars="Date") %>%
  ggplot(aes(x=Date, y=value, group=variable, color=variable)) + geom_line(size=1)
```

	Date	IBM	GOOG
	<date>	<dbl>	<dbl>
1	2018-01-02	142.4840	1065.00
2	2018-01-03	146.4006	1082.48
3	2018-01-04	149.3657	1086.40
4	2018-01-05	150.0954	1102.23
5	2018-01-08	151.0007	1106.94
6	2018-01-09	151.3332	1106.26
	Date	variable	value
	<date>	<fct>	<dbl>
	2018-01-02	IBM	142.4840
	2018-01-03	IBM	146.4006
	...	...	...
	2018-11-15	GOOG	1064.71
	2018-11-16	GOOG	1061.49



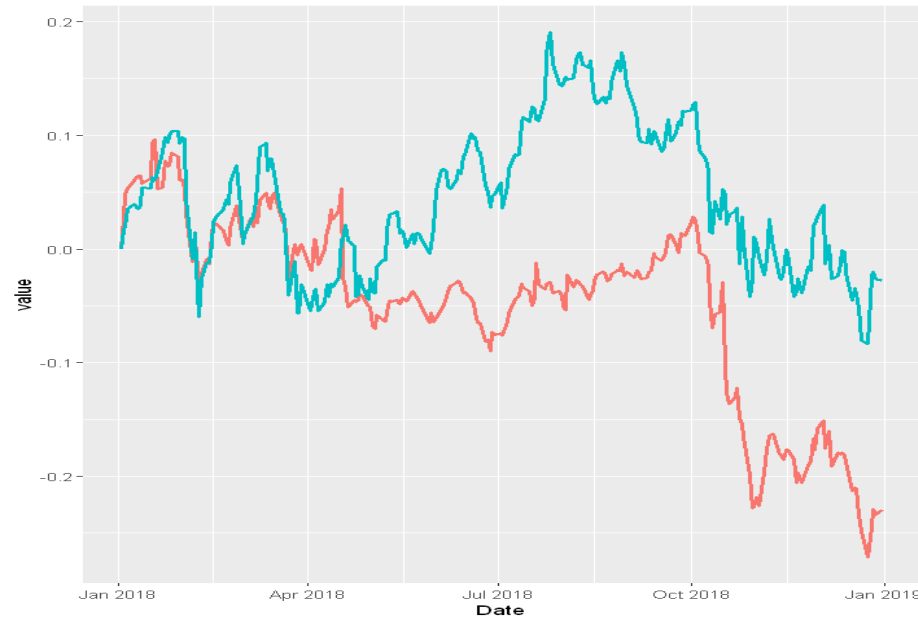
What's wrong with it?

# Multiple Time Series examples

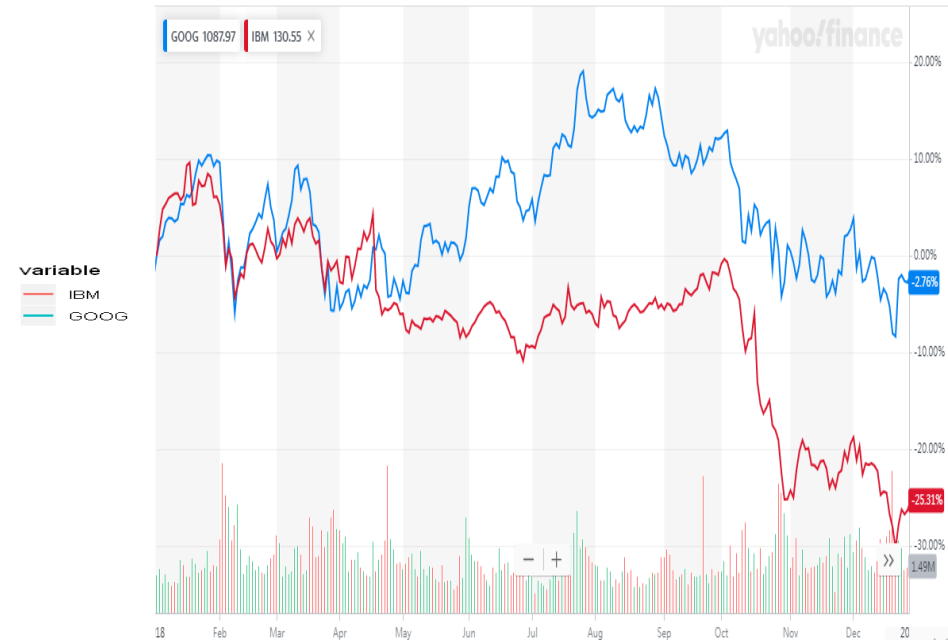
Use percent change when plotting multiple stocks

```
ibm_startprice=IBM[1, 6]
goog_startprice=GOOG[1, 6]
data.frame(Date=IBM$Date,
            IBM=transmute(IBM, IBM=(Adj.Close-ibm_startprice)/ibm_startprice),
            GOOG=transmute(GOOG, GOOG=(Adj.Close-goog_startprice)/goog_startprice)) %>%
reshape2::melt(id.vars="Date") %>%
ggplot(aes(x=Date, y=value, group=variable, color=variable)) + geom_line(size=1)
```

From R code



From Yahoo finance



# Interactive data visualization: plotly

- Plotly is helping to close the gap between data science and the business.
- Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST.
- Create interactive and dynamic web graphics for data analysis
- No need web technology (e.g., JavaScript, HTML, CSS)

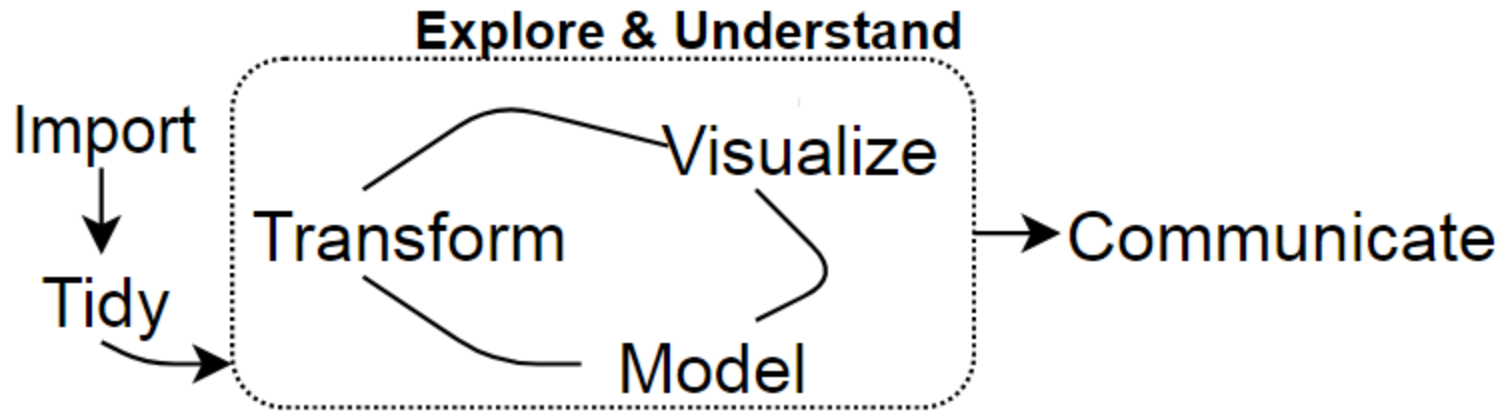


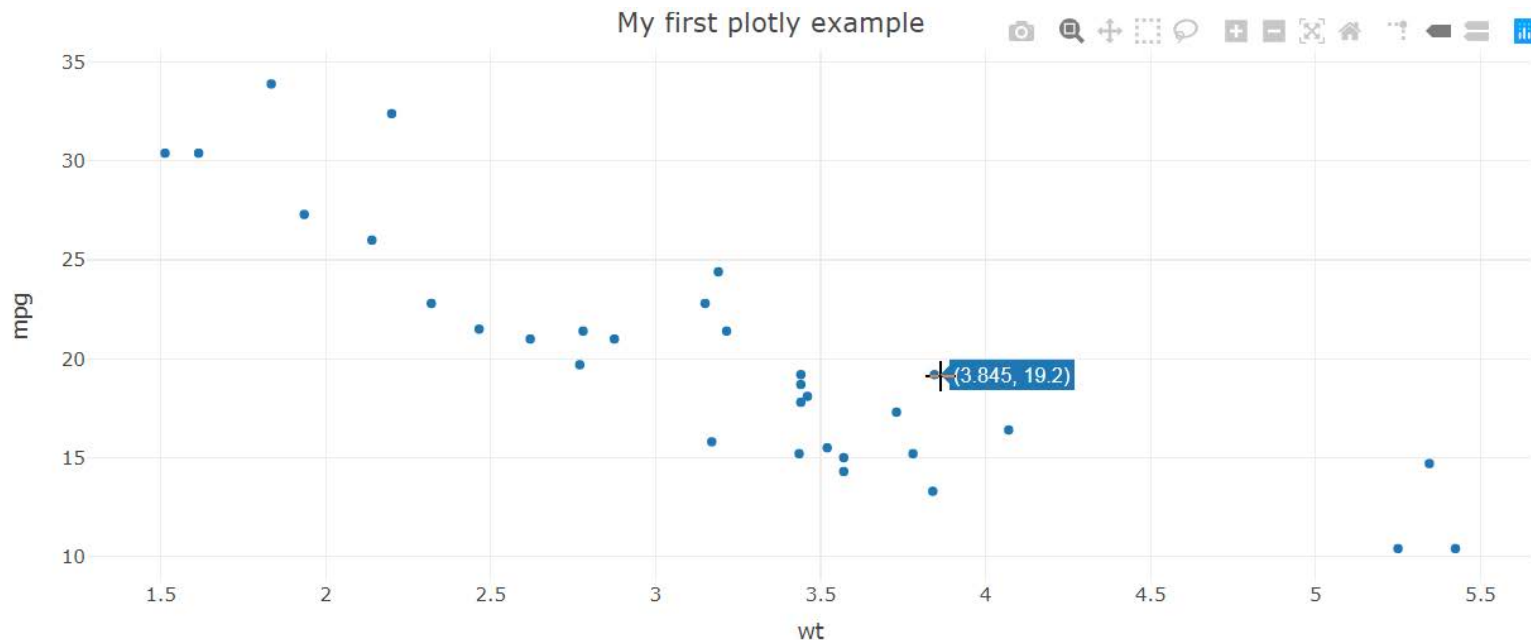
Diagram from: *R for Data Science* (Grolemund & Wickham 2016)



# Install plotly and your first example

The graph is interactive when view on a webpage such as jupyter notebook. Tooltips appear when your mouse hovers over each point.

```
install.packages("plotly")  
library(plotly)  
plot_ly(mtcars, x=~wt, y=~mpg) %>% layout(title = "My first plotly example")
```



# Key components in plotly

- In plotly terminology, a *figure* has two key components
- data (aka, traces)
  - Defines a mapping from data and visuals
  - Every trace has a *type* (e.g., histogram, pie, scatter, etc)
  - The trace type determines what other attributes (i.e., visual and/or interactive properties, like x, hoverinfo, name) are available to control the trace mapping
- layout
  - layout() function anticipates a **plotly** object in it's first argument
  - Other arguments add and/or modify various layout components of that object (e.g., the title)

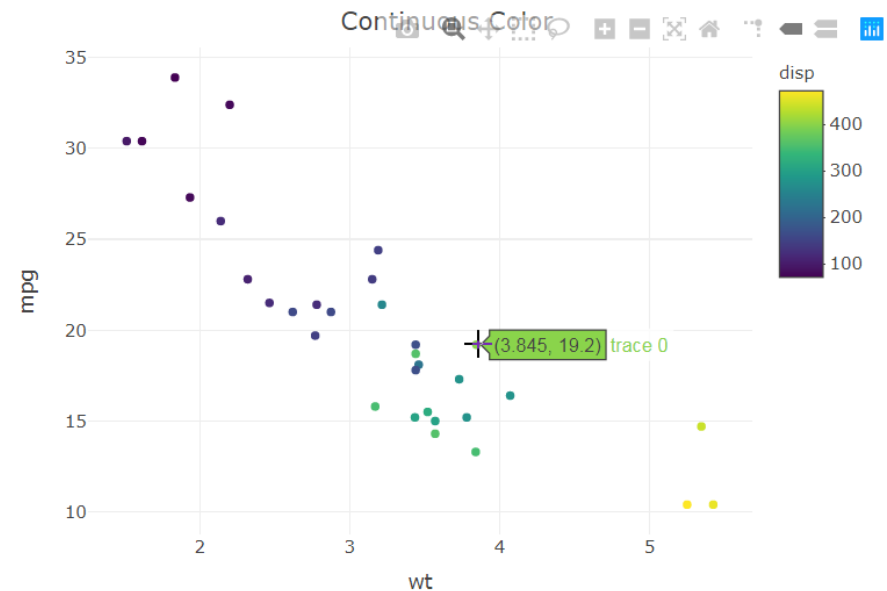
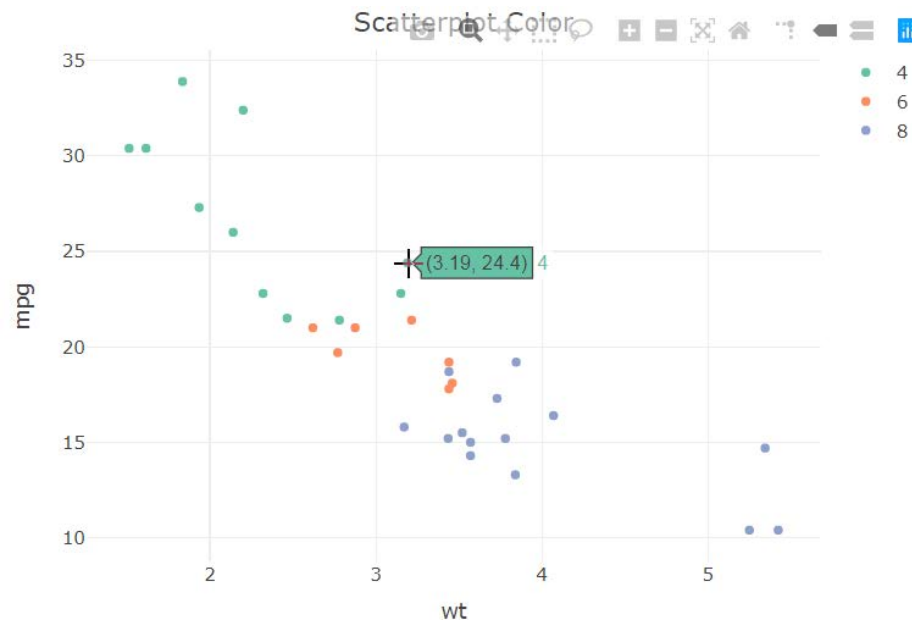
```
plot_ly(mtcars, x=~wt, y=~mpg) %>% layout(title = "My first plotly example")
```

↑  
Trace

↑  
Layout

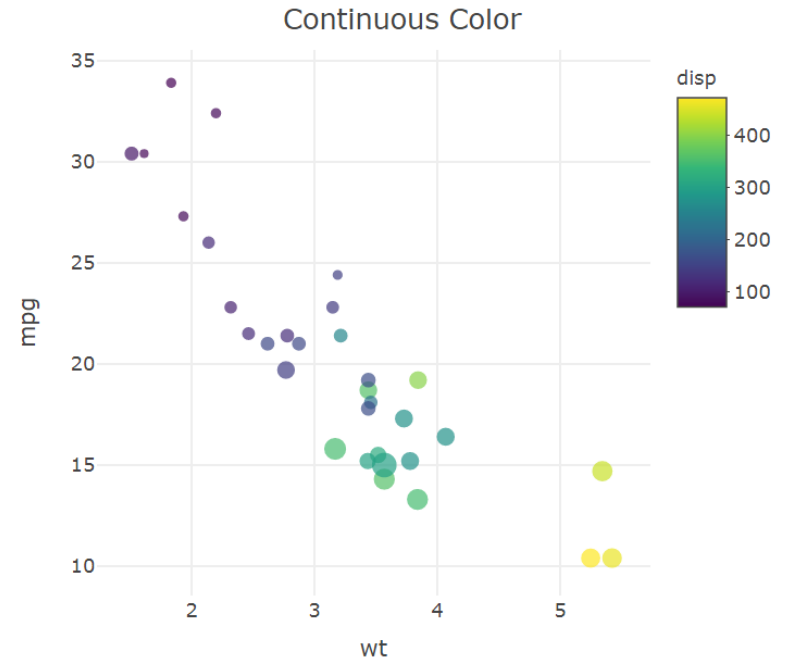
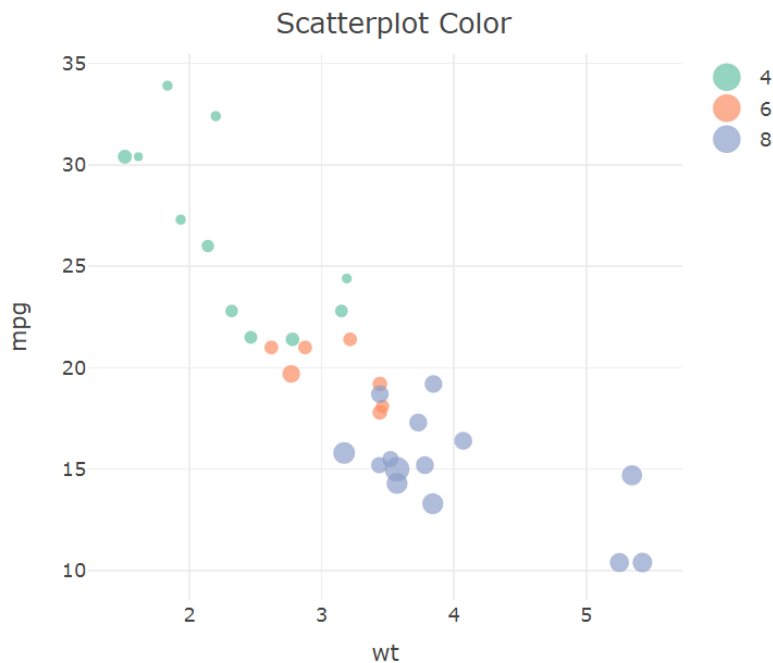
# Add the 3<sup>rd</sup> dimension to scatterplot

```
plot_ly(mtcars, x=~wt, y=~mpg, mode="markers", type="scatter",  
        color=~factor(cyl)) %>% layout(title = "Scatterplot Color")  
plot_ly(mtcars, x=~wt, y=~mpg, mode="markers", type="scatter",  
        color=~disp) %>% layout(title = "Continuous Color")
```



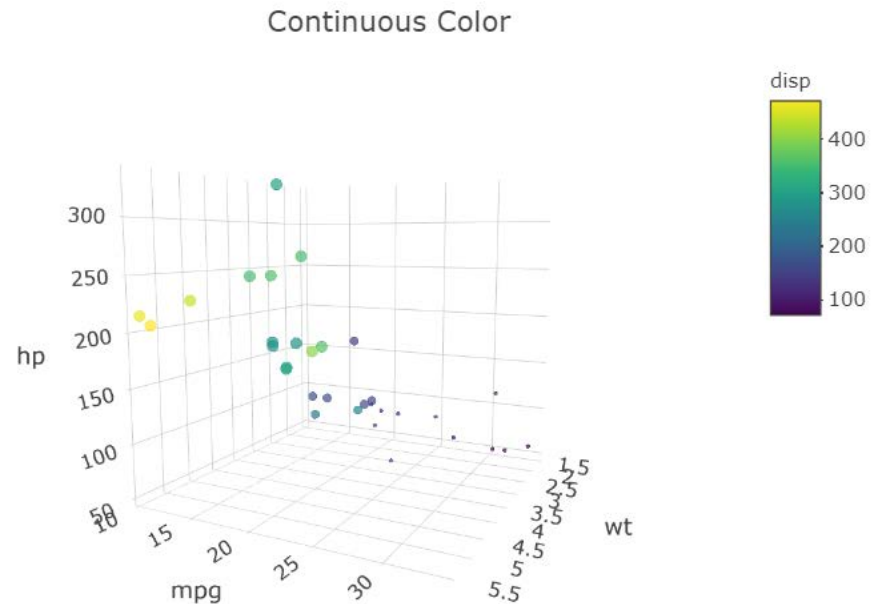
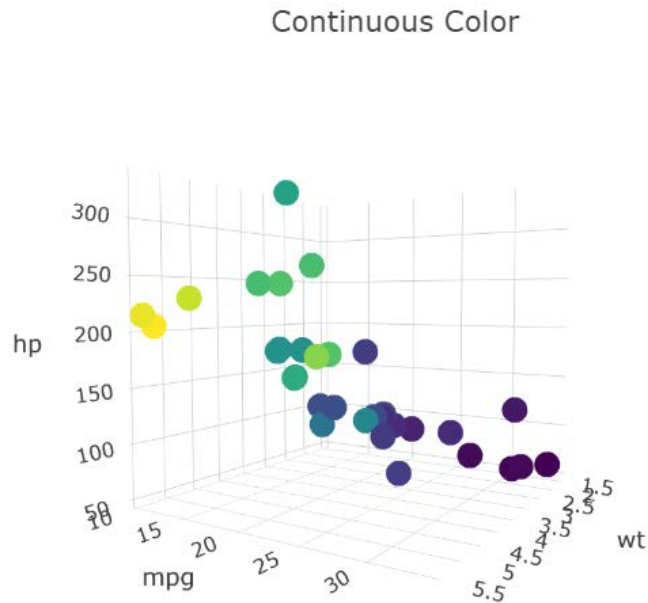
# Add the 4<sup>th</sup> dimension to scatterplot

```
plot_ly(mtcars, x=~wt, y=~mpg, mode="markers", type="scatter",  
        color=~factor(cyl), size=~hp) %>% layout(title = "Scatterplot Color")  
plot_ly(mtcars, x=~wt, y=~mpg, mode="markers", type="scatter",  
        color=~disp, size=~hp) %>% layout(title = "Continuous Color")
```



# Plot 4 and 5 dimensions in 3d scatterplot

```
plot_ly(mtcars, x=~wt, y=~mpg, z=~hp, mode="markers", type="scatter3d",  
        color=~disp) %>% layout(title = "Continuous Color")  
plot_ly(mtcars, x=~wt, y=~mpg, z=~hp, mode="markers", type="scatter3d",  
        color=~disp, size=~cyl) %>% layout(title = "Continuous Color")
```



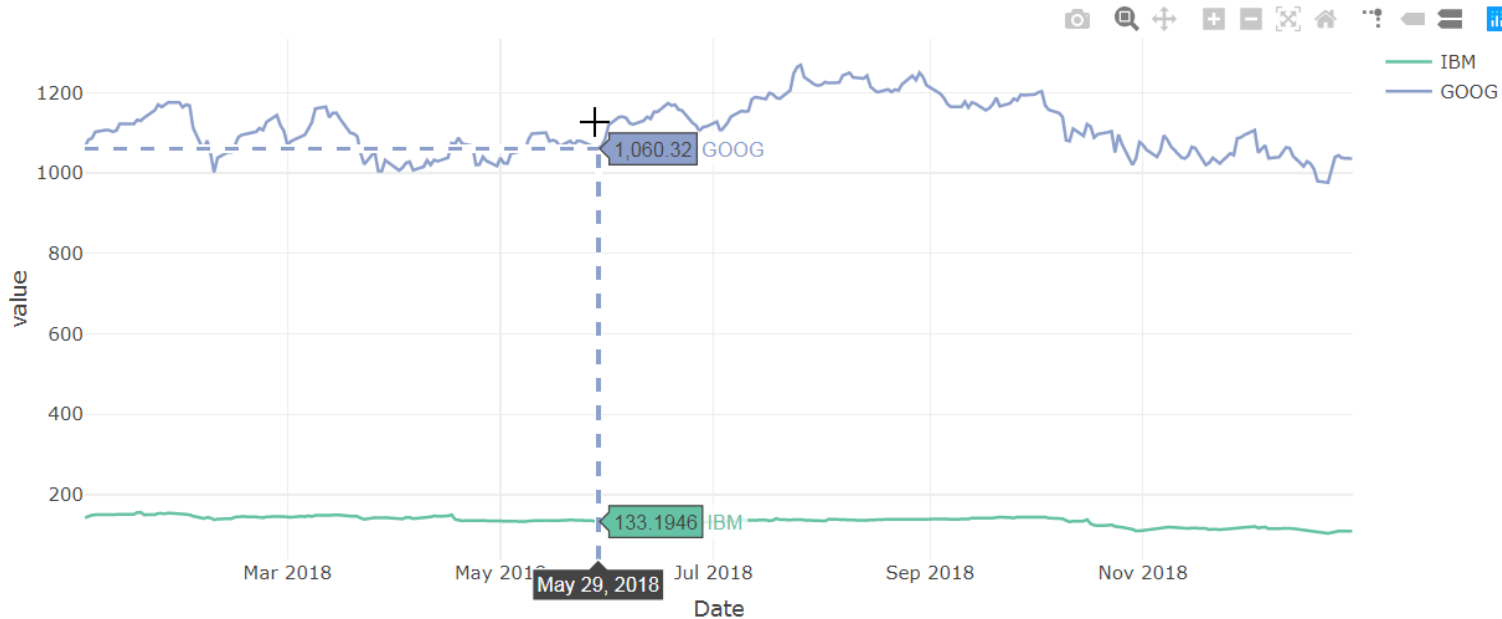
# Lineplot in plotly: time series

```
plot_ly(x=IBM$Date, y=IBM$Adj.Close) %>% add_lines  
or  
IBM %>% plot_ly(x=~Date, y=~Adj.Close) %>% add_lines
```



# Plot multiple time series

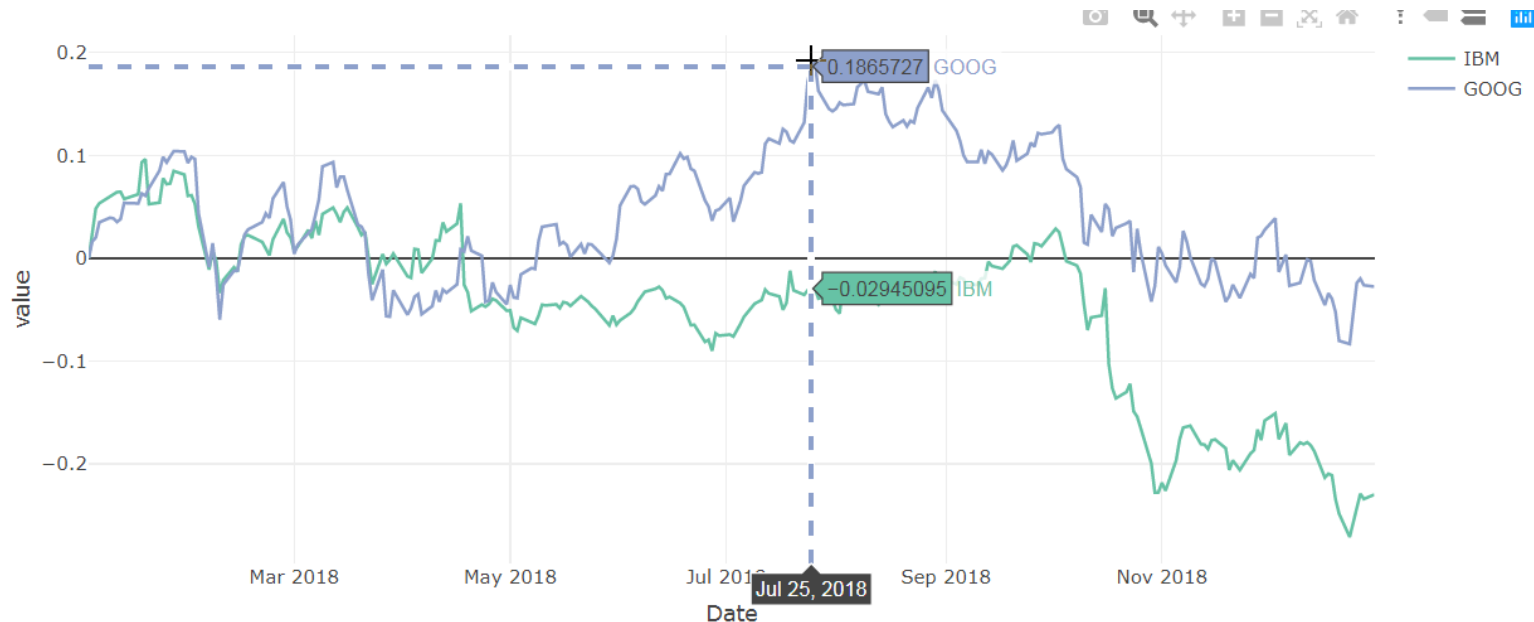
```
data.frame(Date=IBM$Date, IBM=IBM$Adj.Close, GOOG=GOOG$Adj.Close) %>%  
  reshape2::melt(id.vars="Date") %>%  
  plot_ly(x=~Date, y=~value, color=~variable) %>% add_lines
```



Same problem. We need to plot the percent change to compare two stocks

# Plot multiple time series with percent change

```
ibm_startprice=IBM[1, 6]
goog_startprice=GOOG[1, 6]
data.frame(Date=IBM$Date,
            IBM=transmute(IBM, IBM=(Adj.Close-ibm_startprice)/ibm_startprice),
            GOOG=transmute(GOOG, GOOG=(Adj.Close-goog_startprice)/goog_startprice)) %>%
reshape2::melt(id.vars="Date") %>%
plot_ly(x=~Date, y=~value, color=~variable) %>% add_lines
```



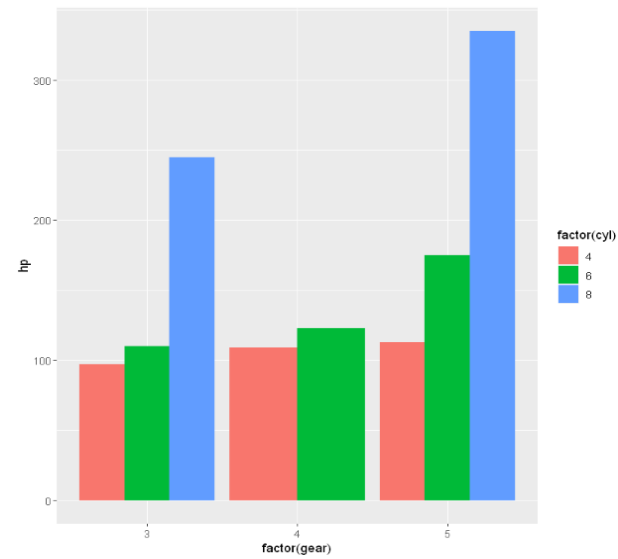
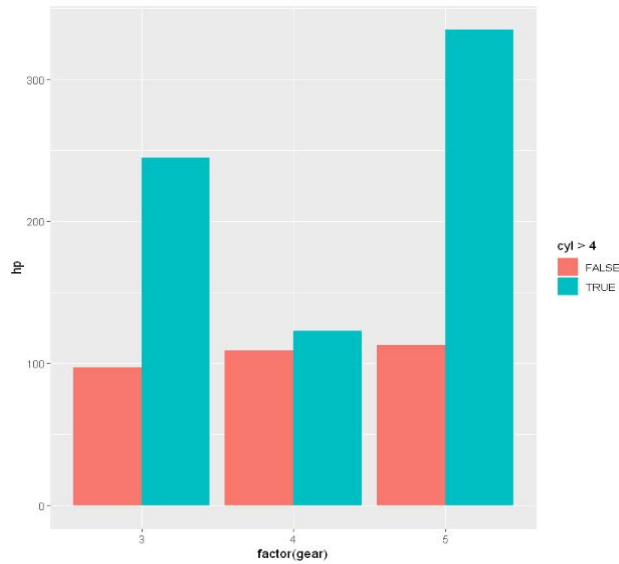


# Lab exercise of today

1. Using plotly, figure out how to plot stock OHLC chart and plot IBM as below



2. Using plotly, write R code to generate grouped barplots similar to ggplot2 as below



# Read more

- <https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html>
  - <http://r-statistics.co/ggplot2-Tutorial-With-R.html>
  - <https://www.r-graph-gallery.com/index.html>
  - <https://plotly-r.com/>
- Issue with ggplot2, geom\_bar, and position= “dodge” : stacked has correct y values, dodged does not
- <https://stackoverflow.com/questions/11604070/issue-with-ggplot2-geom-bar-and-position-dodge-stacked-has-correct-y-values>