BIOC461

Research Design and Analysis in Biochemistry

Lecture 1

Mik Black & Paul Gardner
Department of Biochemistry

7 April 2022

Todays lecture

1. Good practice for project organisation
2. Data organisation (and visualisation) in practice
3. Statistical testing
4. Power calculations

Part 1: Project Organisation

Organising your projects

Based on the papers:

- **Good enough practices in scientific computing**:
  Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK. PLoS Comput Biol. 2017 Jun 22;13(6):e1005510.
  doi: 10.1371/journal.pcbi.1005510

- **Ten Simple Rules for Reproducible Computational Research**
  Sandve GK, Nekrutenko A, Taylor J, Hovig E. PLoS Comput Biol. 2013 Oct;9(10):e1003285.
  doi: 10.1371/journal.pcbi.1003285

Not just for computational research…

- Yes, both those papers are focused on “computational research”.
- BUT, most research we do contains the same basic elements:
  - experiments
  - data
  - analysis methods
  - results
  - collaboration
- We need to keep all of these components ORGANIZED.

Overview

Sections (we'll cover the blue stuff):

- Data Management
- Software
- Collaboration
- Project Organization
- Keeping Track of Changes
- Manuscripts
- Ten Rules for Reproducible Computational Research

Reproducibility + Efficiency + Continuity

- For you:
  - recreate results (with ease)
  - reuse code and workflows
  - develop more efficient work habits
- For everyone:
  - ensure reproducibility (and reduce burden)
  - establish workflows
  - enable collaboration
  - build upon existing work

Quiz time…

Not really, just some thinking

Spend two minutes thinking about the answers to the following questions:

1. If you were abducted by aliens in the next 30 seconds, how easy would it be for someone else to carry on your research? (after we finished mourning your loss, of course).

2. How much time would you need to spend getting your work to the point where another human being *could* carry on with your project? What would this involve?

3. If you laptop/desktop was unexpectedly destroyed by evil anti-research forces, what impact would this have on your project? (it's ok, we'll buy you a shiny new one from our "anti-anti-research forces" insurance fund).

Lets (briefly) discuss your answers…

Data management

1. Save the raw data.
   - preserve raw data - never edit
   - if needed, computationally create modified versions of raw data (e.g., data cleaning).
2. Ensure raw data are backed up in more than one location
   - e.g., Biochem server + ITS High Capacity Storage, Biochem Server + Google Drive etc
   - **Data sensitivity** issues with cloud-based storage providers
3. Create the data you wish to see in the world
   - non-proprietary (preferably text-based) formats
   - informative variable and file names
   - make it the data set you wish you had received…

Data management

4. Create analysis-friendly data

   - each column is a variable
   - each row is an observation
   - tidy data aren't just for R…

5. Record all the steps used to process data.

   - if possible, use scripts to capture (and automate) every step of data processing and cleaning

Software (scripts, code etc)

1. Place a brief explanatory comment at the start of every program.
   - explain what the code does, and how to use it.
   - if approriate, include example of usage.
2. Decompose programs into functions.
   - breaks tasks into small (well-documented) chunks
   - digestible => understandable
3. Be ruthless about eliminating duplication.
   - use functions
   - don’t copy and paste code chunks (write a function)

Software (scripts, code etc)

4. Always search for well-maintained software libraries that do what you need.
   - don’t reinvent the wheel…
5. Test libraries before relying on them.
6. Give functions and variables meaningful names.
   - use tab completion with informative names
7. Make dependencies and requirements explicit.
   - can be as simple as adding a `requirements.txt` file.

Even if you are not working as part of a team…

Even if you are not working as part of a team…

…YOU ARE!!

Collaboration

- Your project is very likely to be part of a larger research programme, and the work you are doing has the potential to contribute to other projects in the future.
- You are also collaborating with yourself: ask “present Mik” about his good friend “future Mik”.

Collaboration

1. Create an overview of your project.
   - `README.txt` or `README.md` in top-level directory
2. Create a shared “to-do” list for the project.
   - even if it is just for you…
3. Decide on communication strategies.

Quiz Time…

Whose project folder looks like this?

Find a partner…

- Spend two minutes discussing how you organise your projects.
- Think about:
  - directory structure
  - documentation
  - where different types of files are kept

This stuff is critical…

Project Organisation

1. Put each project in its own directory, which is named after the project.
2. Put text documents associated with the project in the `doc` directory.
3. Put raw data and metadata in a data directory and files generated during cleanup and analysis in a results directory.
   - critical to separate input data from derived data
   - don’t be afraid to use subdirectories to impose additional order

Project Organisation

4. Put project source code in the src directory.
    - Interpreted: R, Python
    - Compiled: C++, Fortran, Java
    - Shell scripts, SQL snippets
5. Put external scripts or compiled programs in the bin directory.
    - may not be relevant for some projects (e.g., if not using compiled code)
    - why external scripts here? Can make distinction for edited vs non-edited files: binary files and external scripts are not directly edited.
6. Name all files to reflect their content or function.

Example: project layout

```
|-- CITATION
|-- README
|-- LICENSE
|-- requirements.txt
|-- data
|    |-- birds_count_table.csv
|-- doc
|    |-- notebook.md
|    |-- manuscript.md
|    |-- changelog.txt
|-- results
|    |-- summarized_results.csv
|-- src
|    |-- sightings_analysis.py
|    |-- runall.py
```

Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK. PLoS Comput Biol. 2017 Jun 22;13(6):e1005510.

10.1371/journal.pcbi.1005510

Tools and habits

- Tools **enable** reproducible research
  - there are lots of them
  - they are not hard to learn
- Good habits **ensure** reproducible research
  - there are many good habits: each one will improve your work in some way
  - these **are** hard to learn (actually, that is not true - it is just hard to **unlearn** your bad habits)
- Getting into the habit of using these “good practice tools” as a central part of your workflow is critical
- It takes commitment, but it will be hugely beneficial

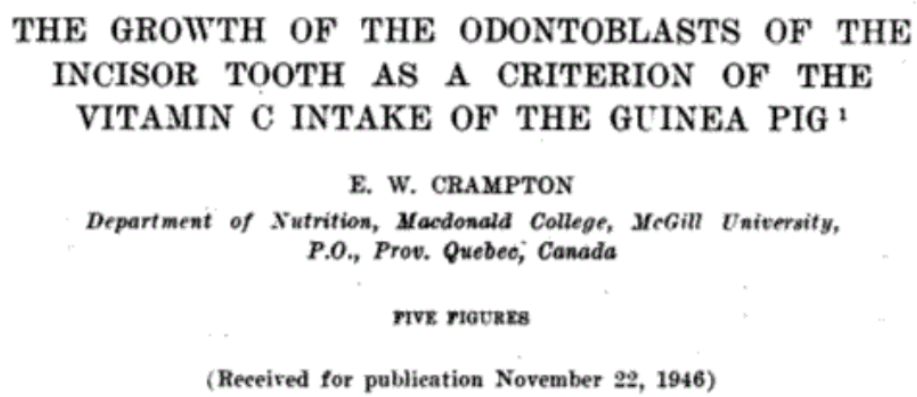
Part 2: Data Organisation

Data organization

- A spreadsheet application (e.g., Microsoft Excel) is often used to “organize” data.
  - Excel has the ability to make raw data look very pretty.
  - Statistics applications (e.g., SPSS, R, SAS, Prism) often have trouble reading in the pretty data.
- Try to keep it simple:
  - Samples in rows, variables in columns
  - No blank cells or rows (unless data are missing)

Example data

We’re going to use some really old data as an example:



THE GROWTH OF THE ODONTOBLASTS OF THE INCISOR TOOTH AS A CRITERION OF THE VITAMIN C INTAKE OF THE GUINEA PIG [1]

E. W. CRAMPTON

Department of Nutrition, Macdonald College, McGill University, P.O., Prov. Quebec, Canada

FIVE FIGURES

(Received for publication November 22, 1946)

We’ll start off looking at a fairly typical (Excel-based) approach to data organisation and analysis.

“Tooth Growth” data set

- Length of odontoblasts (cells responsible for tooth growth) in 60 guinea pigs
- Guinea pigs exposed to different doses of Vitamin C, via two different mechanisms.
- Variables in data set:
  - len = Tooth length (unit unknown…)
  - supp = Vitamin C Supplement: ascorbic acid (VC) or Orange Juice (OJ)
  - dose = Dose in milligrams/day
- Let’s look at how that data “might” get recorded in a typical study.

Tooth Growth data

Often the data would be recorded in Excel, using a format somewhat like this:

Orange Juice										
0.5mg per day	15.2	21.5	17.6	9.7	14.5	10	8.2	9.4	16.5	9.7
1.0mg per day	19.7	23.3	23.6	26.4	20	25.2	25.8	21.2	14.5	27.3
2.0mg per day	25.5	26.4	22.4	24.5	24.8	30.9	26.4	27.3	29.4	23
Vitamin C										
0.5mg per day	4.2	11.5	7.3	5.8	6.4	10	11.2	11.2	5.2	7
1.0mg per day	16.5	16.5	15.2	17.3	22.5	17.3	13.6	14.5	18.8	15.5
2.0mg per day	23.6	18.5	33.9	25.5	26.4	32.5	26.7	21.5	23.3	29.5

Tooth Growth data + summary

Excel makes it easy to add summary dat such as:

- Mean ($\bar{x}$): =`AVERAGE()`
- Standard Deviation (s): =`STDEV()`
- Observations per group (N): =`COUNT()`
- Standard error of the mean (SEM): = $\frac{s}{\sqrt{N}}$

Orange Juice											Mean	SD	N	SEM
0.5mg per day	15.2	21.5	17.6	9.7	14.5	10	8.2	9.4	16.5	9.7	13.23	4.460	10	1.410
1.0mg per day	19.7	23.3	23.6	26.4	20	25.2	25.8	21.2	14.5	27.3	22.7	3.911	10	1.237
2.0mg per day	25.5	26.4	22.4	24.5	24.8	30.9	26.4	27.3	29.4	23	26.06	2.655	10	0.840
Vitamin C											Mean	SD	N	SEM
0.5mg per day	4.2	11.5	7.3	5.8	6.4	10	11.2	11.2	5.2	7	7.98	2.747	10	0.869
1.0mg per day	16.5	16.5	15.2	17.3	22.5	17.3	13.6	14.5	18.8	15.5	16.77	2.515	10	0.795
2.0mg per day	23.6	18.5	33.9	25.5	26.4	32.5	26.7	21.5	23.3	29.5	26.14	4.798	10	1.517

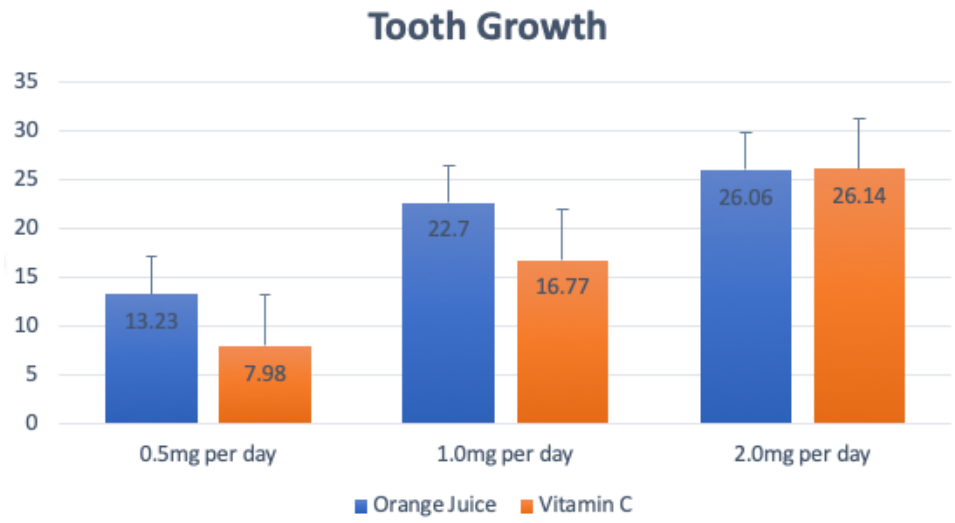
Summary data

Mean	SD	N	SEM
13.23	4.460	10	1.410
22.7	3.911	10	1.237
26.06	2.655	10	0.840
Mean	SD	N	SEM
7.98	2.747	10	0.869
16.77	2.515	10	0.795
26.14	4.798	10	1.517

Often want to plot this summary data

- bar plot per group
- add error bars

And Excel lets you!

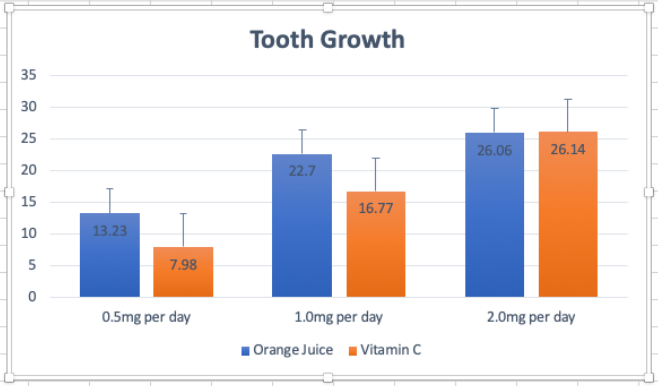


IT'S SO PRETTY!!

And what's more…

YOU CAN MAKE THE PLOT RIGHT IN THE EXCEL SHEET! 🤦

Orange Juice											Mean	SD	N	SEM
0.5mg per day	15.2	21.5	17.6	9.7	14.5	10	8.2	9.4	16.5	9.7	13.23	4.460	10	1.410
1.0mg per day	19.7	23.3	23.6	26.4	20	25.2	25.8	21.2	14.5	27.3	22.7	3.911	10	1.237
2.0mg per day	25.5	26.4	22.4	24.5	24.8	30.9	26.4	27.3	29.4	23	26.06	2.655	10	0.840
Vitamin C											Mean	SD	N	SEM
0.5mg per day	4.2	11.5	7.3	5.8	6.4	10	11.2	11.2	5.2	7	7.98	2.747	10	0.869
1.0mg per day	16.5	16.5	15.2	17.3	22.5	17.3	13.6	14.5	18.8	15.5	16.77	2.515	10	0.795
2.0mg per day	23.6	18.5	33.9	25.5	26.4	32.5	26.7	21.5	23.3	29.5	26.14	4.798	10	1.517



Let’s pause…

Principles of project/data organisation

How many did we just break?

Principles of project/data organisation

How many did we just break?

- create analysis-friendly data
- non-proprietary format
- preserve raw data - never edit
- record all the steps used to process data
- separate input data from derived data

How can we fix this?

- Raw data:
  - analysis-friendly data: re-format data
  - non-proprietary format: save as `.csv` file (comma-separated value: still opens in Excel, but is text-based, so easy to open elsewhere)
  - preserve raw data: separate the data from the analysis
- Analysis
  - record all the steps: use a script-based language (e.g., R or Python)
  - separate input data from derived data: use script to read raw data and produce outputs

Data format

Each column is a variable, each row is an observation.

len	supp	dose
4.2	VC	0.5
11.5	VC	0.5
7.3	VC	0.5
5.8	VC	0.5
6.4	VC	0.5
10	VC	0.5
11.2	VC	0.5
11.2	VC	0.5
5.2	VC	0.5
7	VC	0.5
16.5	VC	1
16.5	VC	1
15.2	VC	1
17.3	VC	1
22.5	VC	1
17.3	VC	1
13.6	VC	1
14.5	VC	1

Save as a `.csv` file, and store in separate `Data` folder.

Analysis

- R (or Python) can be used to process, visualise and analyse the data.
- Let's have a look at a typical R workflow for this.

```
# Read in the data
tooth_growth = read.csv('Data/tooth-growth-long-format.csv')

# Look at the first few rows
head(tooth_growth)
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

Summarize data

The `dplyr` package for R provides a nice way to generate data summaries.

```
# Load dplyr package
library(dplyr)
# Generate data summary object (tg_summary)
tg_summary = tooth_growth %>%
  group_by(supp, dose) %>%
  summarize( MeanLength = mean(len),
             SD = sd(len),
             N = n(),
             SEM = SD / sqrt(N) )
```

Summarize data

What does this produce?

```
# Show the tg_summary object
tg_summary
```

```
## # A tibble: 6 x 6
## # Groups:   supp [2]
##   supp   dose MeanLength    SD     N   SEM
##   <chr> <dbl>      <dbl> <dbl> <int> <dbl>
## 1 OJ      0.5      13.2   4.46    10 1.41
## 2 OJ      1        22.7   3.91    10 1.24
## 3 OJ      2        26.1   2.66    10 0.840
## 4 VC      0.5       7.98  2.75    10 0.869
## 5 VC      1        16.8   2.52    10 0.795
## 6 VC      2        26.1   4.80    10 1.52
```

R can even make it look pretty(ish)

```
tg_summary %>% knitr::kable(., digits=2)
```

supp	dose	MeanLength	SD	N	SEM
OJ	0.5	13.23	4.46	10	1.41
OJ	1.0	22.70	3.91	10	1.24
OJ	2.0	26.06	2.66	10	0.84
VC	0.5	7.98	2.75	10	0.87
VC	1.0	16.77	2.52	10	0.80
VC	2.0	26.14	4.80	10	1.52

How to plot?

- The `ggplot2` packge for R has become (by far) the most popular tool for producing publication quality figures in R.
- It takes a little bit of effort (and Googling) to learn the syntax, but it is an *extremely* powerful and flexible tool.
- Murray Cadzow runs courses that teach R and `ggplot` basics - these are well wroth attending.
- Let's have a look at generating a bar plot of our summary data in R.

Barplot: the code

R code for barplot with `ggplot2`

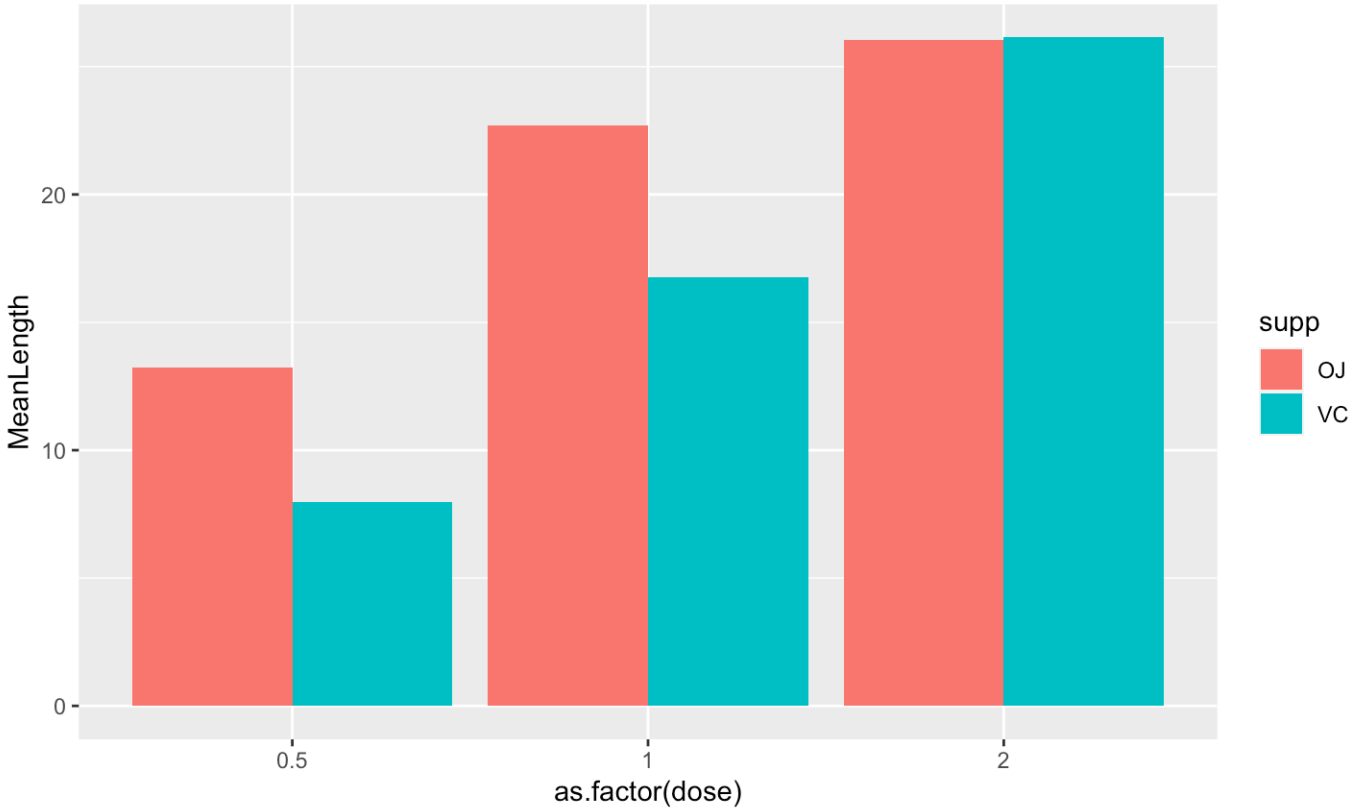
```
# Load the ggplot2 package
library(ggplot2)
# Create a basic bar plot
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  geom_bar(stat="identity", position="dodge")
```

Breaking down the code:

- `ggplot` takes the data object (`tg_summary`), converts `dose` to a factor, plots `dose` on the x axis and `MeanLength` on the y axis, and uses `supp` to define colours.
- `geom_bar` generates the barplot, using the values from `tg_sumamry` (`identity`) and puts the bars next to each other (`dodge`).

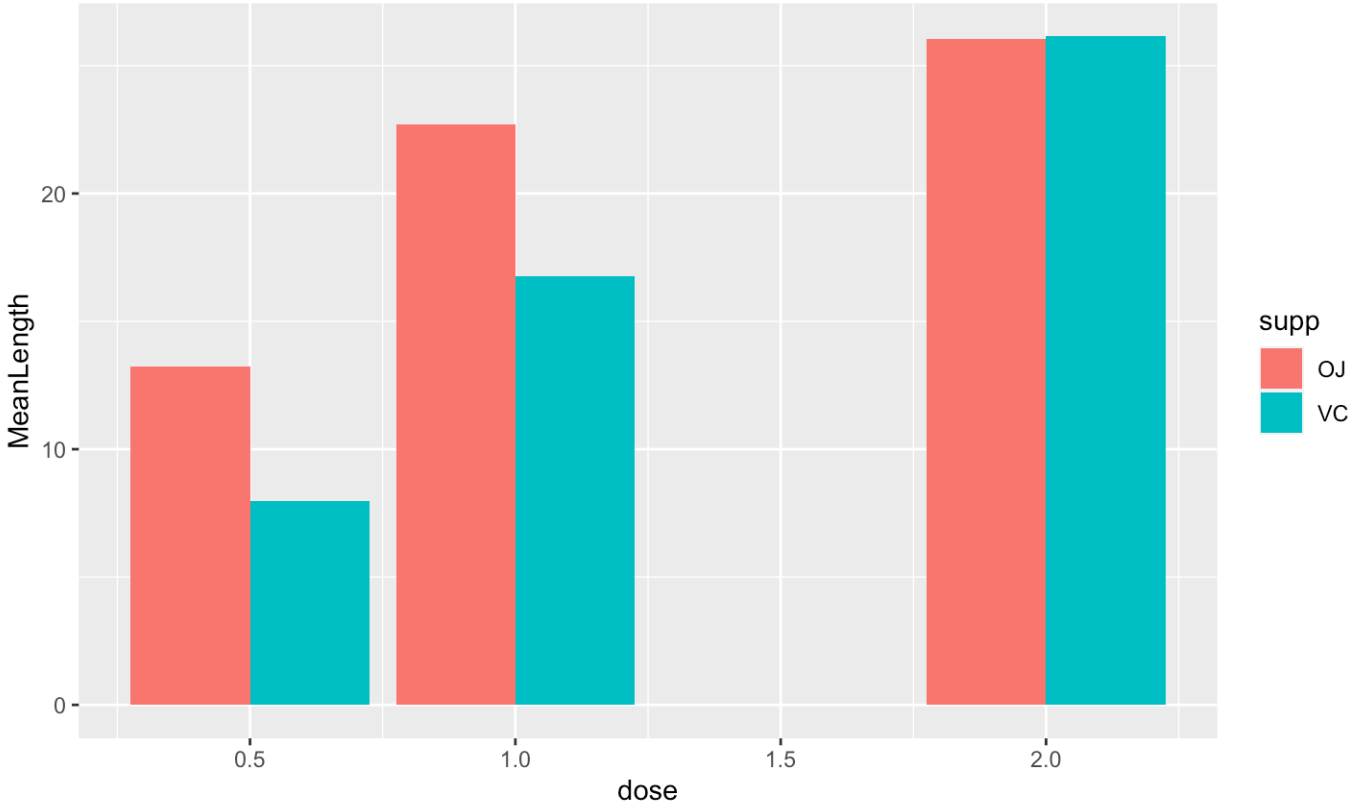
Barplot: the output

```
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  geom_bar(stat="identity", position="dodge")
```



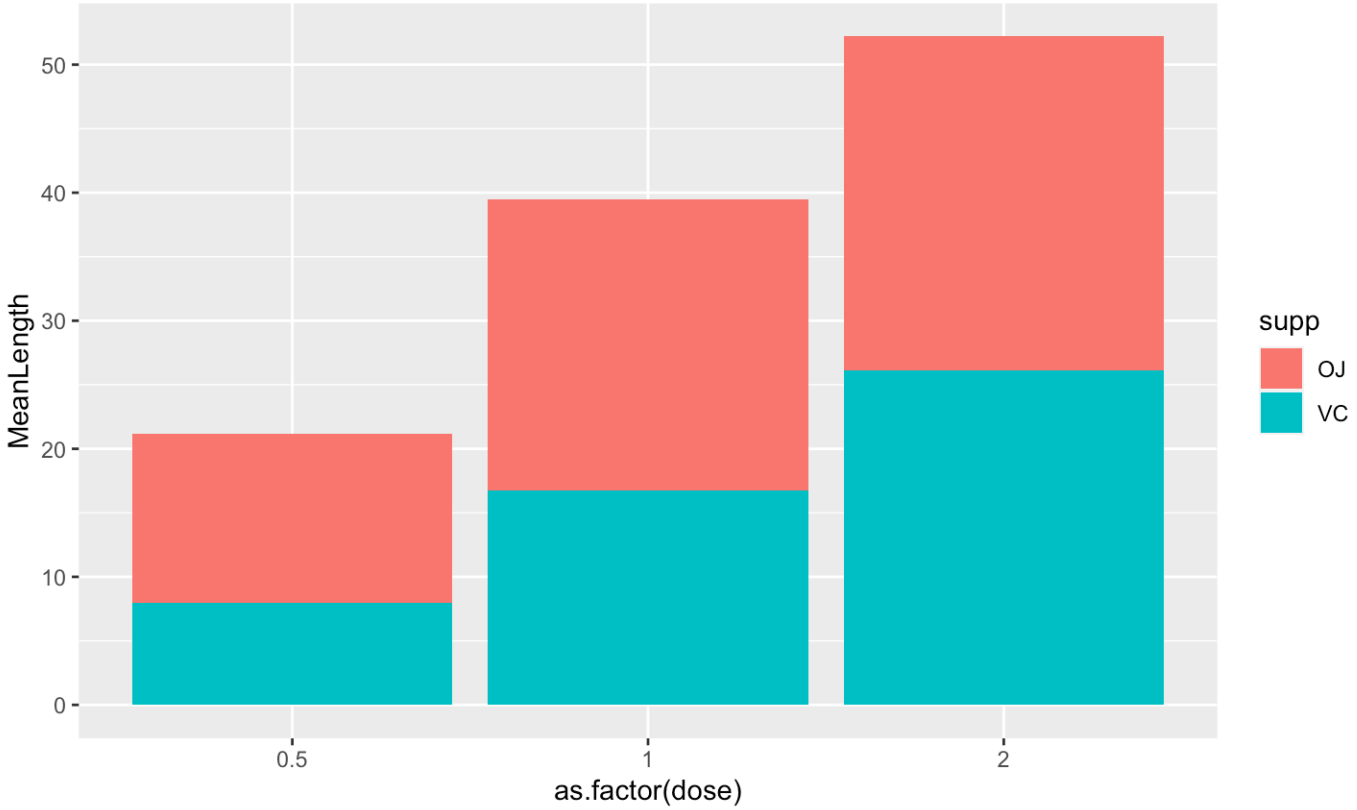
Barplot: without as.factor

```
ggplot(tg_summary, aes(x=dose, y=MeanLength, fill=supp)) +
  geom_bar(stat="identity", position="dodge")
```



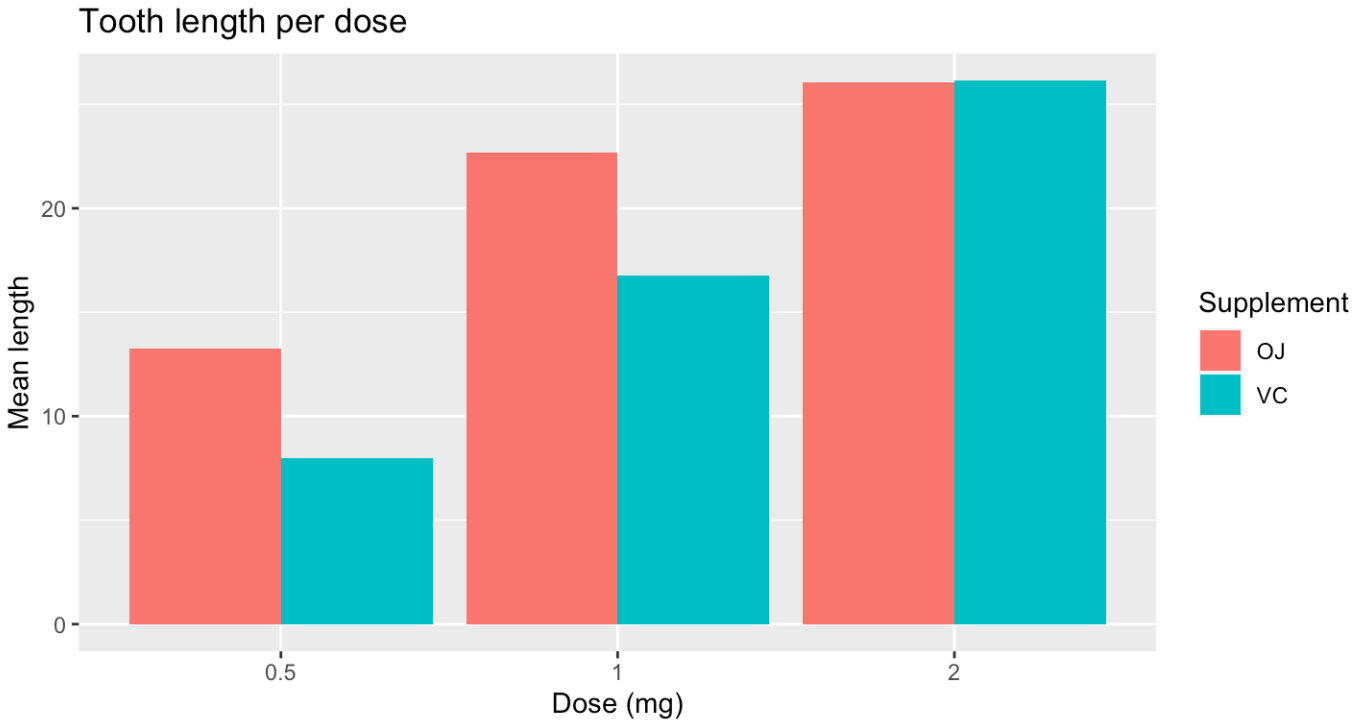
Barplot: without **dodge**

```
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  geom_bar(stat="identity")
```



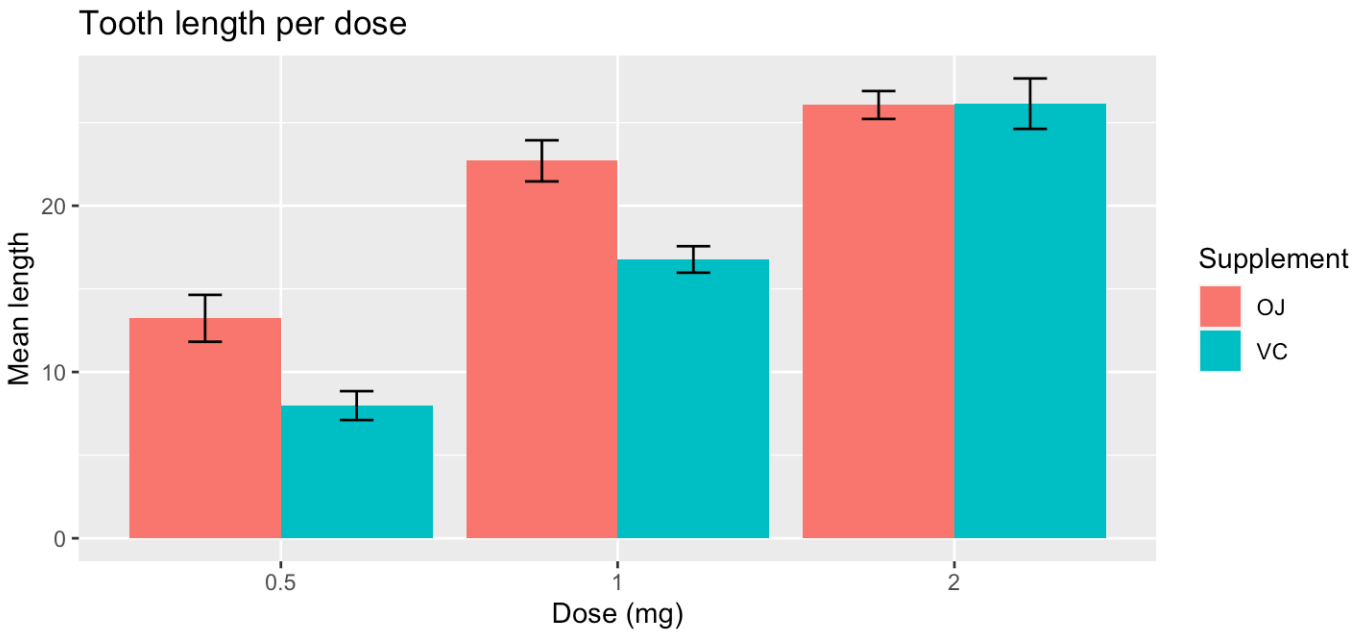
Making in prettier

```
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  geom_bar(stat="identity", position="dodge") +
  labs(title="Tooth length per dose", x="Dose (mg)",
       y = "Mean length", fill='Supplement')
```



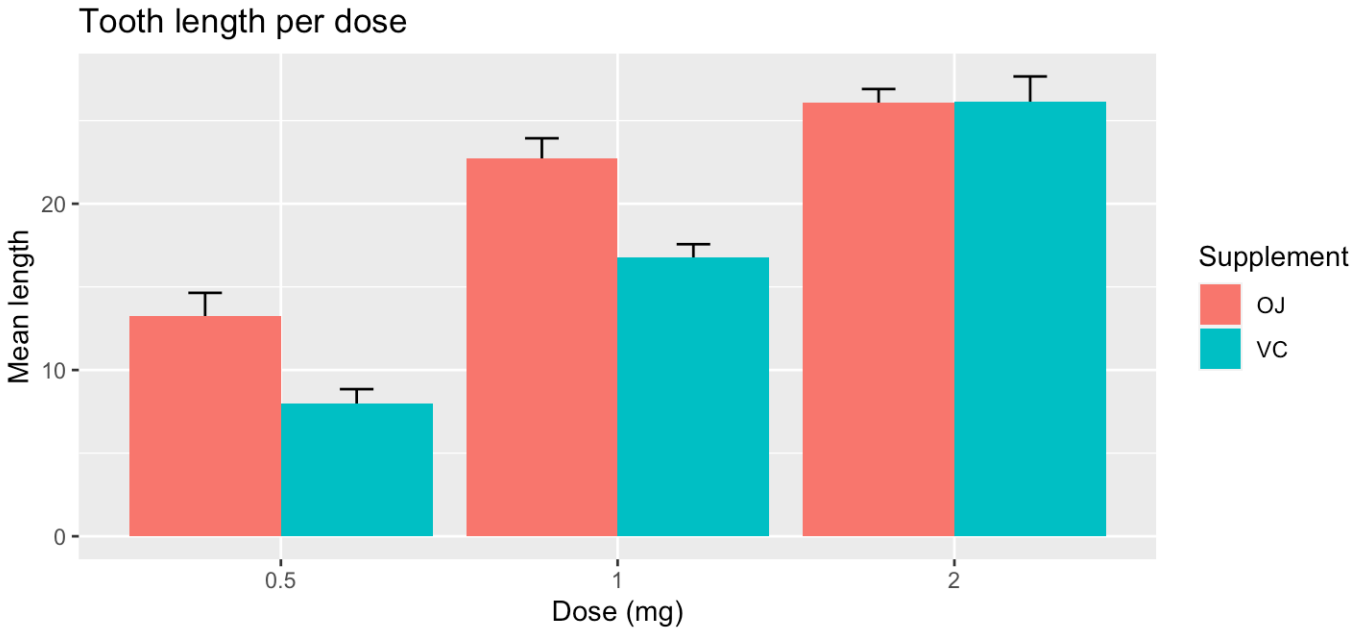
And… error bars

```
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  geom_bar(stat="identity", position="dodge") +
  labs(title="Tooth length per dose", x="Dose (mg)",
       y = "Mean length", fill='Supplement') +
  geom_errorbar(aes(ymin=MeanLength-SEM, ymax=MeanLength+SEM),
                position=position_dodge(0.9), width=0.2)
```



Error bars: upper bar only (change plotting order)

```
ggplot(tg_summary, aes(x=as.factor(dose), y=MeanLength, fill=supp)) +
  labs(title="Tooth length per dose", x="Dose (mg)",
       y = "Mean length", fill='Supplement') +
  geom_errorbar(aes(ymin=MeanLength-SEM, ymax=MeanLength+SEM),
                position=position_dodge(0.9), width=0.2) +
  geom_bar(stat="identity", position="dodge")
```



Aside: why are we plotting the SEM?

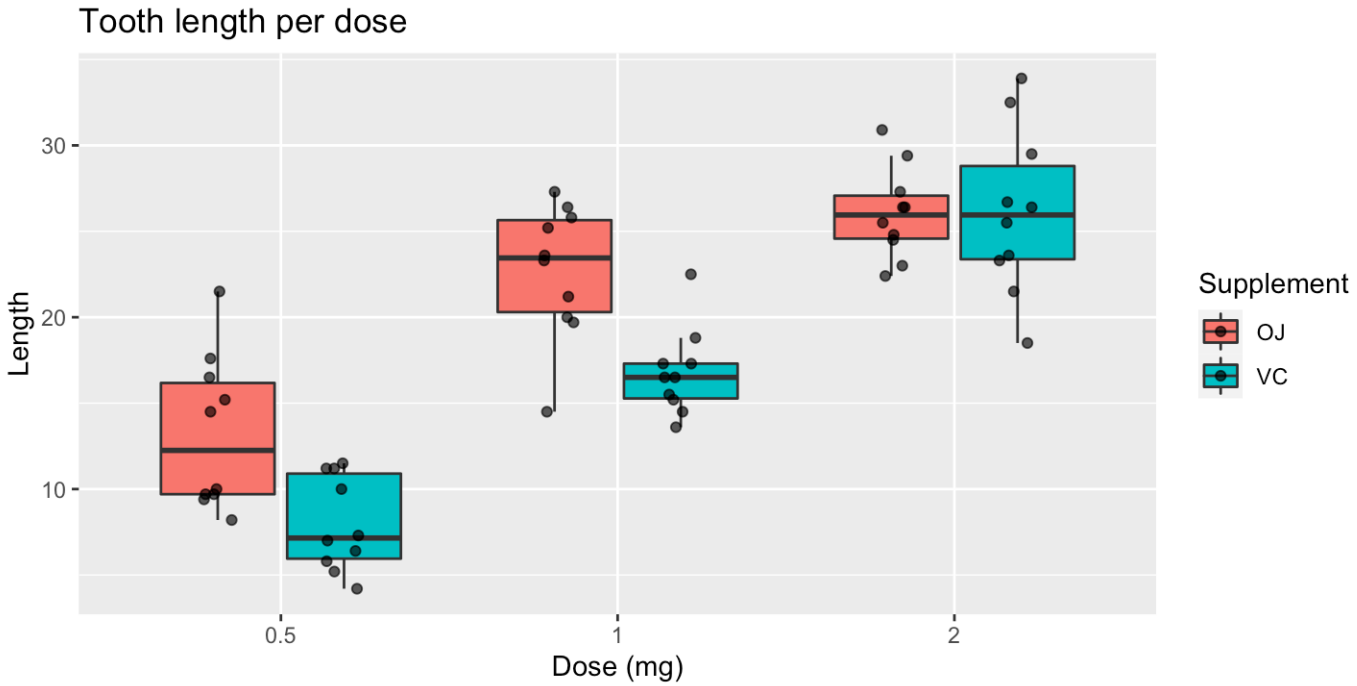
- The sample standard deviation, s, measures the spread of the sample observations around the sample mean
- When we test for differences between the groups, we are testing to see if the *means* are the same.
- To do this we use the standard error of the mean (SEM, or SE):

$$SEM = \frac{s}{\sqrt{N}}$$

- It measures our certainty in our estimate of the population mean, (we are assuming that the sample mean, $\bar{x}$ provides an estimate of the true population mean, μ).
- As we increase the sample size, N, the SEM gets smaller, because with more data, we have a better estimate of μ.

Aside: a more informative plot?

```
ggplot(tooth_growth, aes(x=as.factor(dose), y=len, fill=supp)) +
  labs(title="Tooth length per dose", x="Dose (mg)",
       y = "Length", fill='Supplement') +
  geom_boxplot(outlier.alpha = 0) +
  geom_jitter(position = position_jitterdodge(jitter.width=0.12), alpha=0.7)
```



Part 3 - Statistical Testing

Two sample test

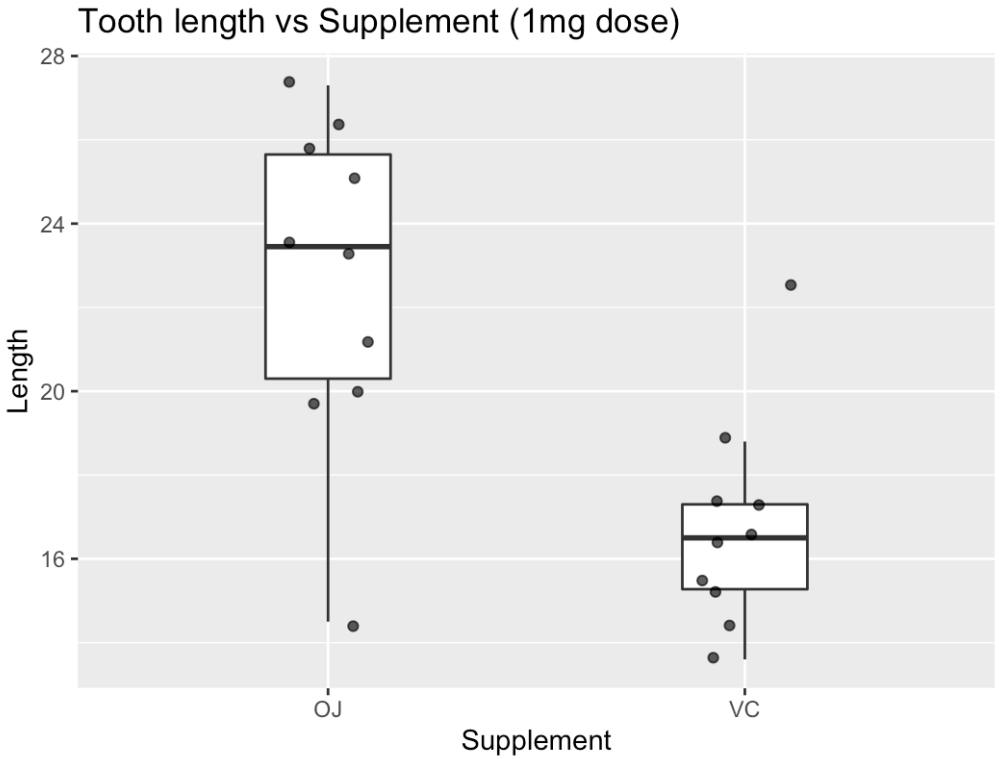
Lets start with a smaller data set: just the OJ vs VC data for a dose of 1mg.

```
# Read in reduced data set
tg_1mg = read.csv('Data/tooth-growth-long-format-dose-1mg.csv')
# Show first few rows
head(tg_1mg)
```

```
##    len supp
## 1 16.5   VC
## 2 16.5   VC
## 3 15.2   VC
## 4 17.3   VC
## 5 22.5   VC
## 6 17.3   VC
```

Plot the data

```
ggplot(tg_1mg, aes(x=supp, y=len)) +
  labs(title="Tooth length vs Supplement (1mg dose)", x="Supplement", y = "Length") +
  geom_boxplot(outlier.alpha = 0, width=0.3) + geom_jitter(width=0.12, alpha=0.7)
```



Hypothesis testing

- We'd like to formally test for a difference between the means of OJ and VC.
- Standard hypothesis testing setup:
  - H_0: the population means are same
  - H_A: the population means are different
- Assuming that the data are normally distributed, we can use a t-test for this:
  - $T = \frac{\bar{x}_1 - \bar{x}_2}{SE(\bar{x}_1 - \bar{x}_2)} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$ (SE for unequal variance across groups)
  - If H_0 is true (i.e., no difference between the means) then T will follow a t-distribution, and we can use this to calculate a p-value.

T-test

In R we can perform a t-test between OJ and VC mean lengths via:

```
t.test(len ~ supp, data = tg_1mg)
```

```
##
##  Welch Two Sample t-test
##
## data:  len by supp
## t = 4.0328, df = 15.358, p-value = 0.001038
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.802148 9.057852
## sample estimates:
## mean in group OJ mean in group VC
##            22.70            16.77
```

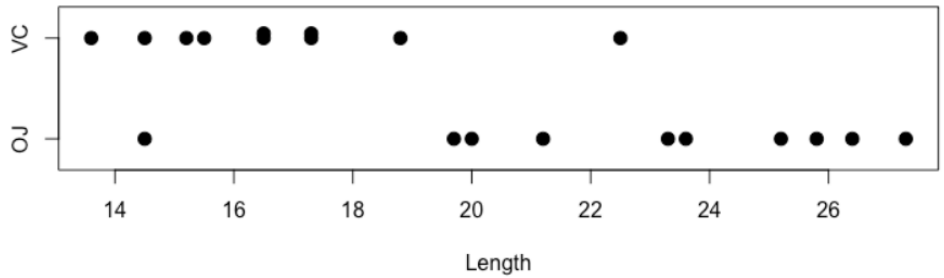
The p-value is 0.00104, which suggests a difference between the means of this size is unlikely to occur by chance if H_0 is true.

What if data aren't normal?

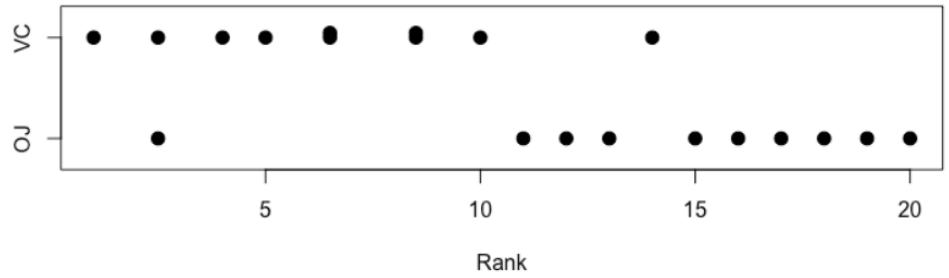
- If the data aren't normally distributed (e.g., skewed, bimodal etc), we can use *non-parametric* methods.
- Note, it may be hard to determine normality when your sample sizes are small.
- The non-parametric analog to the t-test is the Wilcoxon Rank-Sum Test.
  - tests for a difference between the two groups by ranking the combined data, and then testing to see if the ranks are distributed differently (enough) across the groups.
  - in R this is implemented by the `wilcox.test()` function.

Rank-based data

Stripchart of length distribution for each group:



Converted to ranks, the data look like this:



Wilcoxon test ignores the length distribution, and tests for rank-based differences.

Wilcoxon test

```
wilcox.test(len ~ supp, data = tg_1mg)
```

```
## Warning in wilcox.test.default(x = c(19.7, 23.3, 23.6, 26.4, 20, 25.2, 25.8, :
## cannot compute exact p-value with ties
```

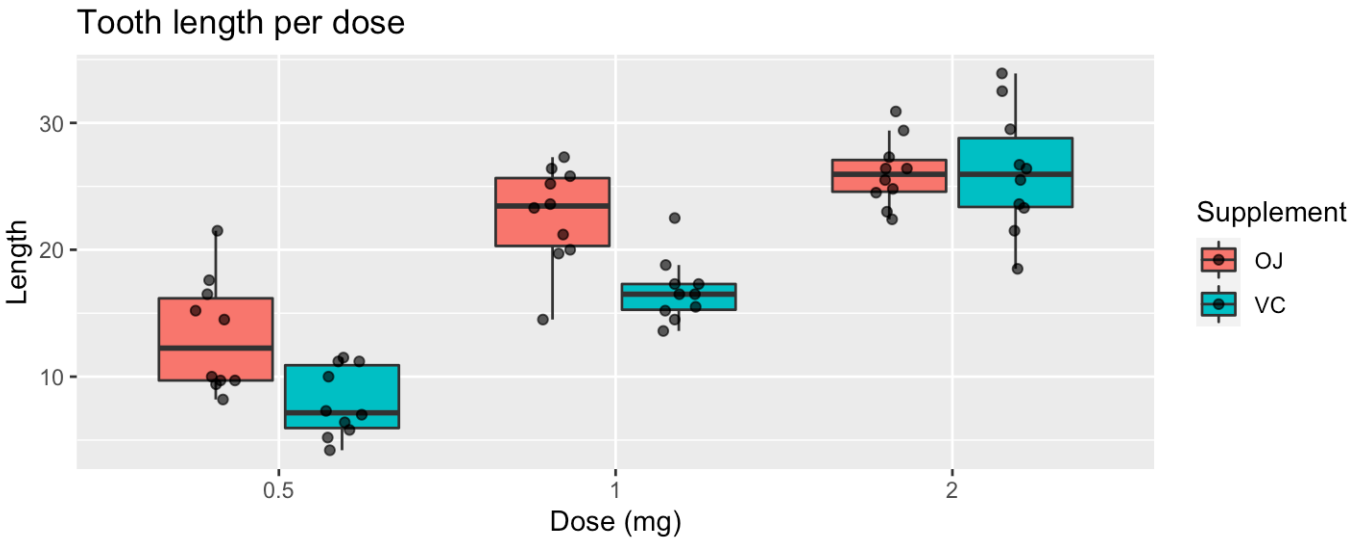
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  len by supp
## W = 88.5, p-value = 0.00403
## alternative hypothesis: true location shift is not equal to 0
```

P-value (0.004) again suggests there is a significant length difference between the groups.

NB: the warning just means that an approximation is being used to calculate the p-value, as there are ties pesent in the data (same value).

Testing more hypotheses

Let’s return to the full tooth length data set.



- If we wanted to test for VC vs OJ at each dose level, we’d need to perform three hypothesis tests.
- Note that there are better statistical ways to perform this analysis, but we’ll keep it simple.

Three hypothesis tests

```
# Create empty vector to fill with p-values
pval = c()

# Dose = 0.5
pval[1] = t.test(len ~ supp, data = tooth_growth %>% filter(dose==0.5) )$p.value

# Dose = 1
pval[2] = t.test(len ~ supp, data = tooth_growth %>% filter(dose==1) )$p.value

# Dose = 2
pval[3] = t.test(len ~ supp, data = tooth_growth %>% filter(dose==2) )$p.value

# Show p-values
names(pval) = c("Dose_0.5mg", "Dose_1.0mg", "Dose_2.0mg")
pval
```

```
##  Dose_0.5mg  Dose_1.0mg  Dose_2.0mg
## 0.006358607 0.001038376 0.963851589
```

Multiple hypothesis testing

- Based on the three p-values, it looks like there is a difference in length at doses 0.5mg and 1.0mg, but not 2.0mg (amusing that $p < 0.05$ indicates significant).
  - By setting a significance at $\alpha = 0.05$, we are specifying the Type I error rate for our tests - also known as the false positive rate.
  - For a test where H_0 is true, we will incorrectly detect a difference between the means $100\alpha\%$ of the time (i.e., 5%).
  - The error probability applies to each test - so we have a 5% chance of error for each test that we perform.
- If we are performing a large number of tests (e.g., genome-wide testing of 1 million SNP loci), then we can expect to make a large number of errors (on average, 0.05 x 1,000,000 = 50,000) in a situation where H_0 is true for most tests.

Multiple testing corrections

- A prevent large numbers of false positives when testing many hypotheses, we employ **multiple testing correction**. Two ways to think about it:
  - make the p-values BIGGER, and use the same threshold
  - make the threshold SMALLER, and use the same p-values
- In Genome Wide Association Studies, the latter approach is generally used: typical genome-wide significance is set at 5×10^{-8}.
  - Bonferroni Correction: $\alpha^* = \frac{\alpha}{\text{number of tests}} = \frac{0.05}{1000000} = 5 \times 10^{-8}$
  - P-value < 5×10^{-8} is required for significance.
  - Alternatively, could multiple each p-value by 1,000,000 and use a threshold of 0.05 (first approach, above).
- This controls the Family-wise Error Rate (FWER), so that P(Type I error) < 0.05 across *all tests being performed*.

Types of error rate control

- - FWER: the **Holm** method is slightly more powerful than Bonferroni.
  - False Discovery Rate (FDR) control provides more liberal p-value adjustment (i.e., more power), but at the cost of a higher Type I error rate (i.e., more false positives).

	RawP	FDR	Holm	Bonferroni
Dose_0.5mg	0.0064	0.0095	0.0127	0.0191
Dose_1.0mg	0.0010	0.0031	0.0031	0.0031
Dose_2.0mg	0.9639	0.9639	0.9639	1.0000

Multiple testing correction in R: `p.adjust`

```
# Bonferroni
p.adjust(pval, method="bonferroni")
```

```
##  Dose_0.5mg  Dose_1.0mg  Dose_2.0mg
## 0.019075820 0.003115128 1.000000000
```

```
# Holm (default for p.adjust)
p.adjust(pval, method="holm")
```

```
##  Dose_0.5mg  Dose_1.0mg  Dose_2.0mg
## 0.012717214 0.003115128 0.963851589
```

```
# FDR
p.adjust(pval, method="fdr")
```

```
##  Dose_0.5mg  Dose_1.0mg  Dose_2.0mg
## 0.009537910 0.003115128 0.963851589
```

Impact

- In this case, it hasn't changed our conclusions (differences between OJ and VC for 0.5mg and 1.0mg, but not for 2.0mg)
- **BUT** when you've tested multiple hypothesis, and your "best" p-values are between 0.01 and 0.05, then performing multiple testing correction will likely make your results non-significant.
- **THIS IS A GOOD THING!**
  - it is telling you that you could have gotten these results by chance
  - either there is no difference, or your need to perform additional experimentation (i.e., increase the sample size) to be more certain.