Dataframes with Tidyverse

Intro to qCMB

Kristin Fluke, CMB PhD Candidate

Agenda

Monday

- Download the power point presentation and the RMarkdown document, Tidyverse_HW.rmd, and toy datasets. **Put everything in the same folder**.
- Follow along with the live demonstrations

Wednesday

- Finish presentation, complete HW individually or in a group.
- Knit your assignment to PDF or HTML and submit on canvas

Friday

I am available to help! I will hold office hours after CMB seminar

AZ H210, 11am-1pm

Dataframes

- A data structure
- 2-dimensional table of rows and columns (like an excel spreadsheet)
- One of the most common data structures; flexible and intuitive
- Data types:
  - **Character** (chr; “hello world!”)
  - **Factor** (fctr, categorical; up/down, +/-, A/T/U/G/C)
  - **Numeric** (integer/int, double/dbl, floats, longs, etc)
  - **Boolean** (bool, True/False, 0/1, yes/no)
  - **date/time**
  - **null/missing values** (NULL or NA)

Column names (header)

Row index

df [18x4]

	date	instructor	topic	HW Due
1	01/17/22	NA	NA	NA
2	01/19/22	David	Intro/syllabus	NA
3	01/24/22	Erin	R/Rstudio	NA
4	01/26/22	Erin	R/Rstudio	NA
5	01/31/22	Brooke	Rmarkdown	Assignment 1
6	02/02/22	Brooke	Rmarkdown	NA
7	02/07/22	Jessica	Biological data/genome data	NA
8	02/09/22	Jessica	Biological data/genome data	Assignment 2
9	02/14/22	volunteer	Student vignettes	Assignment 3
10	02/16/22	volunteer	Student vignettes	NA
11	02/21/22	Kristin	Tidyverse	Assignment 4
12	02/23/22	Kristin	Tidyverse	NA
13	02/28/22	David	CMD1	Assignment 5
14	03/02/22	David	CMD1	NA
15	03/07/22	David	CMD2	Assignment 6
16	03/09/22	David	CMD2	NA
17	03/14/22	NA	NA	NA
18	03/16/22	NA	NA	NA

What data types is each column?

character (“hello world!”)
factor (ordered catagorical data)
numeric (integer, double)
dates and times
null/missing values (NA)

	chromosome	genomic_coordinate	nucleotide	strand	upstream_sequence	seq_length	sequence_frequnency
1	3	25136245	A	+	AGTCATGCTG	10	0.002
2	4	8692145	T	-	TGCTGGTC	8	0.134
3	2	69314589	G	+	GCTAGTCAAAC	11	0.894
4	x	93148894	C	-	AGCTCGAAA	9	0.000

Wide format

Multiple data points (observations) per row

observations

variables

group	Replicate 1	Replicate 2	Replicate 3
control	12	11	14
treatment	29	21	27

variables

observations

replicate	control	treamtment
Replicate 1	12	29
Replicate 2	11	21
Replicate 3	14	27

"Tidy data"

→ Every column is a variable

→ Every row is an observation

Long format

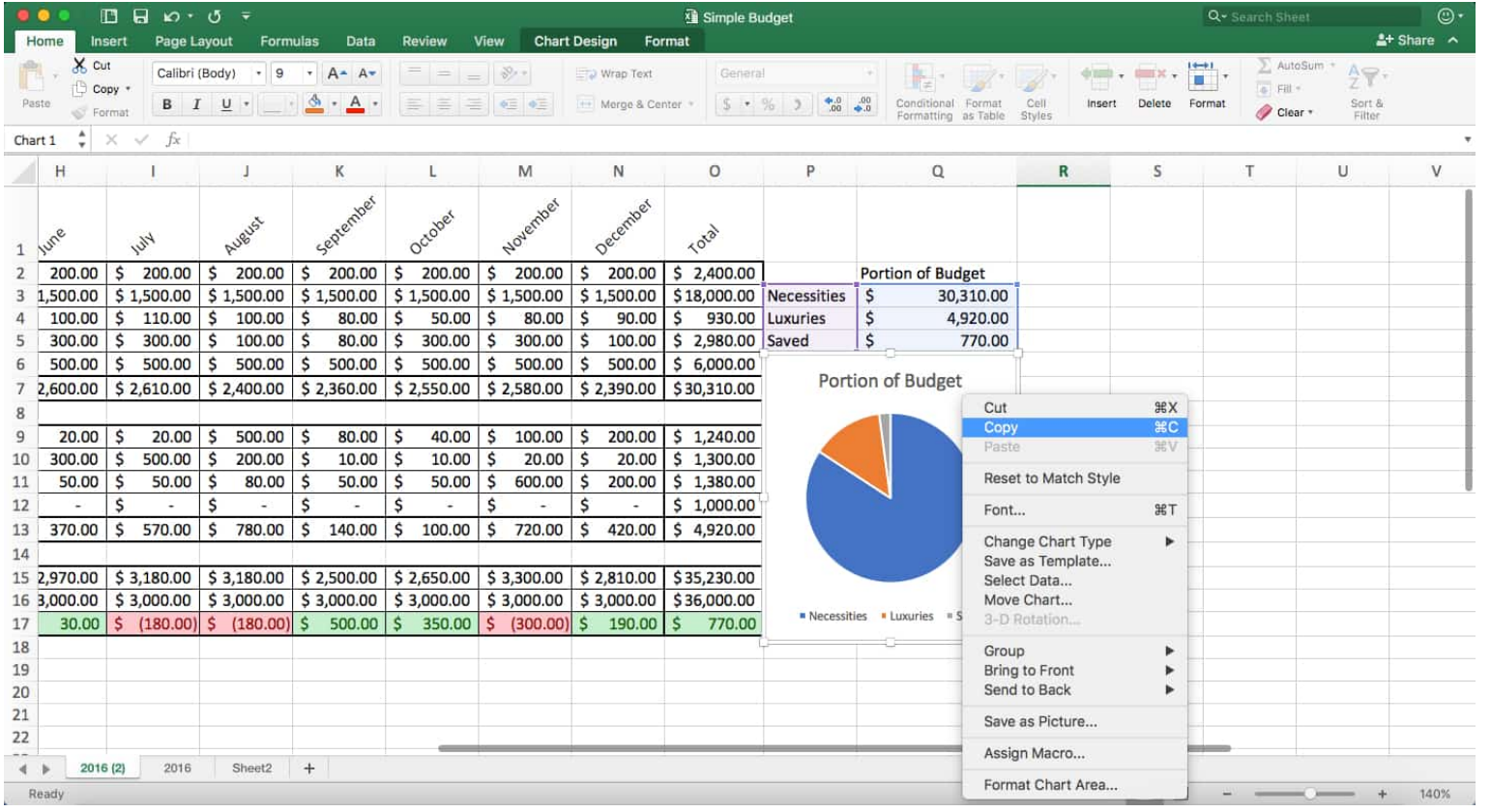
One data point (observation) per row

group	replicate	observation
control	1	12
control	2	11
control	3	14
treatment	1	29
treatment	2	21
treatment	3	27

All information for each observation is encoded by a row values, and only ONE observation is listed per row

Storing 2D Data

Excel spreadsheet



	H	I	J	K	L	M	N	O
1	June	July	August	September	October	November	December	Total
2	200.00	$ 200.00	$ 200.00	$ 200.00	$ 200.00	$ 200.00	$ 200.00	$ 2,400.00
3	1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 18,000.00
4	100.00	$ 110.00	$ 100.00	$ 80.00	$ 50.00	$ 80.00	$ 90.00	$ 930.00
5	300.00	$ 300.00	$ 100.00	$ 80.00	$ 300.00	$ 300.00	$ 100.00	$ 2,980.00
6	500.00	$ 500.00	$ 500.00	$ 500.00	$ 500.00	$ 500.00	$ 500.00	$ 6,000.00
7	2,600.00	$ 2,610.00	$ 2,400.00	$ 2,360.00	$ 2,550.00	$ 2,580.00	$ 2,390.00	$ 30,310.00
8								
9	20.00	$ 20.00	$ 500.00	$ 80.00	$ 40.00	$ 100.00	$ 200.00	$ 1,240.00
10	300.00	$ 500.00	$ 200.00	$ 10.00	$ 10.00	$ 20.00	$ 20.00	$ 1,300.00
11	50.00	$ 50.00	$ 80.00	$ 50.00	$ 50.00	$ 600.00	$ 200.00	$ 1,380.00
12	-	$ -	$ -	$ -	$ -	$ -	$ -	$ 1,000.00
13	370.00	$ 570.00	$ 780.00	$ 140.00	$ 100.00	$ 720.00	$ 420.00	$ 4,920.00
14								
15	2,970.00	$ 3,180.00	$ 3,180.00	$ 2,500.00	$ 2,650.00	$ 3,300.00	$ 2,810.00	$ 35,230.00
16	3,000.00	$ 3,000.00	$ 3,000.00	$ 3,000.00	$ 3,000.00	$ 3,000.00	$ 3,000.00	$ 36,000.00
17	30.00	$ (180.00)	$ (180.00)	$ 500.00	$ 350.00	$ (300.00)	$ 190.00	$ 770.00
18								
19								
20								
21								
22								

	P	Q
		Portion of Budget
Necessities		$ 30,310.00
Luxuries		$ 4,920.00
Saved		$ 770.00

Storing 2D Data

Delimited text file

Tab separated (TSV)

```
date    instructor  topic   HW Due
01/17/22    NA  NA  NA
01/19/22    David   Intro/syllabus  NA
01/24/22    Erin    R/Rstudio   NA
01/26/22    Erin    R/Rstudio   NA
01/31/22    Brooke  Rmarkdown   Assignment 1
02/02/22    Brooke  Rmarkdown   NA
02/07/22    Jessica Biological data/genome data NA
02/09/22    Jessica Biological data/genome data Assignment 2
02/14/22    volunteer   Student vignettes   Assignment 3
02/16/22    volunteer   Student vignettes   NA
02/21/22    Kristin Tidyverse   Assignment 4
02/23/22    Kristin Tidyverse   NA
02/28/22    David   CMD1    Assignment 5
03/02/22    David   CMD1    NA
03/07/22    David   CMD2    Assignment 6
03/09/22    David   CMD2    NA
03/14/22    NA  NA  NA
03/16/22    NA  NA  NA
```

Comma separated (CSV)

```
date,instructor,topic,HW Due
01/17/22,NA,NA,NA
01/19/22,David,Intro/syllabus,NA
01/24/22,Erin,R/Rstudio,NA
01/26/22,Erin,R/Rstudio,NA
01/31/22,Brooke,Rmarkdown,Assignment 1
02/02/22,Brooke,Rmarkdown,NA
02/07/22,Jessica,Biological data/genome data,NA
02/09/22,Jessica,Biological data/genome data,Assignment 2
02/14/22,volunteer,Student vignettes,Assignment 3
02/16/22,volunteer,Student vignettes,NA
02/21/22,Kristin,Tidyverse,Assignment 4
02/23/22,Kristin,Tidyverse,NA
02/28/22,David,CMD1,Assignment 5
03/02/22,David,CMD1,NA
03/07/22,David,CMD2,Assignment 6
03/09/22,David,CMD2,NA
03/14/22,NA,NA,NA
03/16/22,NA,NA,NA
```

* separated

```
date*instructor*topic*HW Due
01/17/22*NA*NA*NA
01/19/22*David*Intro/syllabus*NA
01/24/22*Erin*R/Rstudio*NA
01/26/22*Erin*R/Rstudio*NA
01/31/22*Brooke*Rmarkdown*Assignment 1
02/02/22*Brooke*Rmarkdown*NA
02/07/22*Jessica*Biological data/genome data*NA
02/09/22*Jessica*Biological data/genome data*Assignment 2
02/14/22*volunteer*Student vignettes*Assignment 3
02/16/22*volunteer*Student vignettes*NA
02/21/22*Kristin*Tidyverse*Assignment 4
02/23/22*Kristin*Tidyverse*NA
02/28/22*David*CMD1*Assignment 5
03/02/22*David*CMD1*NA
03/07/22*David*CMD2*Assignment 6
03/09/22*David*CMD2*NA
03/14/22*NA*NA*NA
03/16/22*NA*NA*NA
```

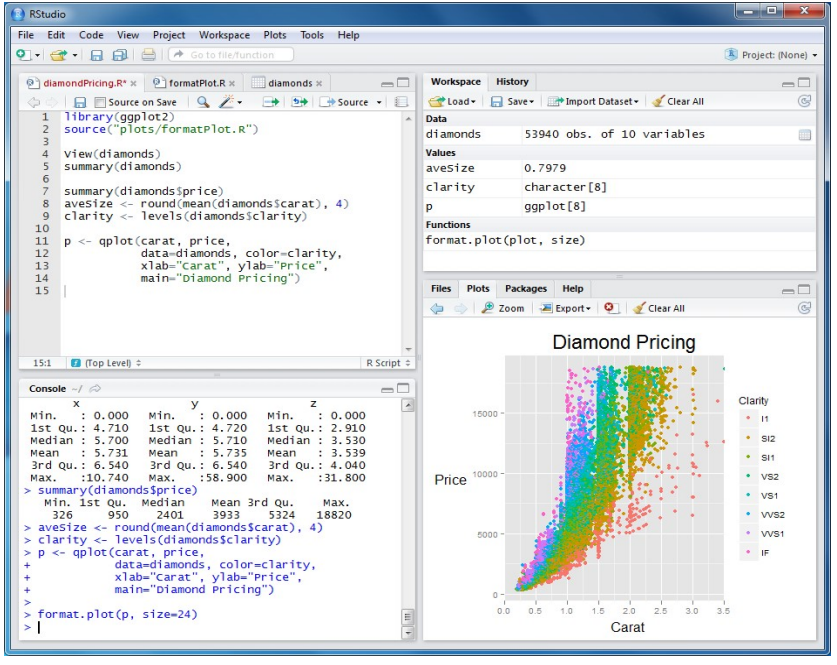
Manipulating Dataframes

- Add/remove/modifying/retrieving data
- Large data frames can be difficult to compute
- Efficiency?
- Accuracy?
- Reproducible?
- Transparent?

R is a programming language for **statistical computing** and **graphics** supported by the R Core Team and the R Foundation for Statistical Computing.

- **Created by statisticians** Ross Ihaka and Robert Gentleman
- R is used among data miners and statisticians **for data analysis** and developing statistical software

RStudio®


RStudio
File Edit Code View Project Workspace Plots Tools Help
diamondPricing.R* formatPlot.R diamonds
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
Workspace History
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Diamond Pricing
Price
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Console
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
> format.plot(p, size=24)


vctrs
%>%
fs
test that
blogdown
sparklyr
withr
parsnip
tidypredict
scales
lobstr
rvest
reprex
tidymodels
dbplot
feather
covr
usethis
googledrive
modeldb
roxygen2
plumber
glue
modelr
haven
rmarkdown
yardstick
rlang
tidyposterior
bookdown
Shiny
tidyr
googlesheets
purrr
hms
tidyverse
xaringan
dplyr
FIZZLE
lubridate
forcats
recipes
rsample
readr
pkg down
R2D3
ggplot2
broom
knitr
readxl
devtools
RStudio
stringr

RMarkdown Quick Guide

```
```{r Toothgrowth dataset part 1}

data("ToothGrowth")

ToothGrowth_df <- ToothGrowth %>%
  group_by(supp, dose) %>%
  summarize(n=n(), mean = mean(len), sd = sd(len))

head(ToothGrowth_df) #prints the first 6 rows of the dataframe
#tail(ToothGrowth_df) #prints the last 6 rows of the dataframe

```
```

- A cell (or coding chunk) is denoted with three tick marks that open and close the cell (```)
- At the top of the cell in curly brackets provides options to manipulate the output of the cell. The first argument is the name of the cell. You may learn about additional arguments later on in this course.
- The green play button allows you run the code in the cell

RMarkdown Quick Guide

```
**Change the author on line 3 to your name**

## ToothGrowth part 1
Take a look at the pre-built R dataset "ToothGrowth". Go to this link to learn about the data:

*https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/ToothGrowth*

Also, take a look at the code chunk already written for you. The head() function are written for you, and will result in printing the first 6 rows of the ToothGrowth
dataframe. We are going to build up this dataframe using the functions available in tidyverse. Run the code chunck bellow, then answer the in-line questions.

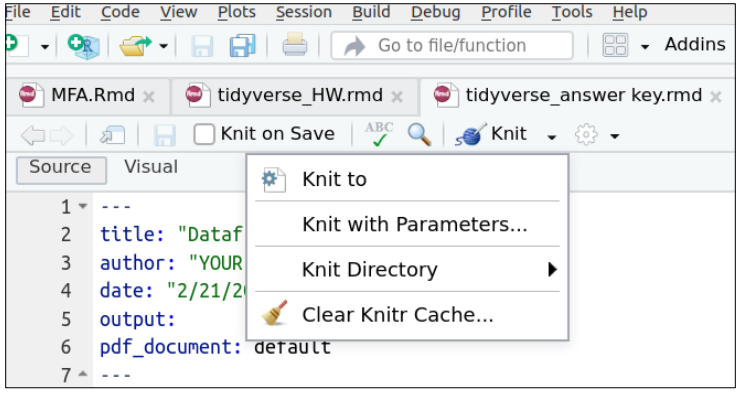
**Is the data in long format or wide format?:**

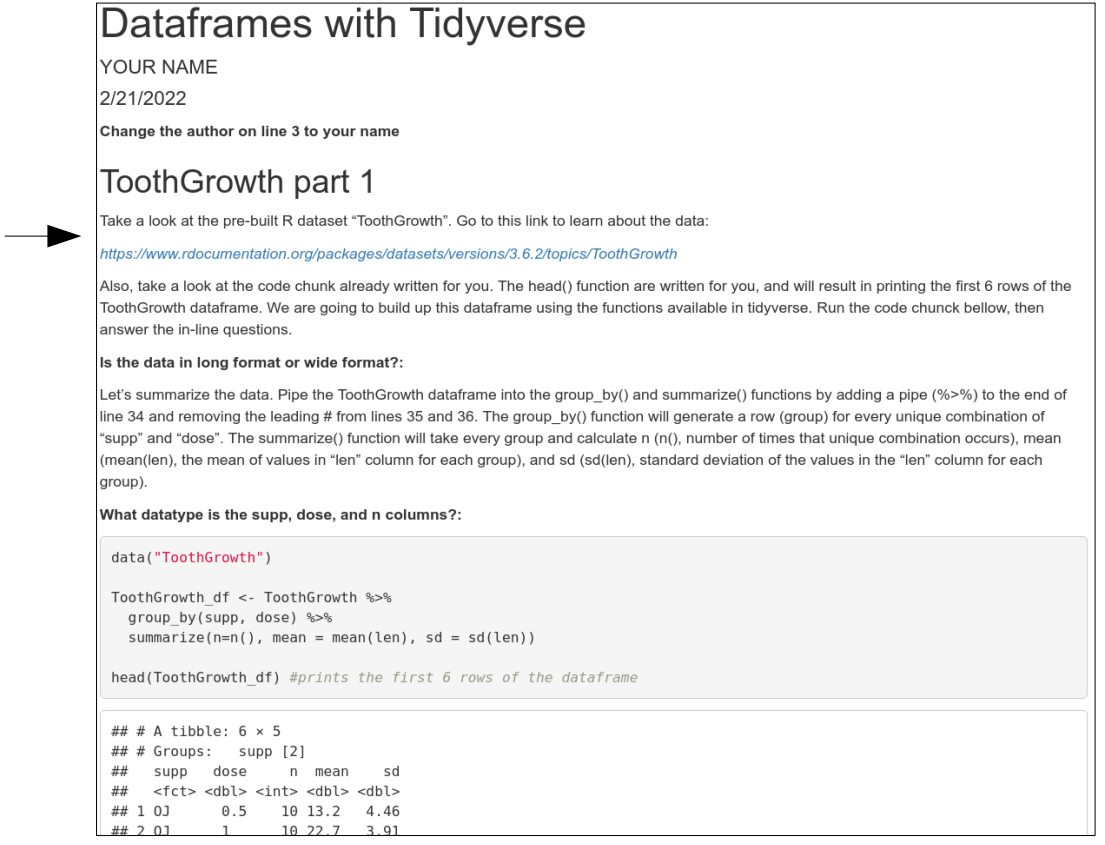
Let's summarize the data. Pipe the ToothGrowth dataframe into the group_by() and summarize() functions by adding a pipe (%>%) to the end of line 34 and removing the
leading # from lines 35 and 36. The group_by() function will generate a row (group) for every unique combination of "supp" and "dose". The summarize() function will take
every group and calculate n (n(), number of times that unique combination occurs), mean (mean(len), the mean of values in "len" column for each group), and sd (sd(len),
standard deviation of the values in the "len" column for each group).

**What datatype is the supp, dose, and n columns?:**
```

- There are many ways to make your RMarkdown document look pretty.
- The hashes (#) and stars (*) are important. Please don’t remove them.

RMarkdown Quick Guide





Dataframes with Tidyverse

YOUR NAME

2/21/2022

Change the author on line 3 to your name

ToothGrowth part 1

Take a look at the pre-built R dataset "ToothGrowth". Go to this link to learn about the data:

https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/ToothGrowth

Also, take a look at the code chunk already written for you. The head() function are written for you, and will result in printing the first 6 rows of the ToothGrowth dataframe. We are going to build up this dataframe using the functions available in tidyverse. Run the code chunck bellow, then answer the in-line questions.

Is the data in long format or wide format?:

Let's summarize the data. Pipe the ToothGrowth dataframe into the group_by() and summarize() functions by adding a pipe (%>%) to the end of line 34 and removing the leading # from lines 35 and 36. The group_by() function will generate a row (group) for every unique combination of "supp" and "dose". The summarize() function will take every group and calculate n (n(), number of times that unique combination occurs), mean (mean(len), the mean of values in "len" column for each group), and sd (sd(len), standard deviation of the values in the "len" column for each group).

What datatype is the supp, dose, and n columns?:

```
data("ToothGrowth")

ToothGrowth_df <- ToothGrowth %>%
  group_by(supp, dose) %>%
  summarize(n=n(), mean = mean(len), sd = sd(len))

head(ToothGrowth_df) #prints the first 6 rows of the dataframe
```

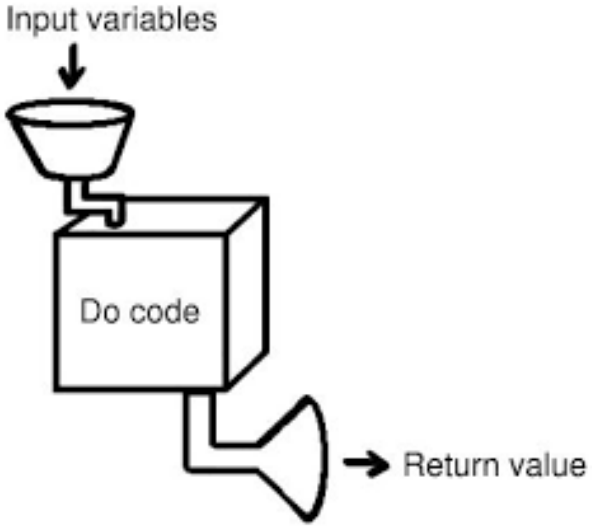
```
## # A tibble: 6 × 5
## # Groups:   supp [2]
##   supp   dose     n  mean    sd
##   <fct> <dbl> <int> <dbl> <dbl>
## 1 OJ      0.5    10 13.2   4.46
## 2 OJ      1      10 22.7   3.91
```

- When your assignment is complete, you will kit your document to a PDF.
- Bugs in your code will prevent the document from knitting.

Data Frames

with

The tidyverse is a **collection of R packages** introduced by Hadley Wickham and his team that share an underlying design philosophy, grammar, and data structures of tidy data.



Functions

```
>   function(argument1, argument2, ….)
 [1] output

>   my_list <- c(3,4,5,6)
>   mean(my_list)
 [1] 4.5
```

df <- read.delim("path/to/df", sep = "\t")

> function(**arg1 = df**, arg2, arg3, ….)
[1] df # modified df with function applied

new_df <- df

df <- read.delim("path/to/df", sep = "\t")

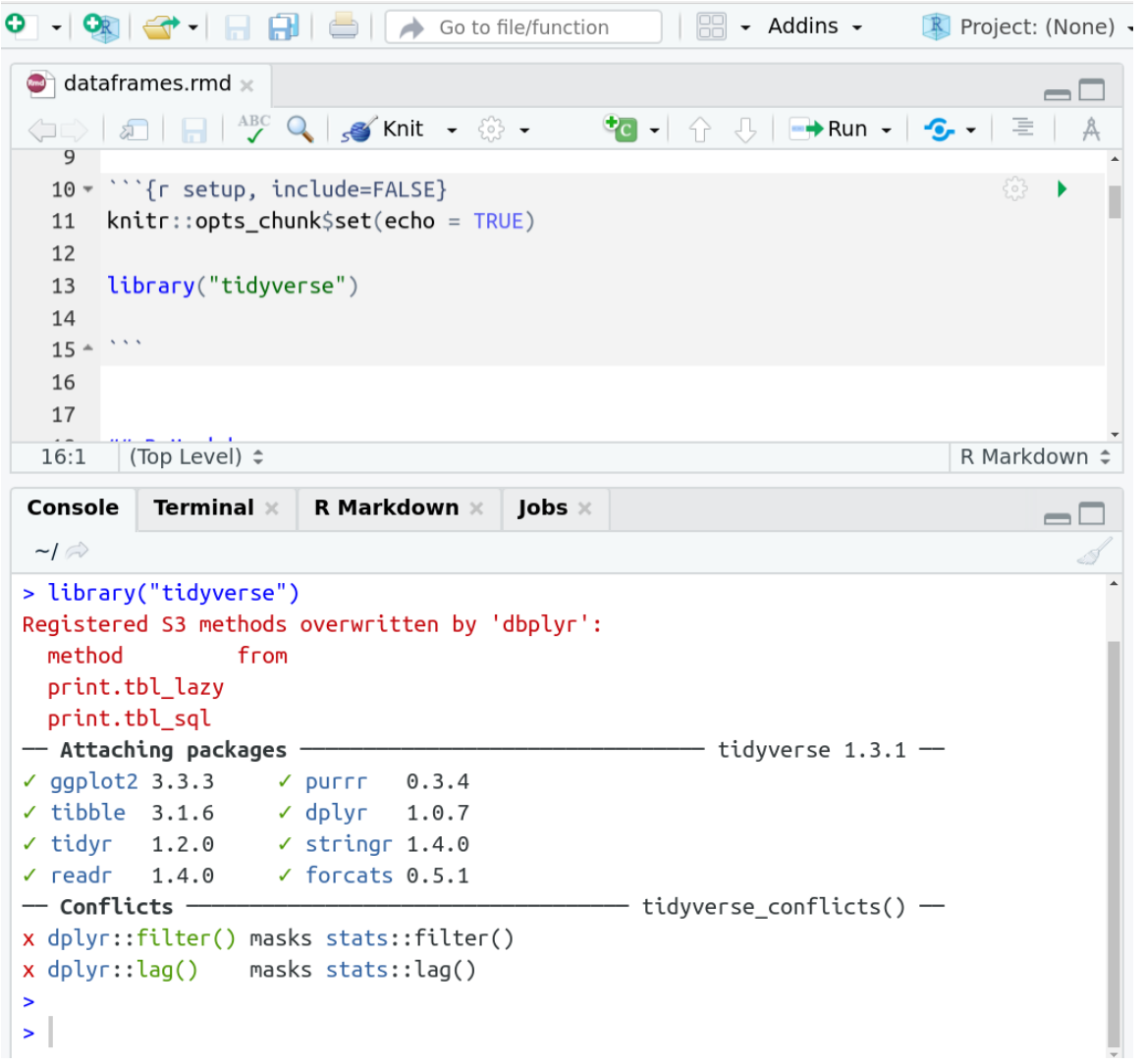
> function(**arg1 = df**, arg2, arg3, ….)
[1] df # modified df with function applied

```
new_df <- df %>%
      function(arg2, arg3, ...) %>%
      function(arg2, arg3, …) %>%
      function(arg2, arg3, …)
```

Install tidyverse:

```
# In your R console, run:
install.packages("tidyverse")
```

Import tidyverse into your session



Tidyverse and associated pacakges are imported

Manipulations we will work on:

- Mutate, select, rename, filter, merge, arrange, gather, spread
- Other functions: grouping & summarizing, reading/writing data
- Datatype conversions

At your discretion:

- Data visualization with ggplot2

Tidyverse grammar

- All functions take a dataframe as **input**
- All functions produce a dataframe as **output**
- It is therefore assumed that the first argument of a tidyverse function is always a dataframe

```
function(data = df, arg = <value> )
```

```
new_df <- df %>%
      function(arg = <value> )
```

group_by / summarize

```
new_df <- df %>%
      group_by(column1, column2, …) %>%
      summarize(<formulas>)
```

> df_long

replicate	group	observation
1	control	12
2	control	11
3	control	14
1	treamtment	29
2	treamtment	21
3	treamtment	27

```
new_df <- df_long %>%
      group_by(group) %>%
      summarize(n=n(),
                mean = mean(observation),
                sd = sd(observation)  )
```

> new_df

group	n	mean	sd
control	3	12.33333	1.527525
treamtment	3	25.66667	4.163332

<demo>

mutate (adding columns)

```
mutate( df, new_column = <formula> )
```

```
new_df <- df %>%
      mutate( new_column = <formula> )
```

```
new_df <- df %>%
      mutate( fold_change = treatment / control )
```

> df

replicate	control	treamtment
1	12	29
2	11	21
3	14	27

> new_df

replicate	control	treamtment	fold_change
1	12	29	2.4
2	11	21	1.91
3	14	27	1.92

mutate (adding columns)

```
mutate( df, new_column = <formula> )
```

```
new_df <- df %>%
      mutate( new_column = <formula> )
```

```
new_df <- df %>%
      mutate( fold_change = treatment / control )
```

```
new_df <- df %>%
      mutate( fold_change = treatment / control ) %>%
      mutate( log2FC = log2(fold_change)  )
```

> df

replicate	control	treamtment
1	12	29
2	11	21
3	14	27

> new_df

replicate	control	treamtment	fold_change	log2FC
1	12	29	2.4	1.2
2	11	21	1.91	0.93
3	14	27	1.92	0.94

select (selecting/removing columns)

```
select(df, column_name, -column_name)
```

```
new_df <- df %>%
      select(column_name, -column_name)
```

```
new_df <- df %>%
      select(replicate, control, treatment,
        fold_change)
```

OR...

```
new_df <- df %>%
      select(-log2FC)
```

select() can also be used to re-order columns

> df

replicate	control	treamtment	fold_change	log2FC
1	12	29	2.4	1.2
2	11	21	1.91	0.93
3	14	27	1.92	0.94

> new_df

replicate	control	treamtment	fold_change
1	12	29	2.4
2	11	21	1.91
3	14	27	1.92

rename (renaming columns)

```
dplyr::rename(df, new_column_name=old_column_name)
```

```
new_df <- df %>%
      dplyr::rename(new_name = old_name)
```

```
new_df <- df %>%
      dplyr::rename(FC = fold_change)
```

> df

replicate	control	treamtment	fold_change
1	12	29	2.4
2	11	21	1.91
3	14	27	1.92

> new_df

replicate	control	treamtment	FC
1	12	29	2.4
2	11	21	1.91
3	14	27	1.92

arrange

```
arrange(df, column_name)
```

```
arrange(df, desc(column_name) )
```

```
arrange(df, -column_name)
```

```
new_df <- df %>%
      arrange(column_name )
```

```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1) %>%
      filter(pvalue < 0.05) %>%
      merge(y=df2, by= "gene", all.x=TRUE)
```

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue	symbol	description
gene 5	32	10	0.31	-1.68	0.02	GAGE1 2F	G antigen 12F
gene 7	12	1	0.08	-3.58	0.001	NAALA D2	N-acetylated acidic dipeptidase 2
gene 10	19	70	3.68	1.88	0.011	ZBTB3 3	zinc finger and BTB domain containing 33

arrange

```
arrange(df, column_name)
```

```
arrange(df, desc(column_name) )
```

```
arrange(df, -column_name)
```

```
new_df <- df %>%
      arrange(column_name )
```

```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1) %>%
      filter(pvalue < 0.05) %>%
      merge(y=df2, by= "gene", all.x=TRUE) %>%
      arrange(pvalue)
```

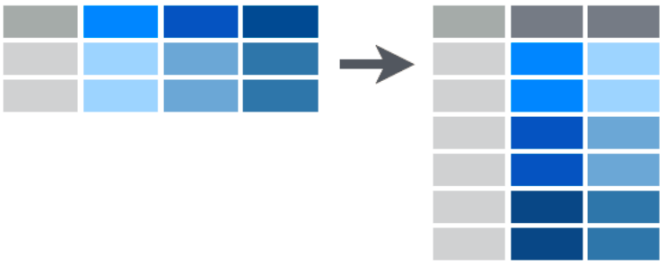
> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue	symbol	description
gene 7	12	1	0.08	-3.58	0.001	NAALAD2	N-acetylated acidic dipeptidase 2
gene 10	19	70	3.68	1.88	0.011	ZBTB33	zinc finger and BTB domain containing 33
gene 5	32	10	0.31	-1.68	0.02	GAGE12F	G antigen 12F

<demo>

gather (melt)

Wide form→ long form



```
new_df <- df %>%
    gather(key = <column_name>,
           value = <column_name>,
           <column_range>)
```

> df_wide

replicate	control	treamtment
1	12	29
2	11	21
3	14	27

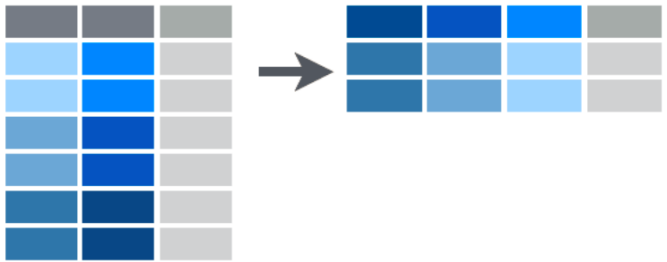
```
df_long <- df_wide %>%
      gather(key= “group”,
             value = “observation”, 2:3)
```

> df_long

replicate	group	observation
1	control	12
2	control	11
3	control	14
1	treamtment	29
2	treamtment	21
3	treamtment	27

spread (pivot)

Long form→ wide form



```
df_wide <- df_long %>%
    spread(key = <column_name>,
           value = <column_name>)
```

> df_long

replicate	group	observation
1	control	12
2	control	11
3	control	14
1	treamtment	29
2	treamtment	21
3	treamtment	27

```
df_wide <- df_long %>%
    spread(key = "group",
           value = "observation")
```

> df_wide

replicate	control	treamtment
1	12	29
2	11	21
3	14	27

<demo>

filter (select rows based on value)

```
filter(df, column_name <operation> <formula>)
```

```
new_df <- df %>%
      filter( column_name <operation> <formula> )
```

```
Operations:
==, !=, >, <
&, |
```

```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1)
```

> df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene1	15	14	0.93	-0.10	0.95
gene2	2	5	2.50	1.32	0.68
gene3	4	7	1.75	0.81	0.45
gene4	6	10	1.67	0.74	0.79
gene5	32	10	0.31	-1.68	0.02
gene6	45	56	1.24	0.32	0.55
gene7	12	1	0.08	-3.58	0.001
gene8	65	57	0.88	-0.19	0.37
gene9	12	12	1.00	0.00	1
gene10	19	70	3.68	1.88	0.011

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene2	2	5	2.50	1.32	0.68
gene5	32	10	0.31	-1.68	0.02
gene7	12	1	0.08	-3.58	0.001
gene10	19	70	3.68	1.88	0.011

```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1) %>%
      filter(pvalue < 0.05)
```

OR...

```
new_df <- df %>%
    filter( (log2FC > 1 | log2FC < -1) & pvalue < 0.05)
```

> df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene1	15	14	0.93	-0.10	0.95
gene2	2	5	2.50	1.32	0.68
gene3	4	7	1.75	0.81	0.45
gene4	6	10	1.67	0.74	0.79
gene5	32	10	0.31	-1.68	0.02
gene6	45	56	1.24	0.32	0.55
gene7	12	1	0.08	-3.58	0.001
gene8	65	57	0.88	-0.19	0.37
gene9	12	12	1.00	0.00	1
gene10	19	70	3.68	1.88	0.011

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene5	32	10	0.31	-1.68	0.02
gene7	12	1	0.08	-3.58	0.001
gene10	19	70	3.68	1.88	0.011

```
new_df <- df %>%
      filter( gene == "gene1" )
```

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene1	15	14	0.93	-0.10	0.95

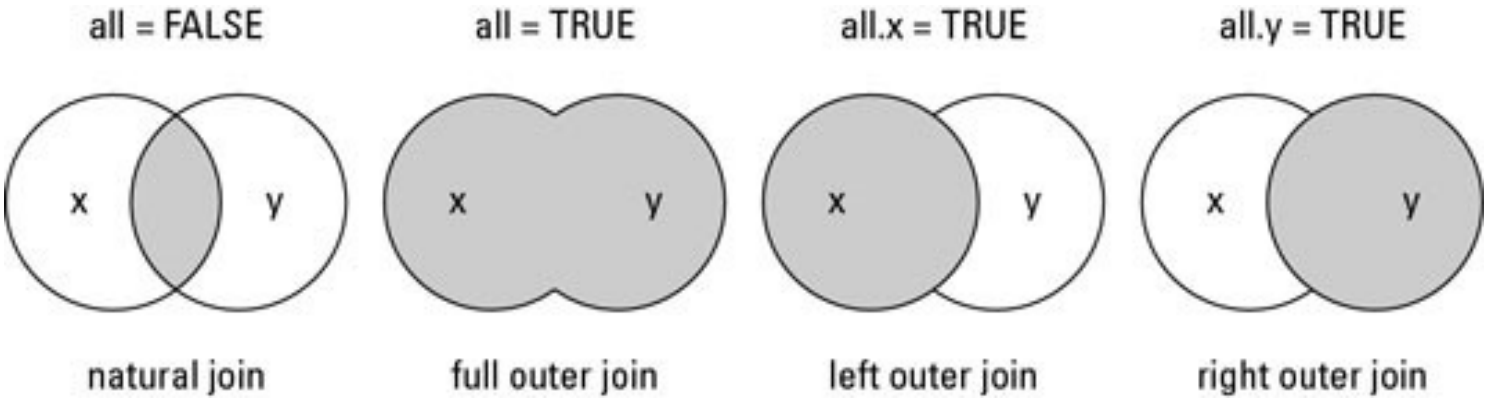
```
new_df <- df %>%
      filter( gene != "gene1" )
```

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene2	2	5	2.50	1.32	0.68
gene3	4	7	1.75	0.81	0.45
gene4	6	10	1.67	0.74	0.79
gene5	32	10	0.31	-1.68	0.02
gene6	45	56	1.24	0.32	0.55
gene7	12	1	0.08	-3.58	0.001
gene8	65	57	0.88	-0.19	0.37
gene9	12	12	1.00	0.00	1
gene10	19	70	3.68	1.88	0.011

merge dataframes

```
merge(x = df.x, y = df.y, by = c("column_name"), all.x = TRUE)
                                                 all.y = TRUE
                                                 all = TRUE
                                                 all = FALSE
```



```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1) %>%
      filter(pvalue < 0.05)
```

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue
gene5	32	10	0.31	-1.68	0.02
gene7	12	1	0.08	-3.58	0.001
gene10	19	70	3.68	1.88	0.011

> df2

gene	symbol	description
gene1	A1BG	alpha-1-B glycoprotein
gene2	NAT2	N-acetyltransferase 2
gene3	ADA	adenosine deaminase
gene4	CDH2	cadherin 2
gene5	GAGE12F	G antigen 12F
gene6	MED6	mediator complex subunit 6
gene7	NAALAD2	N-acetylated acidic dipeptidase 2
gene8	DUXB	double homeobox B
gene9	DDTL	D-dopachrome tautomerase like
gene10	ZBTB33	zinc finger and BTB domain containing 33

```
new_df <- df %>%
      filter(log2FC > 1 | log2FC < -1) %>%
      filter(pvalue < 0.05) %>%
      merge(y=df2, by= "gene", all.x=TRUE)
```

> new_df

gene	RPKM control	RPKM treatment	fold change	log2FC	pvalue	symbol	description
gene 5	32	10	0.31	-1.68	0.02	GAGE1 2F	G antigen 12F
gene 7	12	1	0.08	-3.58	0.001	NAALA D2	N-acetylated acidic dipeptidase 2
gene 10	19	70	3.68	1.88	0.011	ZBTB3 3	zinc finger and BTB domain containing 33

Data type conversions

replicate	group	observation
1	control	12
2	control	11
3	control	14
1	treamtment	29
2	treamtment	21
3	treamtment	27

=

	replicate <int>	group <chr>	observation <int>
1	1	control	12
2	2	control	11
3	3	control	14
4	1	treamtment	29
5	2	treamtment	21
6	3	treamtment	27

```
# to convert data types:
df$column <- as.<type>(df$column)
```

```
#options:
as.character(df$column)
as.factor(df$column);
      factor(df$column, levels=c(...) )
as.integer(df$column)
as.double(df$column)
```

```
# Examples
df$group <- as.factor(df$group)
```

	replicate <int>	group <fctr>	observation <int>
1	1	control	12
2	2	control	11
3	3	control	14
4	1	treamtment	29
5	2	treamtment	21
6	3	treamtment	27

Importing/Exporting Data

```
# Importing dataframe from text file
df <- read.delim( <path_to_data>, sep = <delimiter> )
```

```
# Exporting dataframe to text file
write.table(df, <path_to_destination>, sep = <delimiter>, row.name=FALSE)
```

```
# Example
df <- read.delim("./project/DGE.tsv", sep = "\t" )

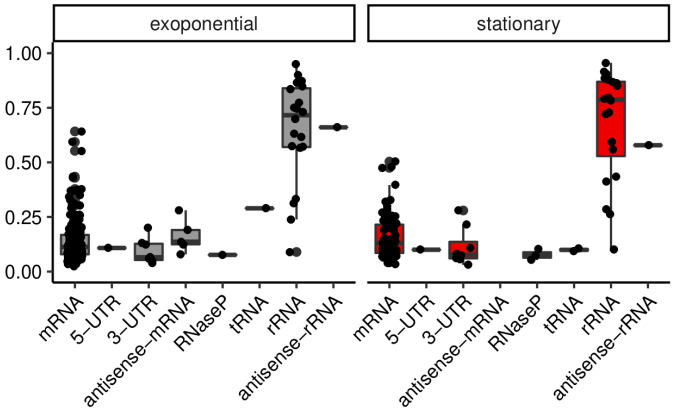
new_df <- df %>% filter( (log2FC > 1 | log2FC < -1) & pvalue < 0.05)

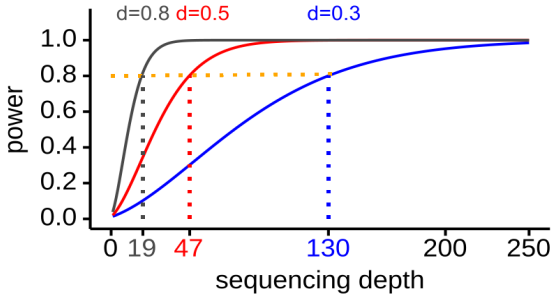
write.table(new_df, "./DGE_analysis.csv", sep = "," , row.name = FALSE)
```

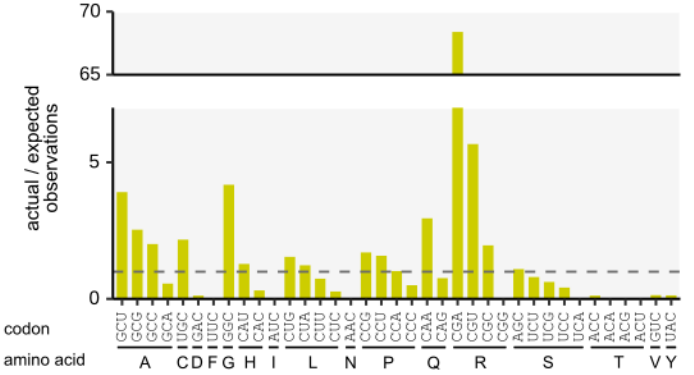
Data Visualization
with

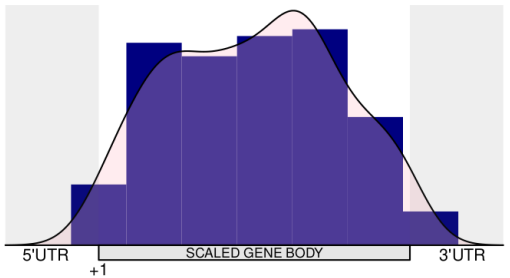
*This section is optional and self-taught

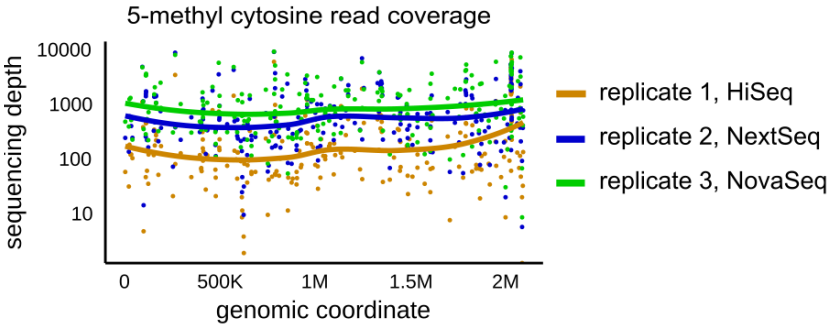
Some of my own ggplots

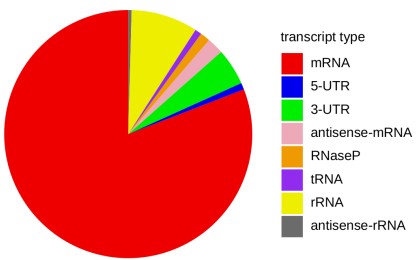


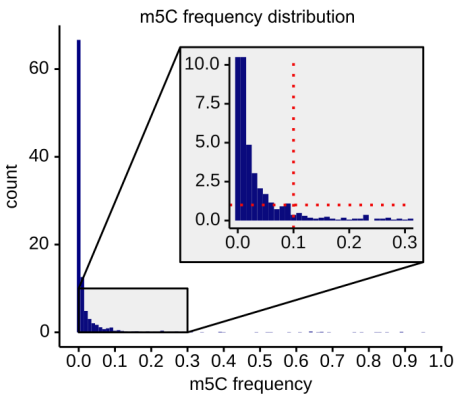


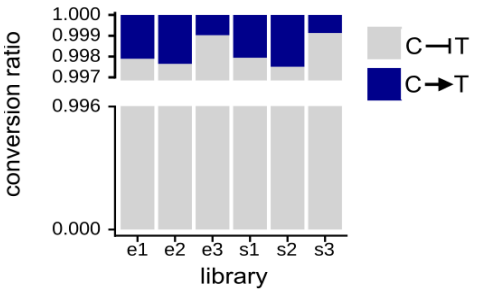








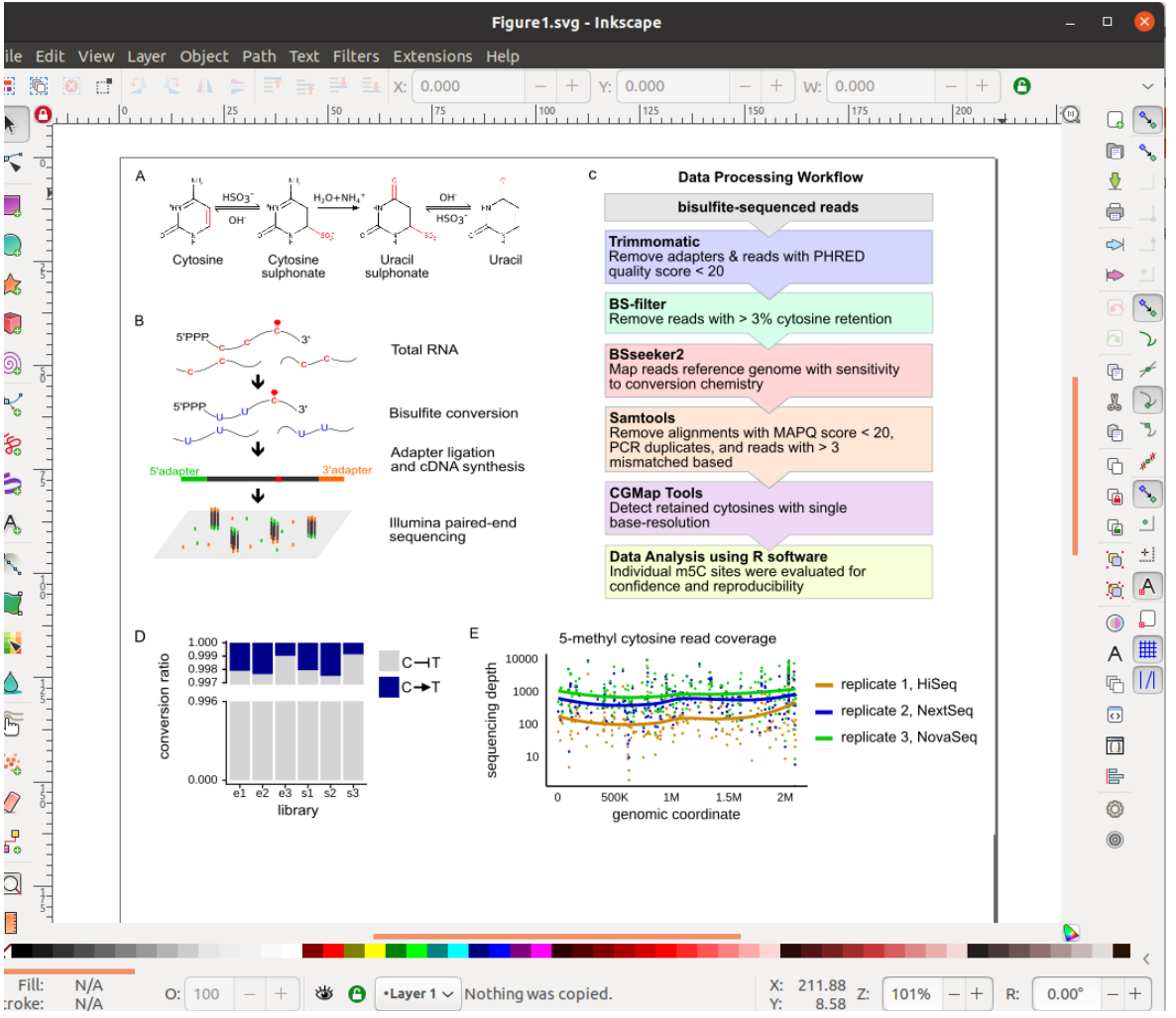




INKSCAPE

Draw Freely.

- Free, open source graphics design platform
- Similar to photoshop and adobe illustrator
- Allows you to arrange and edit figures
- Saved ggplot images can be imported into inkscape and easily edited



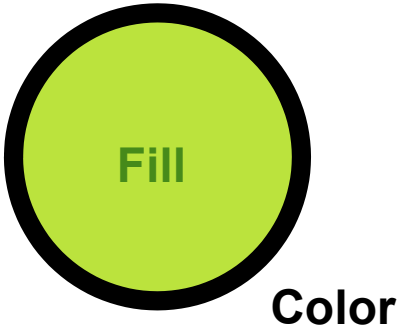
ggplot2

- Building graphs layer-by-layer
- piping operator: +
- **Input**: dataframe (tidy or long format)
- **Output**: ggplot

```
ggplot(data, aes(x=…, y=…, color = …, fill = …)) +
    geom_point() +
    theme_classic() +
    labs(x = "…", y = "…" , title = "…") +
    theme(<args>)
```

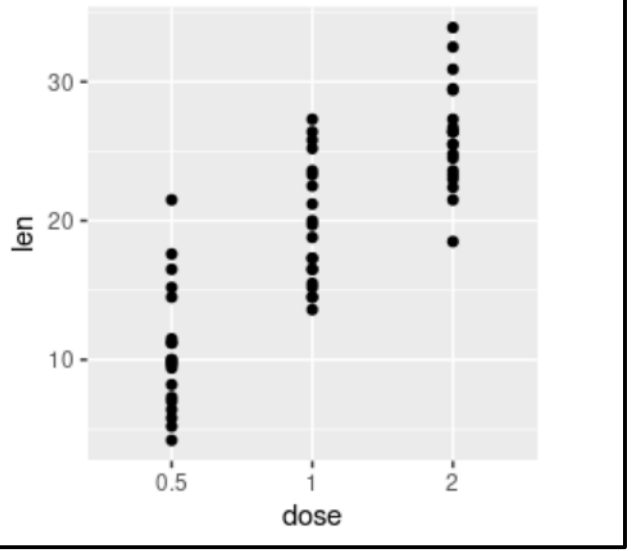
```
ggplot(df, aes(x=…, y=…, color = …, fill = …))
```

```
ggplot(data = df,
        mapping = aes(x = col_name,
                      y = col_name,
                      color = col_name,
                      fill = col_name)
       )
```



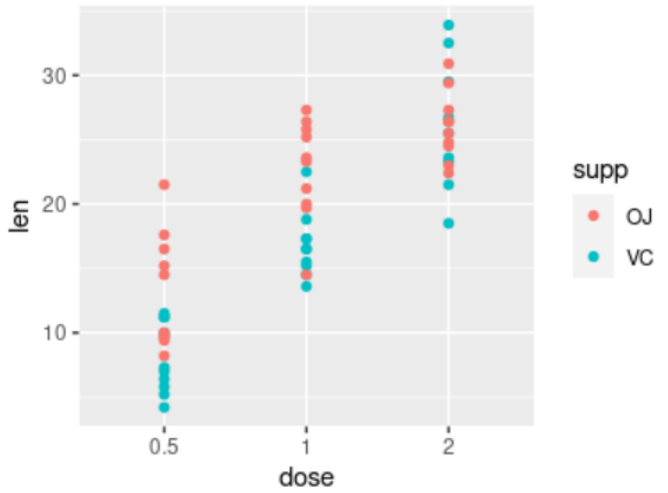
No color assignment

```
ggplot(ToothGrowth_df, aes( x = dose, y = len)) +
    geom_point()
```



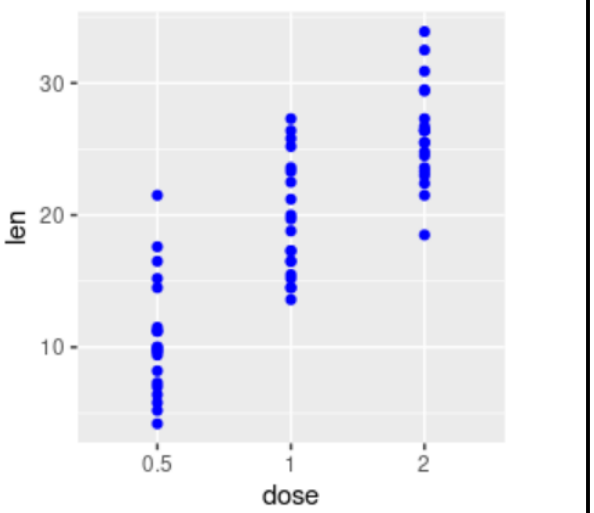
Color by group:
Color assigned in **aes()**

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point()
```



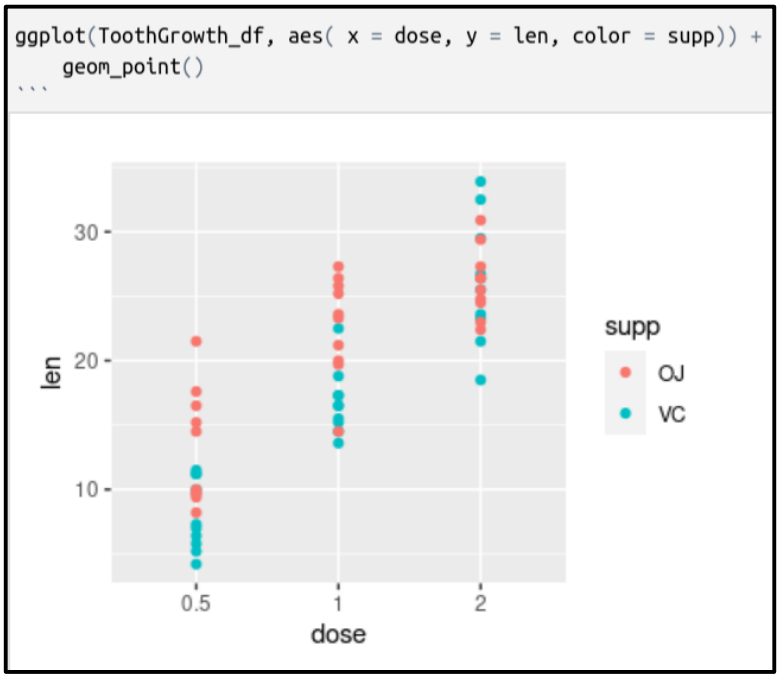
Color all data points:
Color assigned in **geom()**

```
ggplot(ToothGrowth_df, aes( x = dose, y = len)) +
    geom_point( color = "blue")
```



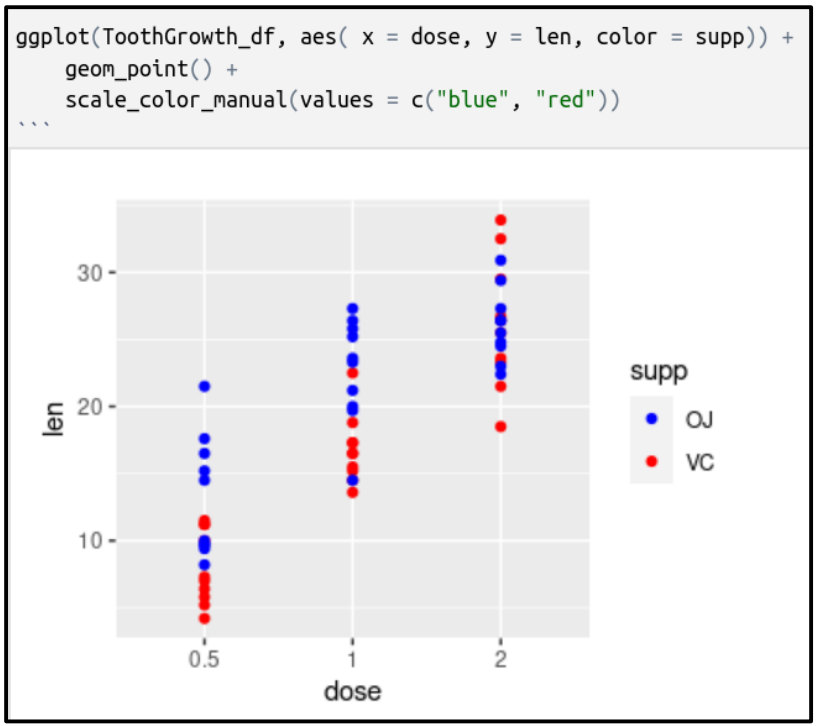
Color assigned in **aes()**

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point()
```



Customize colors in
scale_color_manual(values = c("…"))

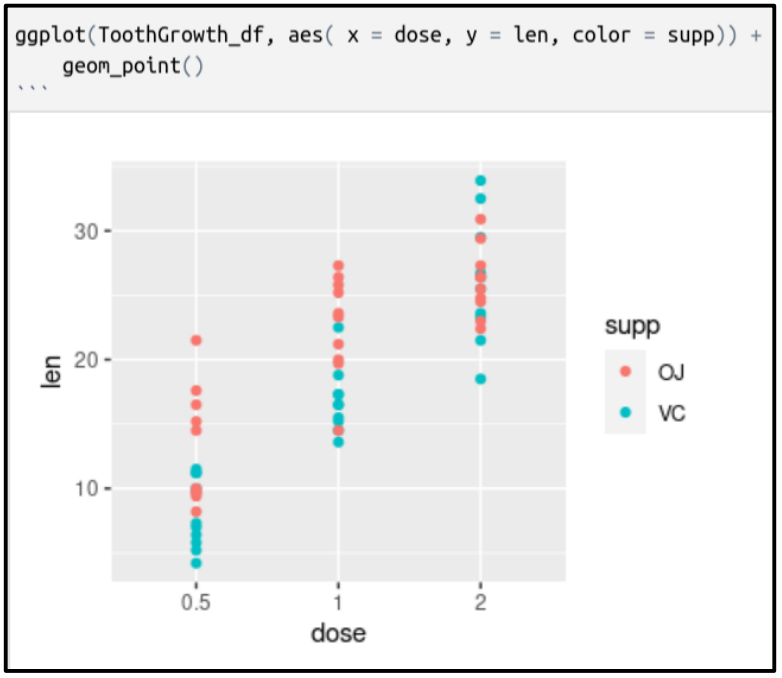
```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point() +
    scale_color_manual(values = c("blue", "red"))
```



Also see **scale_fill_manual(values = c("…"))**
Number of values ≥ number of unique "supp" values
Order of "supp" values = factor level

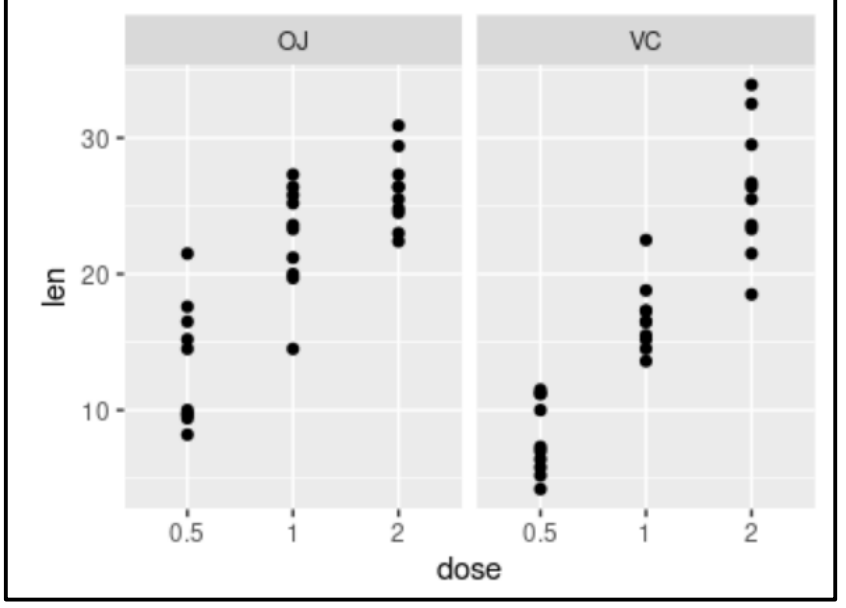
Color assigned in **aes()**

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point()
```



facet_wrap(~ col_name)

```
ggplot(ToothGrowth_df, aes( x = dose, y = len)) +
    geom_point() +
    facet_wrap( ~ supp)
```

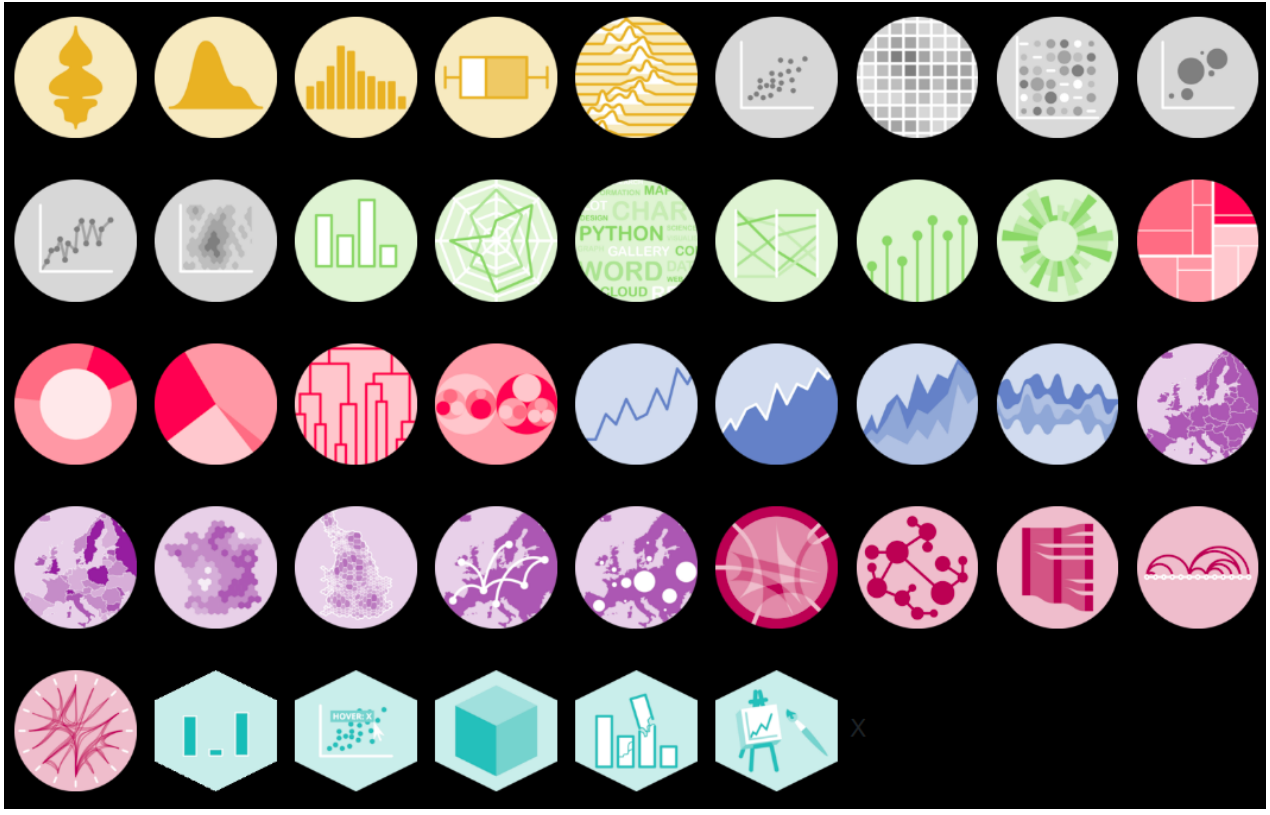


Also see **facet_grid(**col_name **~** col_name**)**

```
ggplot(df, aes(x=…, y=…, color = …, fill = …)) +
    geom_point()
```

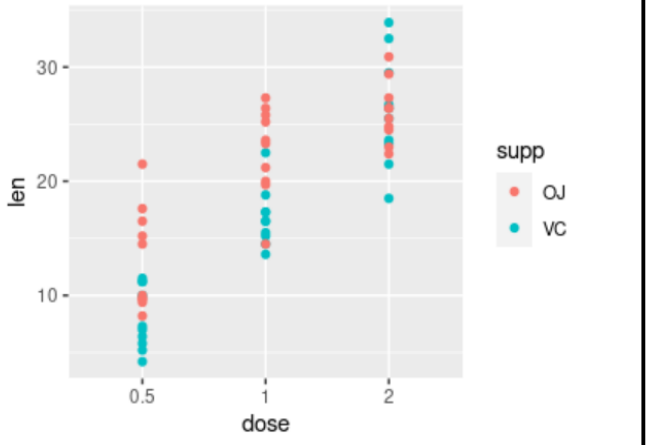
geom_point generates a dot plot, but there are many other geoms/graphs

Other types of graphs: https://www.r-graph-gallery.com/



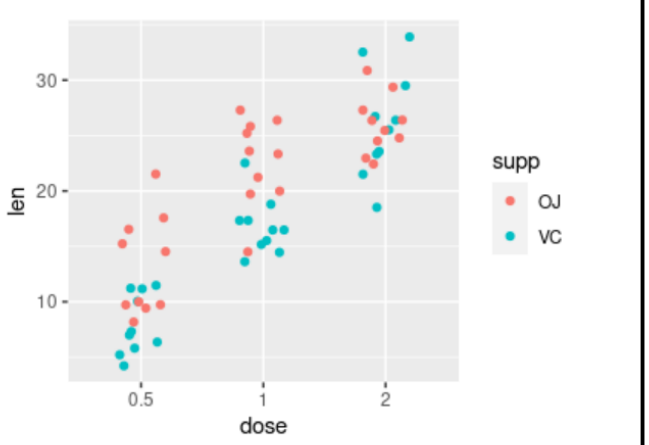
geom_point()

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point()
```



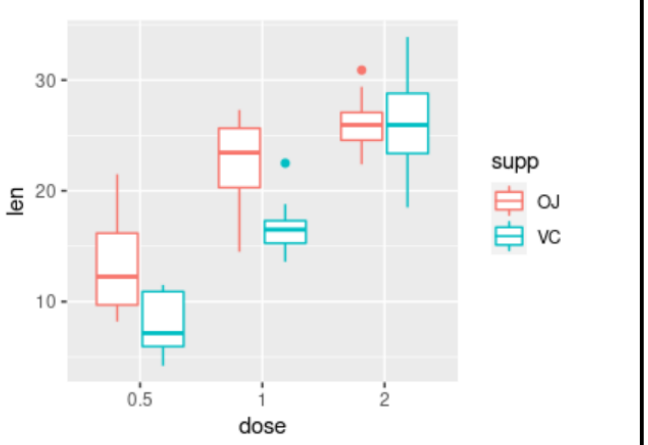
geom_jitter()

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_jitter(position = position_jitter(.2))
```



geom_boxplot()

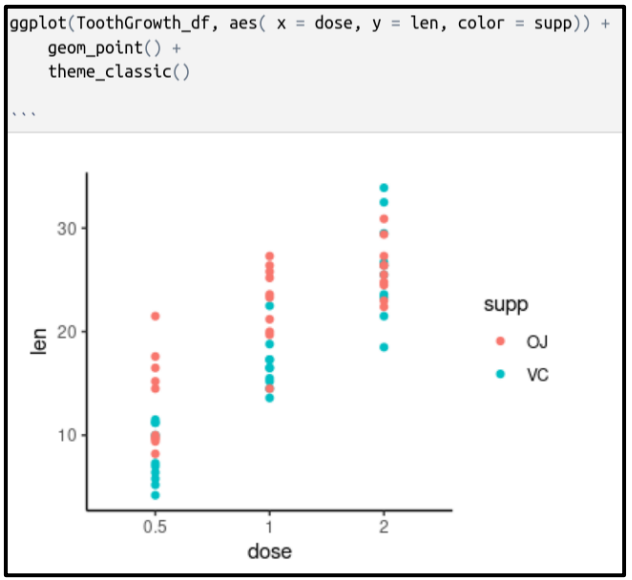
```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_boxplot()
```



```
ggplot(df, aes(x=…, y=…, color = …, fill = …)) +
    geom_point() +
    theme_classic()
```

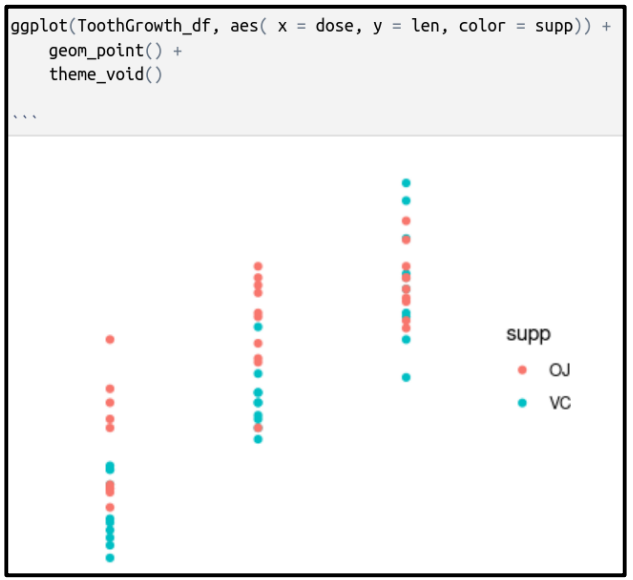
theme_classic()

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point() +
    theme_classic()
```



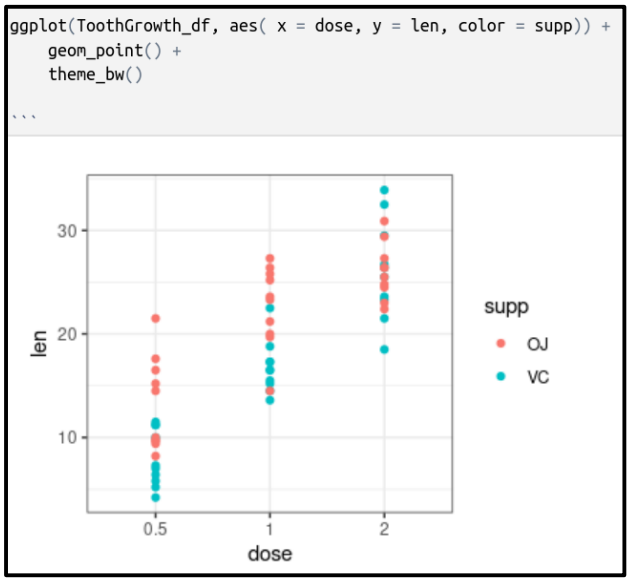
theme_void()

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point() +
    theme_void()
```



theme_bw()

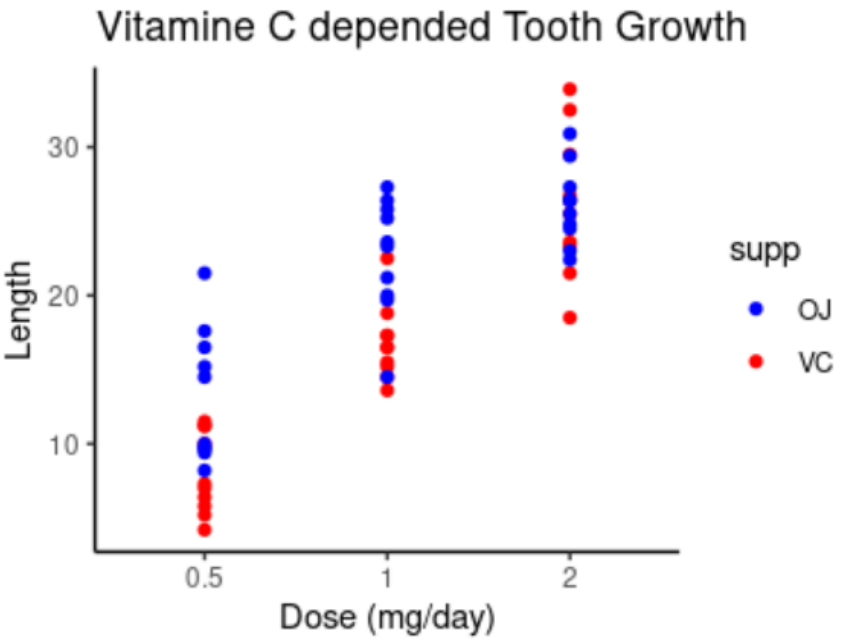
```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_point() +
    theme_bw()
```



Other themes: https://ggplot2.tidyverse.org/reference/ggtheme.html

```
ggplot(df, aes(x=…, y=…, color = …, fill = …)) +
      geom_point() +
      theme_classic() +
      labs(x = "…", y = "…" , title = "...")
```

```
labs(x="Dose (mg/day)",
     y = "Length",
     title = "Vitamine C depended Tooth Growth")
```



```
ggplot(data, aes(x=…, y=…, color = …, fill = …)) +
    geom_point() +
    theme_classic() +
    labs(x = "…", y = "…" , title = "…") +
    theme(<args>)
```

themes() allows you to customize text options and more!

All text on graph can be customized using **theme**()

```
theme(
        plot.title = element_text(hjust = 0.5, size = 14, color = “pink”),
        axis.title = element_text(size = 10, color="black"),
        axis.text = element_text(size = 10, color="black"),
        axis.ticks =  element_line(color="red" )   )
```

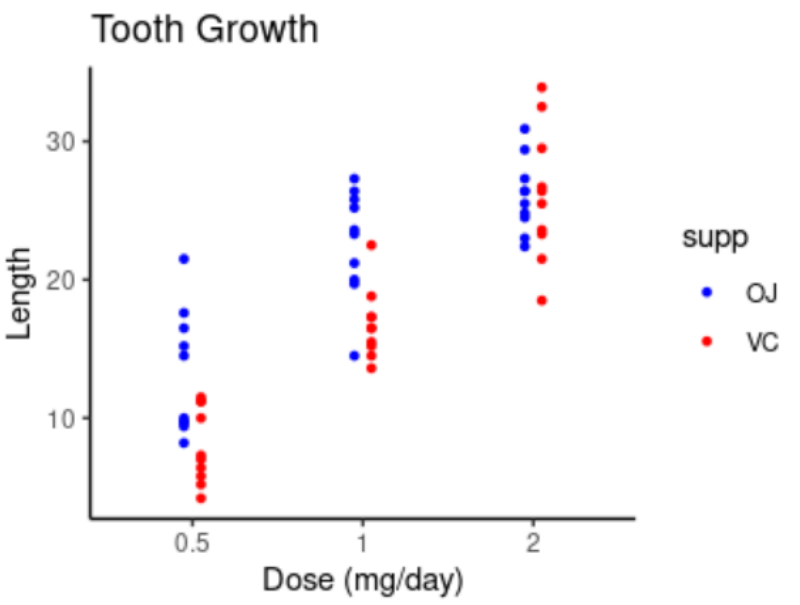
axis customization

```
theme(
        axis.text.x = element_text(size = 10, color="black"),
        axis.text.y = element_text(size = 10, color="black"),
        axis.title.x = element_text(size = 10, color="black"),
        axis.title.y = element_text(size = 10, color="black")   )
```

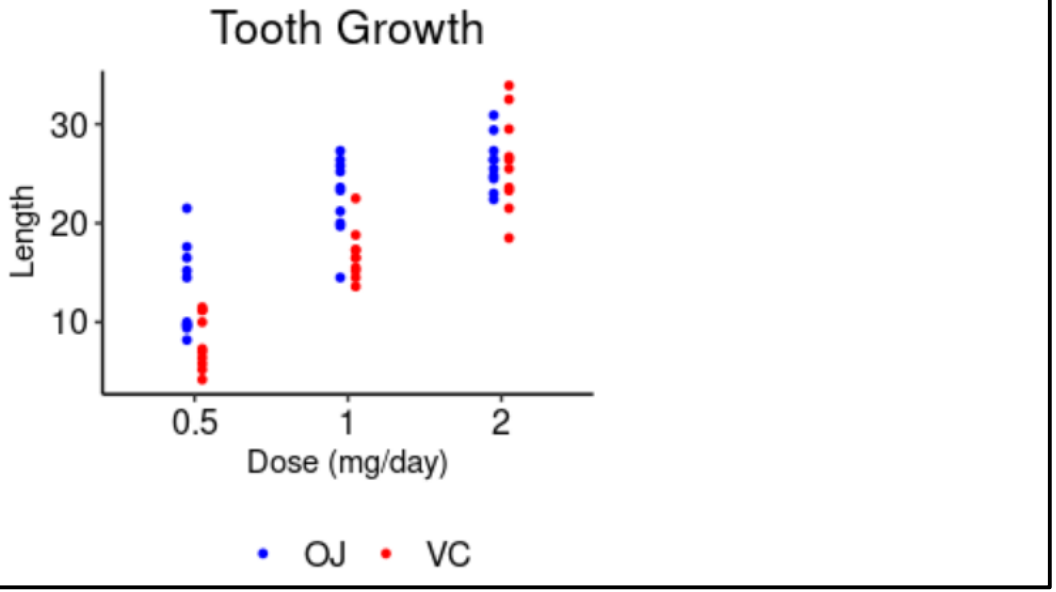
legend customization http://www.sthda.com/english/wiki/ggplot2-legend-easy-steps-to-change-the-position-and-the-appearance-of-a-graph-legend-in-r-software

```
theme(legend.position = c(5,5) ,  # coordinate position of legend
        legend.title = element_text(size=10),
        legend.text = element_text(size=10)   )
```

```
ggplot(ToothGrowth_df, aes( x = dose, y = len, color = supp)) +
    geom_jitter(size = 1, position = position_dodge(0.2)) +
    theme_classic() +
    scale_color_manual(values=c("blue", "red")) +
    labs(x = "Dose (mg/day)",  y = "Length",
         title = "Tooth Growth")
```




```
   theme(plot.title = element_text(hjust = 0.5, color="black", size = 16),
          axis.text = element_text(color="black", size=12),
          legend.position = "bottom",
          legend.title = element_blank(),
          legend.text = element_text(size=12)
         )
```



Save your graph

https://ggplot2.tidyverse.org/reference/ggsave.html

```
ggsave(plot = <plot>,       #default is last ggplot
       filename = "plot.pdf"
       path = "./path/to/destination/",
       device = "pdf",    #png, jpeg, ...
       width = <inches>,
       height = <inches>,
       dpi =  <dpi>        #quality, dots per inch
```

Save your graph

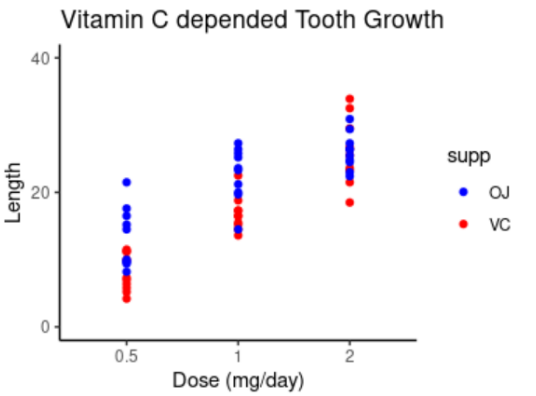
```
ggsave(filename = "plot.pdf",      #you are able to skip the first argument
       path = "./figures",
       device = "pdf",
       width = 4,
       height = 3,
       dpi =  200 )
```

Other useful options:

- Change axis break points and limits
  scale_x_continuous(**limits**=c(0,100), **breaks** = (0,25,50,100))
  The x axis will have limits with a lower boud of 0 and upper bound of 100
  The x axis tick marks will include 0, 25, 50, 75, and 100.

  Also see **scale_y_continuous**()

```
scale_y_continuous(limits = c(0,40), breaks =c(0,20,40)
```



- Add text, shapes and lines to graphs
  https://ggplot2.tidyverse.org/reference/annotate.html
  annotate("text", **x** = 1, **y** = 25, **label** = "Some text")
  Will add text at coordinate positions 1,25 (x,y coordinates) that will say “Some text”

```
annotate("text", x = 1, y = 25, label = "Some text")
```