

# Intro to Data Visualization and Statistics in R

## Session #2

Genome Institute of Singapore

3rd October 2019

## RECAP SESSION 1 & ASSIGNMENT

# What we learnt last week

## Programming aspects:

- ▶ R and Rstudio
- ▶ R as a calculator
- ▶ Scalars and assignment
- ▶ Scripting
- ▶ Control flow

## Visualization:

- ▶ continuous vs categorical (histogram, density, boxplot)
- ▶ continuous vs continuous (scatterplots)

## Statistics:

- ▶ Exponential

# Learning objectives for Session 2

## Objectives:

1. Review of assignment + extensions
2. `tabyl()` and `melt()` commands
3. Learn tidyverse grammar
4. Confounding

## Exercise 1: Road crossing example

Let's assume you have 5% chance of a fatal accident when crossing a road. What are your chances of surviving ten such crossings?

## Exercise 1: Road crossing example

Let's assume you have 5% chance of a fatal accident when crossing a road. What are your chances of surviving ten such crossings?

### Solution:

Let  $p$  be the probability of death crossing a road.  
So the probability of surviving a road is  $1 - p$ .

Prob. of surviving 10 road crossing  
= Prob. of surviving 1st road x  
  Prob. of surviving 2nd road x  
  ... x  
  Prob. of surviving 10th road  
=  $(1-p)^{10}$

Thus the probability of surviving all ten crossings is:

```
p <- 0.05  
(1-p)^10  
## [1] 0.5987369
```

## Exercise 1: Road crossing example | Extension

40% chance of **NOT** surviving 10 crossings? WTH.

You convinced the Minister for Transport to improve road safety so that the death rate for 10 crossings reduces to just 5%. Yippee!

What is now your new per-crossing death rate ( $p_{new}$ )?

## Exercise 1: Road crossing example | Extension

40% chance of **NOT** surviving 10 crossings? WTH.

You convinced the Minister for Transport to improve road safety so that the death rate for 10 crossings reduces to just 5%. Yippee!

What is now your new per-crossing death rate ( $p_{new}$ )?

$$\text{Prob. of surviving 1 crossing} = 1 - p_{new}$$

$$\text{Prob. of surviving 10 crossing} = (1 - p_{new})^{10}$$



## Exercise 1: Road crossing example | Extension

40% chance of **NOT** surviving 10 crossings? WTH.

You convinced the Minister for Transport to improve road safety so that the death rate for 10 crossings reduces to just 5%. Yippee!

What is now your new per-crossing death rate ( $p_{new}$ )?

$$\text{Prob. of surviving 1 crossing} = 1 - p_{new}$$

$$\text{Prob. of surviving 10 crossing} = (1 - p_{new})^{10}$$

$$(1 - p_{new})^{10} = 0.95$$

## Exercise 1: Road crossing example | Extension

40% chance of **NOT** surviving 10 crossings? WTH.

You convinced the Minister for Transport to improve road safety so that the death rate for 10 crossings reduces to just 5%. Yippee!

What is now your new per-crossing death rate ( $p_{new}$ )?

$$\text{Prob. of surviving 1 crossing} = 1 - p_{new}$$

$$\text{Prob. of surviving 10 crossing} = (1 - p_{new})^{10}$$

$$(1 - p_{new})^{10} = 0.95$$

$$1 - p_{new} = 0.95^{1/10}$$

$$1 - p_{new} = 0.9948838$$

$$p_{new} = 0.0051$$

## Exercise 1: Road crossing example | Extension

40% chance of **NOT** surviving 10 crossings? WTH.

You convinced the Minister for Transport to improve road safety so that the death rate for 10 crossings reduces to just 5%. Yippee!

What is now your new per-crossing death rate ( $p_{new}$ )?

$$\text{Prob. of surviving 1 crossing} = 1 - p_{new}$$

$$\text{Prob. of surviving 10 crossing} = (1 - p_{new})^{10}$$

$$(1 - p_{new})^{10} = 0.95$$

$$1 - p_{new} = 0.95^{1/10}$$

$$1 - p_{new} = 0.9948838$$

$$p_{new} = 0.0051$$

You can use a simple approximation: final  $p$  / number of crossings.  
In this case, the approximation  $0.05/10$  gives 0.005.

## Basic probability | Multiple hypothesis testing

The road crossing example is an illustration of the **multiple hypothesis testing** problem.

The correction method  $p/n_{test}$  is the **Bonferroni method**.

Can you name an example from genomics research?

## Basic probability | Multiple hypothesis testing

The road crossing example is an illustration of the **multiple hypothesis testing** problem.

The correction method  $p/n_{test}$  is the **Bonferroni method**.

Can you name an example from genomics research?

### **GWAS study example**

HapMap consortium used the ENCODE data to show  $\sim 1$  million independent chromosomal segments (among Europeans).

Therefore,  $5 * 10^{-8}$  ( $= 0.05/1,000,000$ ) is frequently taken as genomewide significance threshold.

## Basic probability | Multiple hypothesis testing

The road crossing example is an illustration of the **multiple hypothesis testing** problem.

The correction method  $p/n_{test}$  is the **Bonferroni method**.

Can you name an example from genomics research?

### **GWAS study example**

HapMap consortium used the ENCODE data to show  $\sim 1$  million independent chromosomal segments (among Europeans).

Therefore,  $5 * 10^{-8}$  ( $= 0.05/1,000,000$ ) is frequently taken as genomewide significance threshold.

**Note:** Bonferroni is a very stringent procedure. Thus for transcriptomics, the **false discovery rate (FDR)** method is more popular as it can tolerate some false positives.

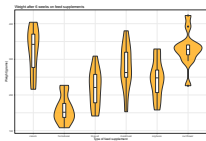
## Exercise 2: Chick weight | Solution

```
setwd("C:/Users/aramasamy/Desktop/R_workshop") # change
pacman::p_load(tidyverse, readxl, gridExtra, janitor)

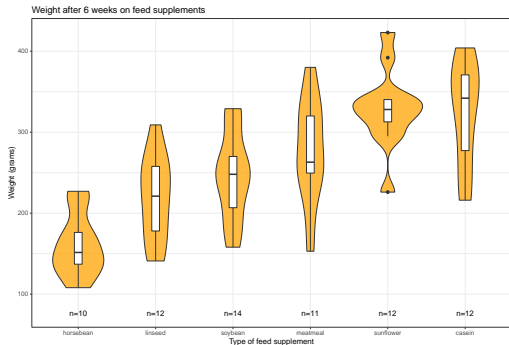
cw <- read_excel("data/session1_data.xlsx", sheet="chick_weight")

mylabs <- labs(
  title="Weight after 6 weeks on feed supplements",
  caption="McNeil (1977) Interactive Data Analysis",
  x="Type of feed supplement", y="Weight (grams)")

ggplot(cw, aes(x=feed, y=weight)) +
  geom_violin(fill="orange", alpha=0.75) +
  geom_boxplot(width=0.1) + mylabs + theme_bw()
```



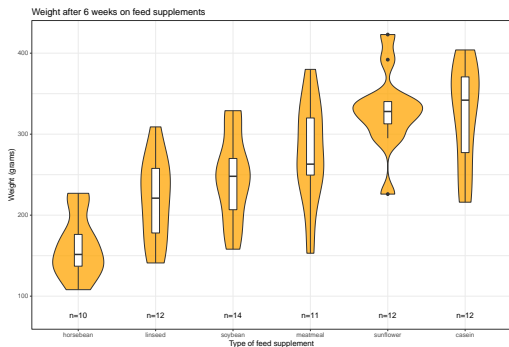
## Exercise 2: Chick weight | Better solution



McNeil (1977) Interactive Data Analysis



## Exercise 2: Chick weight | Better solution



McNeil (1977) Interactive Data Analysis

```
pacman::p_load(EnvStats)    ## for stat_n_text() function

ggplot(cw, aes(x=reorder(feed, weight, median), y=weight))
  geom_violin(fill="orange", alpha=0.75) +
  geom_boxplot(width=0.1) +
  stat_n_text() + mylabs + theme_bw()
```

## Take home assignment | General points

1. Remember to `setwd()` and to load packages at the start of your script.
2. Please use **consistent spacing**.
3. **Highly recommend** using space around the assignment operator.
4. Be careful of excessive line widths

Compare

```
chick_weight<-read_excel("data/session1_data.xlsx",sheet="chick_weight")
```

with

```
chick_weight <- read_excel("data/session1_data.xlsx",  
                           sheet="chick_weight")
```

## Exercise 3: Height vs weight by Gender

**Background:** 10,000 measurements of height and weight for men and women.

**Source:** Machine Learning for Hackers, Drew Conway & John Myles-White, O'Reilly Media.

**Exercise:** Read in the data from the height\_weight sheet in the session1\_data.xlsx file. Name your script **height\_weights.R**

1. Reproduce the following graph as closely as possible. Hint: `grid.arrange()`
2. Does anything look strange?
3. How many men and women are there in this dataset?

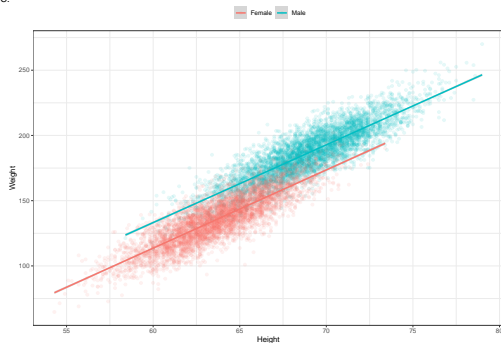
## Exercise 3: Height vs weight by Gender (Solution part 1)

Let us read in the data and make the density plot for height and weight. Use the script height\_weights.R

```
hw <- read_excel("data/session1_data.xlsx",  
                 sheet="height_weight")  
  
g.A <- ggplot(hw, aes(x=Height, fill=Gender)) +  
  geom_density(alpha=0.4) +  
  labs(x="", y="", title="\nHeight", tag="A.") +  
  theme_bw() + theme(legend.position="none")  
  
g.B <- ggplot(hw, aes(x=Weight, fill=Gender)) +  
  geom_density(alpha=0.4) +  
  labs(x="", y="", title="\nWeight", tag="B.") +  
  theme_bw() + theme(legend.position="none")
```

## Exercise 3: Height vs weight by Gender (Solution part 2)

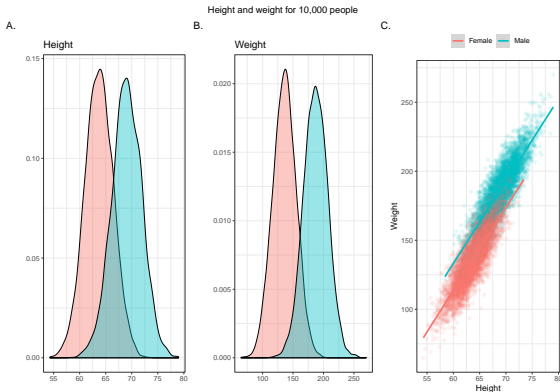
C.



```
g.C <- ggplot(hw, aes(x=Height, y=Weight, col=Gender)) +  
  geom_point(alpha=0.1) +  
  geom_smooth(method="lm") +  
  labs(x="Height", y="Weight", tag="C.", col=NULL) +  
  theme_bw() + theme(legend.position="top")  
print(g.C)
```

## Exercise 3: Height vs weight by Gender (Solution part 3)

```
grid.arrange(g.A, g.B, g.C, nrow=1,  
             top="Height and weight for 10,000 people")
```



```
rm(g.A, g.B, g.C)
```

**Does anything look weird?**

### Exercise 3: Height vs weight by Gender (this session)

**Height is recorded in inches. Weight is recorded in pounds.**

## Exercise 3: Height vs weight by Gender (this session)

**Height is recorded in inches. Weight is recorded in pounds.**

In this session, you will learn to

- 1) Rename the columns:
  - a) height as height\_inches
  - b) weight as weight\_pounds
- 2) Add new columns:
  - a) height\_m and weight\_kg
  - b) body mass index (BMI)
  - c) Generate BMI class according to WHO definition
- 3) Cross-table between gender and BMI class
- 4) Calculate the mean and SD of BMI by sex

In later sessions, we will learn how to estimate the slopes and intercepts from a linear regression.



## Exercise 3: Height vs weight by Gender

How many men and women in the dataset? `summary(hw)` does not help here. You could try

```
table(hw$Gender)
## Female    Male
##   5000    5000
```

Or you can use `tabyl` from the `janitor` package.

```
hw %>% janitor::tabyl(Gender)
## Gender      n percent
## Female 5000      0.5
## Male   5000      0.5
```

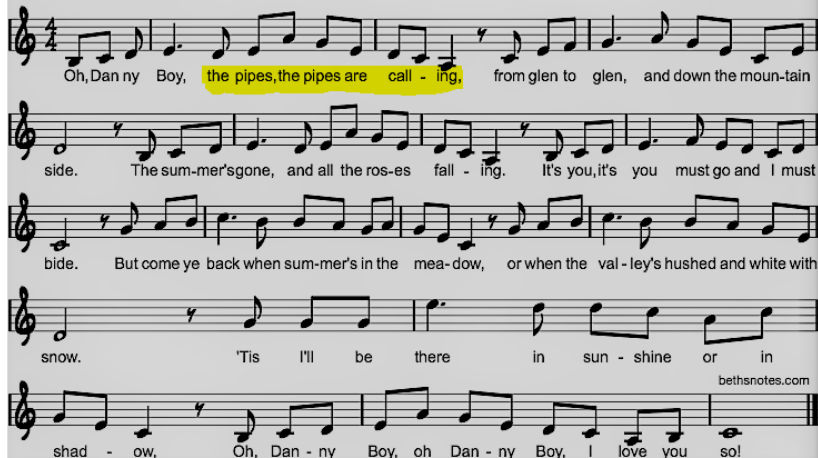
The **pipe command** (`%>%`)

- ▶ passes an intermediate result onto next function (chaining)
- ▶ without storing the intermediate result
- ▶ The keystroke “Ctrl-Shift-M” generates this symbol

# Danny Boy

Frederic Weatherly, 1910

Irish folk tune  
(Londonderry Air)



Oh, Dan ny Boy, the pipes, the pipes are call - ing, from glen to glen, and down the moun-tain

side. The sum-mer's gone, and all the ros-es fall - ing. It's you, it's you must go and I must

bide. But come ye back when sum-mer's in the mea-dow, or when the val - ley's hushed and white with

snow. 'Tis I'll be there in sun - shine or in

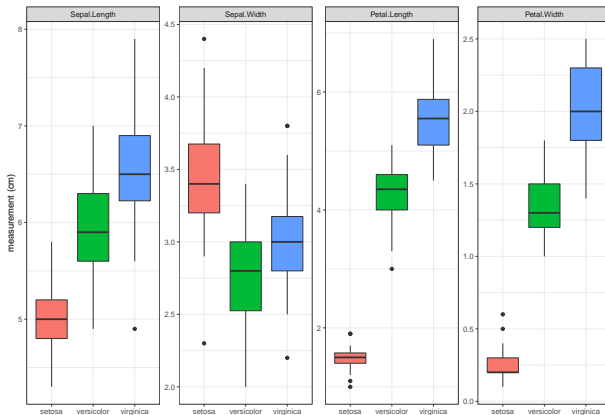
shad - ow, Oh, Dan - ny Boy, oh Dan - ny Boy, I love you so!

bethsnotes.com

MELT (from data.table package)

## Motivating example for melt

To generate the following plot, you can write four separate ggplot with `geom_boxplot()` and then combine it with `grid.arrange()`.



But this can get tedious if you had to plot many variables. This is one example where `melt()` can be very handy.

## Melt | What is it?

A tool to reshape from wide to long format.

Subject	Status	gene1	gene2	gene3
S1	Case	10.1	6.3	9.5
S2	Case	10.5	5.7	9.1
S3	Control	12	6.1	7.5

MELT



```
df %>%  
  melt(id.vars=c("Subject", "Status"))
```

Subject	Status	variable	value
S1	Case	gene1	10.1
S2	Case	gene1	10.5
S3	Control	gene1	12
S1	Case	gene2	6.3
S2	Case	gene2	5.7
S3	Control	gene2	6.1
S1	Case	gene3	9.5
S2	Case	gene3	9.1
S3	Control	gene3	7.5

To reshape from long to wide format, use `dcast()`.

## melt() mini demonstration with Iris data

Let's use the first two rows of iris for demonstration:

```
head(iris, 2) # 2 rows
#Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#           5.1           3.5           1.4           0.2 setosa
#           4.9           3           1.4           0.2 setosa
```

```
head(iris, 2) %>% melt(id.vars="Species") # 8 rows
#   Species      variable value
# 1  setosa Sepal.Length    5.1
# 2  setosa Sepal.Length    4.9
# 3  setosa Sepal.Width     3.5
# 4  setosa Sepal.Width     3.0
# 5  setosa Petal.Length    1.4
# 6  setosa Petal.Length    1.4
# 7  setosa Petal.Width     0.2
# 8  setosa Petal.Width     0.2
```

## Iris dataset | Boxplot for all 4 variables

Here are the codes to generate plot for four variables presented three slides earlier.

```
pacman::p_load(data.table)  ## for melt()

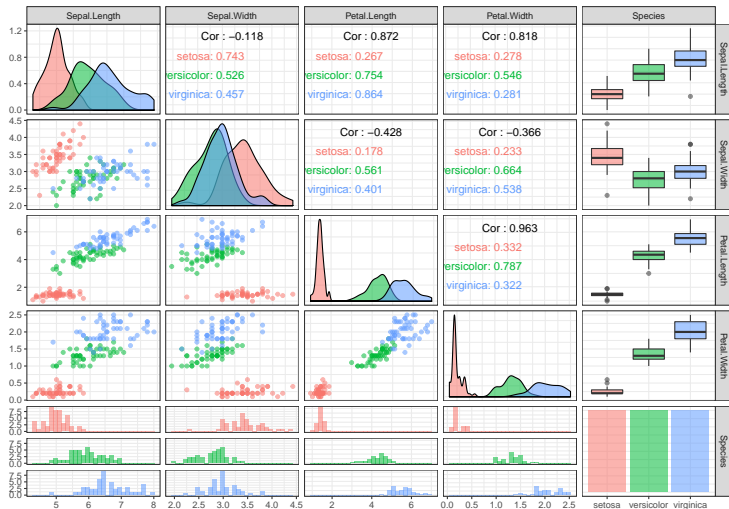
mdf <- melt(iris, id.vars="Species")
dim(mdf)  ## 600 rows = 150 samples by 4 variables
# View(mdf)

ggplot(mdf, aes(x=Species, y=value, fill=Species)) +
  geom_boxplot() +
  facet_wrap(~ variable, scales="free_y", nrow=1) +
  labs(x="", y="measurement (cm)") +
  theme_bw() + theme(legend.position="none")
```

What does the scales="free\_y" do?

# Iris dataset | ggpairs

```
pacman::p_load(GGally)
ggpairs(iris, aes(color=Species, alpha=0.6)) + theme_bw()
```





# Iris dataset | skimr

You can summarize every column in the data frame concisely:

```
pacman::p_load(skimr)
skim(iris)
```

Skim summary statistics

n obs: 150

n variables: 5

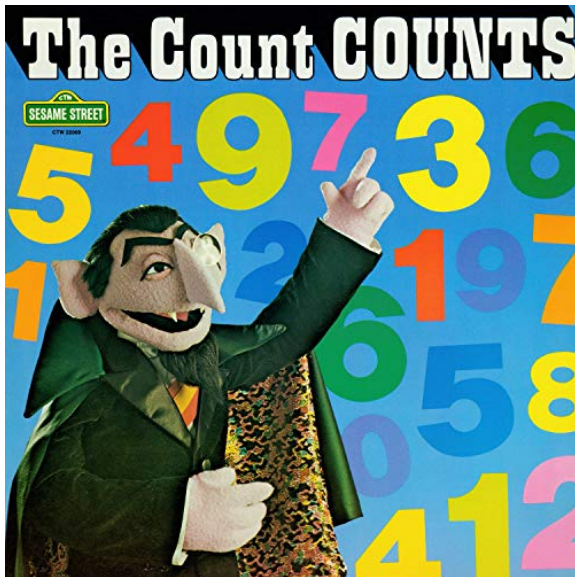
```
-- Variable type:factor -----
variable missing complete  n n_unique          top_counts ordered
Species      0       150 150           3 set: 50, ver: 50, vir: 50, NA: 0  FALSE
```

```
-- Variable type:numeric -----
variable missing complete  n mean  sd  p0 p25  p50 p75 p100      hist
Petal.Length    0       150 150 3.76 1.77 1   1.6 4.35 5.1  6.9  [
Petal.Width     0       150 150 1.2  0.76 0.1 0.3 1.3  1.8  2.5  [
Sepal.Length    0       150 150 5.84 0.83 4.3 5.1 5.8  6.4  7.9  [
Sepal.Width     0       150 150 3.06 0.44 2   2.8 3   3.3  4.4  [
```

TABYL (from janitor package)

# Counting

A lot of data science is about counting.



## One-way tabyl

Let's use the chick\_weight example here:

```
cw %>% tabyl(feed)
##      feed  n  percent
## casein  12 0.1690141
## horsebean 10 0.1408451
## linseed  12 0.1690141
## meatmeal 11 0.1549296
## soybean  14 0.1971831
## sunflower 12 0.1690141
```

If you have missing values in the column, you will also get a "valid\_percent" column in the output.

## Aside | Quantiles

Quantiles are cut points dividing the **sorted continuous data** into parts containing equal numbers of samples.

Famous quantiles include:

N	Name
2	Median
3	Terciles
4	Quartiles
10	Deciles
100	Percentile

In the next slide, we will calculate the terciles and group the data points by their terciles.

## Aside | Terciles

Let's use chick\_weight for demo. First calculate the terciles:

```
quantile( cw$weight, c(1/3, 2/3) )  
## 33.33333% 66.66667%  
## 226.3333 313.0000
```

## Aside | Terciles

Let's use `chick_weight` for demo. First calculate the terciles:

```
quantile( cw$weight, c(1/3, 2/3) )  
## 33.33333% 66.66667%  
## 226.3333 313.0000
```

Let's categorize the original weight by their terciles. You can use multiple nested `ifelse()` but it is simpler to use `cut()`

```
cw$weight_cat <- cut(cw$weight,  
                     breaks = c(-Inf, 226.3, 313.0, Inf),  
                     labels = c("low", "mid", "high") )  
table(cw$weight_cat)  
## low mid high  
## 24 23 24
```

1. I am using the base R functions (`$`, `table`) here.
2. Number of breakpoints is one more than the number of labels

## Two-way tabyl | basic version

Let's do a two-way tabulation between feed and weight category:

```
tab <- cw %>% tabyl(feed, weight_cat)
print(tab)
```

```
##           feed low mid high
## 1    casein    2   2   8
## 2 horsebean    9   1   0
## 3   linseed    6   6   0
## 4  meatmeal    2   5   4
## 5   soybean    4   7   3
## 6 sunflower    1   2   9
```

Note: `table( cw$feed, cw$weight_cat )` will give you exactly the same answer, but it will not be able to do the fancy steps in next slide very easily.



## Two-way tabyl | pretty version

feed	low	mid	high	Total
horsebean	9 (90%)	1 (10%)	0 (0%)	10 (100%)
linseed	6 (50%)	6 (50%)	0 (0%)	12 (100%)
soybean	4 (29%)	7 (50%)	3 (21%)	14 (100%)
meatmeal	2 (18%)	5 (45%)	4 (36%)	11 (100%)
casein	2 (17%)	2 (17%)	8 (67%)	12 (100%)
sunflower	1 (8%)	2 (17%)	9 (75%)	12 (100%)

## Two-way tabyl | pretty version

feed	low	mid	high	Total
horsebean	9 (90%)	1 (10%)	0 (0%)	10 (100%)
linseed	6 (50%)	6 (50%)	0 (0%)	12 (100%)
soybean	4 (29%)	7 (50%)	3 (21%)	14 (100%)
meatmeal	2 (18%)	5 (45%)	4 (36%)	11 (100%)
casein	2 (17%)	2 (17%)	8 (67%)	12 (100%)
sunflower	1 (8%)	2 (17%)	9 (75%)	12 (100%)

```
tab %>%  
  arrange(high, mid, low) %>%  
  adorn_totals("col") %>%  
  adorn_percentages("row") %>%  
  adorn_pct_formatting(digits=0) %>%  
  adorn_ns(position="front") %>%  
  knitr::kable()
```

# TIDYVERSE GRAMMAR

## Six common verbs (dplyr)

Verb	Explanation
<code>select()</code>	picks <b>columns</b> based on their names.
<code>filter()</code>	picks <b>rows</b> based on their values.
<code>group_by()</code>	groups the data based on one or more variables.
<code>summarize()</code>	reduces multiple values down to a single summary.
<code>mutate()</code>	<b>adds new columns</b> using existing columns.
<code>arrange()</code>	changes the <b>ordering</b> of the rows.

## select() | Positive column selection

Try the following:

```
tmp <- iris %>% head(3)  ## for illustration

## Basic select. Use comma for multiple variables.
tmp %>% select( Sepal.Length, Sepal.Width, Species )

## Selecting by patterns
tmp %>% select( starts_with("Sepal"), Species )
tmp %>% select( ends_with("Length"), contains("Sp") )

## Select and rename
tmp %>% select( SL=Sepal.Length, Type=Species )

## Rename only
tmp %>% rename( SL=Sepal.Length, Type=Species )
```

## select() | Negative column selection

Sometimes easier to drop columns than specify which ones to keep:

```
tmp %>% select( -Species )
```

**Do not mix positive and negative selection** in one select() command. For example, the following gives different answers:

```
tmp %>% select( Sepal.Length, -Species )  
tmp %>% select( -Species, Sepal.Length )  
rm(tmp)
```

## filter() | Row selection

Try out the following:

```
iris %>%  
  filter(Species=="setosa")
```

```
iris %>%  
  filter(Sepal.Length < 5)
```

```
iris %>%  
  filter(Sepal.Length < 6, Sepal.Width > 3)
```

Pipe the outputs to `dim()`, `count()`, `glimpse()` if you want to get a quick idea on how many rows are being outputted.

Last command has multiple conditions in the filter. Only data points passing **all** of the commands are included.

You can pipe as many times as you like

```
iris %>%  
  select(SL = Sepal.Length, SW = Sepal.Width) %>%  
  filter(SL < 5) %>%  
  summarize( mean(SW) )  
##      mean(SW)  
## 1 3.077273
```

```
iris %>%  
  filter(Sepal.Length < 6, Sepal.Width > 3) %>%  
  tabyl(Species)  
##      Species  n    percent  
##      setosa  42 0.97674419  
## versicolor   1 0.02325581  
##  virginica   0 0.00000000
```

**but the changes are not saved without assignment.**

i.e. `filter(SL < 6, SW > 3)` will not work in 2nd line.



## summarize() or summarise() | Generate summaries

Try out the following:

```
iris %>%  
  summarize( mean(Sepal.Length) )
```

```
iris %>%  
  summarize( SL_mean = mean(Sepal.Length),  
            SW_mean = mean(Sepal.Width),  
            PL_mean = mean(Petal.Length),  
            PW_mean = mean(Petal.Width) )
```

```
iris %>%  
  select( -Species ) %>%  
  summarize_all( mean() )
```

summarize\_all applies the function mean() to all columns.

## group\_by() | Exercise

In the previous slide, you calculated the averages for all 150 flowers. Use `group_by()` to calculate these averages by Species.

```
iris %>%  
  group_by( Species ) %>%  
  summarize( SL_mean = mean(Sepal.Length) )
```

```
iris %>%  
  group_by( Species ) %>%  
  summarize_all( mean )
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

## mutate() | Add columns based on existing columns

Suppose we want to calculate the area of the sepals and petals. If leaves are rectangular shaped, then  $\text{size} = \text{length} \times \text{width}$ .

```
out <- iris %>%  
  
  select(SL=Sepal.Length, SW=Sepal.Width,  
         PL=Petal.Length, PW=Petal.Width, Species) %>%  
  
  mutate(Sepal_Area = SL*SW,  
         Petal_Area = PL*PW)  
  
head(out, 5)
```

	SL	SW	PL	PW	Species	Sepal_Area	Petal_Area
## 1	5.1	3.5	1.4	0.2	setosa	17.85	0.28
## 2	4.9	3.0	1.4	0.2	setosa	14.70	0.28
## 3	4.7	3.2	1.3	0.2	setosa	15.04	0.26
## 4	4.6	3.1	1.5	0.2	setosa	14.26	0.30
## 5	5.0	3.6	1.4	0.2	setosa	18.00	0.28

## Height\_weight | Exercise to bring it all together

- 1) Rename the columns:
  - a) Height as Height\_inches
  - b) Weight as Weight\_pounds
  
- 2) Add new columns:
  - a) Height\_m and Weight\_kg
  - b) body mass index (BMI)
  - c) Generate BMI class according to WHO definition:
    - ▶ underweight if BMI < 18.5
    - ▶ normal weight if BMI is between 18.5 – 24.9
    - ▶ overweight if BMI is between 25 – 29.9
    - ▶ obese BMI is  $\geq 30$
  
- 3) Cross-table between gender and BMI class
  
- 4) Calculate the mean and SD of BMI by sex

# Capstone project

Aim: To practice what you learnt so far.

- 1) Work in teams of 3 - 4 people.
- 2) A simple project based on real life data (either your own or public data) that requires summarization and visualization.
- 3) Gokce and myself can work with you from time to time and comment on improving your codes until final presentation.
- 4) Present in session #6 (~ 10min to present + 5min discussion).
- 5) Share scripts with team at least 2 days before presentation.  
Data sharing is optional.