

Answering questions with data: Lab Manual

Matthew J. C. Crump, Anjali Krishnan, Stephen Volz, and Alla Chavarga

2018: Last Compiled 2025-01-20

Contents

Preface TEST	7
0.1 Important notes	7
Software	11
0.2 Data	11
0.3 R	11
0.4 Excel	18
0.5 SPSS	18
0.6 JAMOVI	18
1 Lab 1: Graphing Data TEST 2 3	19
1.1 General Goals	19
1.2 R	20
1.3 Excel	50
1.4 SPSS	50
1.5 JAMOVI	62
2 Lab 2: Descriptive Statistics	63
2.1 General Goals	63
2.2 R	63
2.3 Excel	72
2.4 SPSS	72
2.5 JAMOVI	86
3 Lab 3: Correlation	87
3.1 General Goals	87
3.2 R	88
3.3 Excel	98
3.4 SPSS	98
3.5 JAMOVI	117
4 Lab 4: Normal Distribution & Central Limit Theorem	119
4.1 General Goals	119
4.2 R	119

4.3	Excel	139
4.4	SPSS	139
4.5	JAMOVI	146
5	Lab 5: Fundamentals of Hypothesis Testing	147
5.1	R	147
5.2	Excel	163
5.3	SPSS	163
5.4	JAMOVI	171
6	Lab 6: t-Test (one-sample, paired sample)	173
6.1	Does Music Convey Social Information to Infants?	173
6.2	Lab skills learned	174
6.3	Important Stuff	174
6.4	R	174
6.5	Excel	194
6.6	SPSS	194
6.7	JAMOVI	212
7	Lab 7: t-test (Independent Sample)	213
7.1	Do you come across as smarter when people read what you say or hear what you say?	213
7.2	Lab skills learned	214
7.3	Important Stuff	214
7.4	R	214
7.5	Excel	222
7.6	SPSS	222
7.7	JAMOVI	230
8	Lab 8 One-way ANOVA	231
8.1	How to not think about bad memories by playing Tetris	231
8.2	Lab Skills Learned	233
8.3	Important Stuff	233
8.4	R	233
8.5	Excel	249
8.6	SPSS	249
8.7	JAMOVI	261
9	Lab 9 Repeated Measures ANOVA	263
9.1	Betcha can't type JHDBZKCO very fast on your first try	263
9.2	Lab Skills Learned	265
9.3	Important Stuff	265
9.4	R	265
9.5	Excel	277
9.6	SPSS	277
9.7	JAMOVI	293

10 Lab 10: Factorial ANOVA	295
10.1 Does standing up make you focus more?	295
10.2 Lab Skills Learned	297
10.3 Important Stuff	297
10.4 R	297
10.5 Excel	310
10.6 SPSS	310
10.7 JAMOVI	327
11 Lab 11: Mixed Factorial ANOVA	329
11.1 Do you remember things better when you take pictures of them?	329
11.2 Lab Skills Learned	330
11.3 Important Stuff	330
11.4 R	330
11.5 Excel	335
11.6 SPSS	335
11.7 JAMOVI	345

Preface TEST

First Draft (version 0.9 = August 15th, 2018) Last Compiled: 2025-01-20

This is the companion lab to our free introductory statistics for undergraduates in psychology textbook, Answering questions with data.

This lab manual involves step by-step tutorials to solve data-analysis problems in software. We use open-data sets that are usually paired with a primary research article.

Each lab has a separate guide for solving the problems in R & R-studio, Excel, SPSS, and JAMOVI.

The manual is a free and open resource. See below for more information about copying, making change, or contributing to the lab manual.

0.1 Important notes

This lab manual is released under a creative commons licence CC BY-SA 4.0. Click the link to read more about the license, or read more below in the license section.

This lab manual is part of a larger OER course package for teaching undergraduate statistics in Psychology. Team members include, Matthew Crump, Alla Chavarga, Anjali Krishnan, Jeffrey Suzuki, and Stephen Volz. As this OER comes together, we will be providing a course website, written in R Markdown, as well as slide decks for the lectures (these will be more fully available by the end of fall 2018). As a result, this textbook, the lab manual, the course website, and the slide decks will all be free, under a creative commons license. The source code for all material is contained in the GitHub repositories for each, and they are written in R-markdown, so they are relatively easy to copy and edit. Have Fun!

0.1.1 Attributions

This lab manual was authored by Matt Crump (R exercises), Anjali Krishnan (JAMOVI exercises), Stephen Volz (EXCEL exercises), and Alla Chavarga

(SPSS exercises).

Labs 6, 7, and 8 were adapted and expanded from Open Stats Labs. Thanks to Open Stats Labs (Dr. Kevin P. McIntyre) for their fantastic work.

0.1.2 CC BY-SA 4.0 license

This license means that you are free to:

- Share: copy and redistribute the material in any medium or format
- Adapt: remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

- Attribution: You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- ShareAlike: If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- No additional restrictions: You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

0.1.3 Copying the lab manual

This lab manual was written in R-Studio, using R Markdown, and compiled into a web-book format using the bookdown package.

All of the source code for compiling the book is available in the GitHub repository for this book:

<https://github.com/CrumpLab/statisticsLab>

In principle, anybody could fork or otherwise download this repository. Load the Rproj file in R-studio, and then compile the entire book. Then, the individual .rmd files for each chapter could be edited for content and style to better suit your needs.

If you want to contribute to this version of the lab manual, you could make pull requests on GitHub, or discuss issues and request on the issues tab.

0.1.4 Acknowledgments

Thanks to the librarians at Brooklyn College of CUNY, especially Miriam Deutch, and Emily Fairey, for their support throughout the process. Thanks

to CUNY for supporting OER development, and for the grant we received to develop this work.

Software

0.2 Data

Data files used for the labs are all taken from open data sources. Links are provided for each lab. For convenience, all of the data files are also available here as single files in the github repository for this lab manual

0.2.1 Data Repository

<https://github.com/CrumpLab/statisticsLab/tree/master/data>

0.2.2 CSV format

All of the data files in .csv format are also available to download as a .zip file

0.2.3 SPSS format

All of the data files in SPSS format are also available to download as a .zip file

0.3 R



In this course we will be using R as a tool to analyze data, and as a tool to help us gain a better understanding of what our analyses are doing. Throughout each lab we will show you how to use R to solve specific problems, and then you will use the examples to solve homework and lab assignments. R is a very deep programming language, and in many ways we will only be skimming the surface of what R can do. Along the way, there will be many pointers to more advanced techniques that interested students can follow to become experts in using R for data-analysis, and computer programming in general.

R is primarily a computer programming language for statistical analysis. It is *free*, and *open-source* (many people contribute to developing it), and runs on

most operating systems. It is a powerful language that can be used for all sorts of mathematical operations, data-processing, analysis, and graphical display of data. I even used R to write this lab manual. And, I use R all the time for my own research, because it makes data-analysis fast, efficient, transparent, reproducible, and exciting.

Statistics Software

- SPSS
- SAS
- JMP
- R
- Julia
- Matlab

0.3.1 Why R?

There are lots of different options for using computers to analyze data, why use R?. The options all have pros and cons, and can be used in different ways to solve a range of different problems. Some software allows you to load in data, and then analyze the data by clicking different options in a menu. This can sometimes be fast and convenient. For example, once the data is loaded, all you have to do is click a couple buttons to analyse the data! However, many aspects of data-analysis are not so easy. For example, particular analyses often require that the data be formatted in a particular way so that the program can analyze it properly. Often times when a researcher wants to ask a new question of an existing data set, they have to spend time re-formatting the data. If the data is large, then reformatting by hand is very slow, and can lead to errors. Another option, is to use a scripting language to instruct the computer how reformat the data. This is very fast and efficient. R provides the ability to everything all in one place. You can load in data, reformat it any way you like, then anlayze it anyway you like, and create beautiful graphs and tables (publication quality) to display your findings. Once you get the hang of R, it becomes very fast and efficient.

0.3.2 Installing R and R Studio

Download and install R onto your computer. The R website is: <http://www.r-project.org>

Find the download R using the link. This will take you to a page with many different mirror links. You can click any of these links to download a version of R that will work on your computer. After you have installed R you can continue.

After you have installed R on your computer, you should want to install another program called R studio. This program provides a user-friendly interface for using R. You must already have installed R before you perform this step. The R-studio website is: <http://www.rstudio.com>

Find the download link on the front-page, and then download R studio desktop version for your computer. After you have installed R studio you will be ready to start using R.

The website R-fiddle allows you to run R scripts in the cloud, so you can practice R from your web-browser!

0.3.3 R studio notes and tips

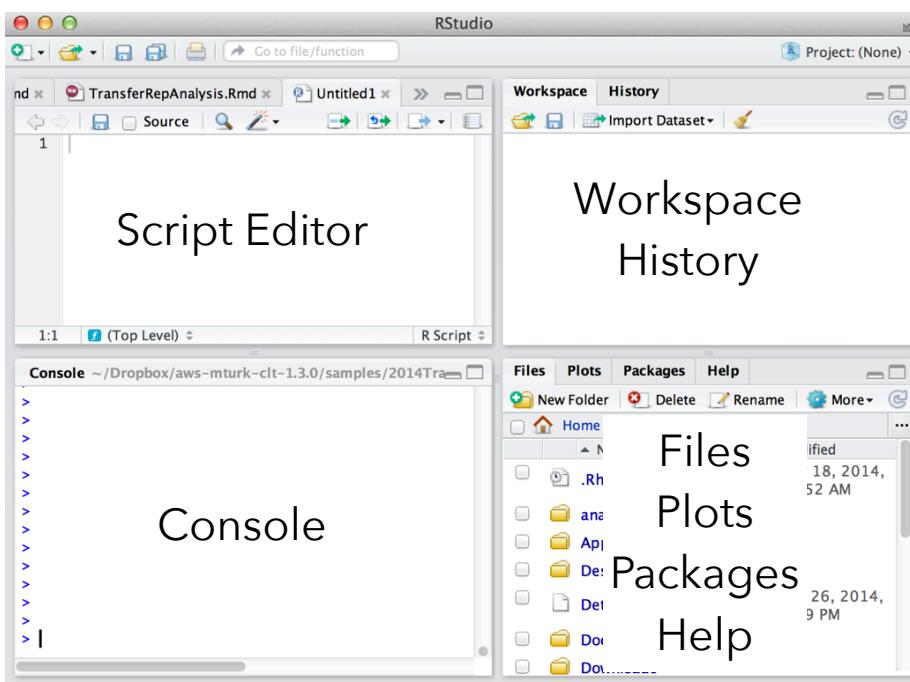


Figure 1: The R-studio workspace

0.3.3.1 Console

When you open up R studio you will see three or four main windows (the placement of each are configurable). In the above example, the bottom left window is the command line (terminal or console) for R. This is used to directly enter commands into R. Once you have entered a command here, press enter to execute the command. The console is useful for entering single lines of code and running them. Oftentimes this occurs when you are learning how to correctly execute a line of code in R. Your first few attempts may be incorrect resulting in errors, but trying out different variations on your code in the command line can help you produce the correct code. Pressing the up arrow while in the console will scroll through the most recently executed lines of code.

0.3.3.2 Script Editor

The top left corner contains the script editor. This is a simple text editor for writing and saving R scripts with many lines. Several tabs can be opened at once, with each tab representing a different R script. R scripts can be saved from the editor (resulting in a .r file). Whole scripts can be run by copy and pasting them into the console and pressing enter. Alternatively, you can highlight portions of the script that you want to run (in the script editor) and press command-enter to automatically run that portion in the console (or press the button for running the current line/section: green arrow pointing right).

0.3.3.3 Workspace and History

The top right panel contains two tabs, one for the workspace and another for history. The workspace lists out all of the variables and functions that are currently loaded in R's memory. You can inspect each of the variables by clicking on them. This is generally only useful for variables that do not contain large amounts of information. The history tab provides a record of the recent commands executed in the console.

0.3.3.4 File, Plot, Packages, Help

The bottom-right window has four tabs for files, plots, packages, and help. The files tab allows browsing of the computers file directory. An important concept in R is the **current working directory**. This is file folder that R points to by default. Many functions in R will save things directly to this direct, or attempt to read files from this directory. The current working directory can be changed by navigating to the desired folder in the file menu, and then clicking on the more option to set that folder to the current working directory. This is especially important when reading in data to R. The current working directory should be set to the folder containing the data to be inputted into R. The plots tab will show recent plots and figures made in R. The packages tab lists the current R libraries loaded into memory, and provides the ability to download and enable new R packages. The help menu is an invaluable tool. Here, you can search for individual R commands to see examples of how they are used. Sometimes the help files for individual commands are opaque and difficult to understand, so it is necessary to do a Google search to find better examples of using these commands.

0.3.4 How to complete the R Labs

Each of the labs focuses on particular data-analysis problems, from graphing data, computing descriptive statistics, to running inferential tests in R. All of the labs come in three parts, a training part, a generalization part, and a writing part. The training part includes step-by-step examples of R code that solves particular problems. The R code is always highlighted in grey. The generalization part gives short assignments to change parts of the provided code

to solve a new problem. The writing part tasks you with answering questions about statistical concepts.

The way to complete each lab is to open a new R Markdown document in R-studio, and then document your progression through each of the parts. By doing this, you will become familiar with how R and R-studio works, and how to create documents that preserve both the code and your notes all in one place. There are a few tricks to getting started that are outlined below.

1. Open R-studio

0.3.4.1 R projects

2. Create a new R project
 - a. Go to the file menu and select new project, or go to the top right-hand corner of R-studio, you should see a blue cube with an R in it, then select New project from the dropdown menu
3. Save the new R project somewhere that you can find it. If you are working on a lab computer, then save the new R project to the desktop.

What is an R project? When you create a new R project you are creating two things, 1) a new folder on your computer, and 2) a “Rproj” file. For example, if you gave your R project the name “Lab1”, then you will have created a folder titled “Lab1”, and inside the folder you will find an R project file called “Lab1.Rproj”.

As you work inside R-studio you will be creating text documents, and you will be doing things like loading data, and saving the results of your analyses. As your work grows and becomes more complex, you can often find yourself creating many different files. **The R project folder is a very useful way of organizing your files all in one place so you can find them later.** If you double-click an R project file, R-studio will automatically load and restore your last session. In the labs, you will be using your R project folder to:

1. save data files into this folder
2. save R-markdown files that you will use to write your R-code and lab notes
3. save the results of your analysis

0.3.4.2 Installing libraries

When you install R and R-studio, you get what is called Base R. Base R contains many libraries that allow you to conduct statistical analyses. Because R is free and open-source, many other developers have created add-on libraries that extend the functionality of R. **We use some of these libraries, and you need to install them before you can do the labs.**

For example, in any of the labs, whenever you see a line of code that uses the word library like this `library(libraryname)`, this line of code telling R to load up

that library so it can be used. The `libraryname` would be replaced with the actual name of the library. For example, you will see code like this in the labs:

```
library(data.table)
```

This line of code is saying that the `data.table` library needs to be loaded. You can check to see if any library is already loaded by clicking on the “packages” tab in the bottom right hand panel. You will see many packages listed in alphabetical order. Packages that are currently loaded and available have a checkmark. If you scroll down and find that you **do not** have `data.table` installed, then you need to install it. To install any package follow these steps:

1. Click on the packages tab
2. Find the “install” button in the top left hand corner of the packages tab.
3. Click the install button
4. Make sure “install from:” is set to CRAN repository
5. Make sure “dependencies” is clicked on (with a checkmark)
6. type the name of the library into the search bar.
7. As you type, you should see the names of different packages you can install pop-up in a drop-down menu. **You must be connected to the internet to install packages from CRAN**
8. Once you find the package (e.g., `data.table`), click it, or just make sure the full, correctly spelled name, is in the search bar
9. Press the install button

You should see some text appear in the console while R installs the package.

10. After you have installed the package, you should now see that it is listed in the packages tab.
11. You can turn the package on by clicking it in the package tab.
12. OR, you can turn the packge on by running the command `library(data.table)` in the console, to do this type `library(data.table)` into the console, and press enter.

0.3.4.3 Quick install

If you are using R on one of the lab computers, you may find that some of the packages are not installed. The lab computers get wiped everynight, so it may be necessary to install packages each time you come back to the lab. Fortunately, we can tell R to install all of the packages we need in one go. Copy the following lines of code into the console, and press enter. Note you can select all of the lines at once, then copy them, then paste all of them into the console, and press enter to run them all. After each of the packages are installed, you will then be able to load them using `library()`.

```
install.packages(ggplot2)
install.packages(dplyr)
install.packages(data.table)
install.packages(summarytools)
```

```
install.packages(gapminder)
install.packages(ggpubr)
```

0.3.4.4 R markdown

Once you have the necessary packages installed you can begin creating R markdown documents for each lab. We admit that at the beginning, R markdown documents might seem a little bit confusing, but you will find they are extremely useful and flexible. Basically, what R markdown allows you to do is combine two kinds of writing, 1) writing R code to conduct analyses, and 2) writing normal text, with headers, sub-headers, and paragraphs. You can think of this like a lab journal, that contains both your writing about what you are doing (e.g., notes to self), and the code that you use for analysis. Additionally, when your code does something like make a graph, or run a statistical test, you can ask R markdown to print the results.

The R markdown website has an excellent tutorial that is well worth your time to check out: <https://rmarkdown.rstudio.com/lesson-1.html>

0.3.4.5 R markdown lab templates

We have created a set of template documents for each lab that can be downloaded here: download lab templates.

When you unzip the file you should find the following:

1. A new folder titled “RMarkdownsLab”
2. Inside the folder you will see the “RMarkdownsLab.Rproj” file
3. A data folder containing data files for the labs
4. A “LabTemplates” folder containing the R markdown templates for each lab.

To get started with Lab 1, follow these steps:

1. copy the template file for lab 1, “Lab 01 Graphing_Student Name.Rmd”, and place it into the “RMarkdownsLab” (copy it out of the template folder, and into the RMarkdownsLab folder).
2. Rename the file to add your own name, eg., “Lab1GraphingMattCrump.Rmd”
3. double-click the “RMarkdownsLab.Rproj” file
4. R-studio will now load up.
5. If you click the files tab, you will see all of the files and folders inside the “RMarkdownsLab” folder
6. Click on your lab1 .rmd file, it will now load into the editor window.

Each lab template .rmd file contains three main sections, one for each part of the lab. You will write things inside each section to complete the lab.

0.3.5 Screencast tutorial

Follow this guide to get up running for Lab 1.

0.3.6 R-studio Cloud

R-studio is also in the cloud. This means that if you want to use R and R-studio through your web-browser you can do that without even installing R or R-studio on your computer. It's also free!

1. sign up for an R-studio cloud account here: <https://rstudio.cloud>
2. You can make new R projects, work inside them, and everything is saved in the cloud!
3. To see how everything would work, follow the steps in this video. You will need to download this .zip file to your computer to get started

The link to the video is <https://www.youtube.com/watch?v=WsbnV0t7FE4>, or you can watch it here:

0.4 Excel**0.5 SPSS****0.6 JAMOVI**

Chapter 1

Lab 1: Graphing Data TEST 2 3

The commonality between science and art is in trying to see profoundly - to develop strategies of seeing and showing. —Edward Tufte

As we have found out from the textbook and lecture, when we measure things, we get lots of numbers. Too many. Sometimes so many your head explodes just thinking about them. One of **the most helpful things** you can do to begin to make sense of these numbers, is to look at them in graphical form. Unfortunately, for sight-impaired individuals, graphical summary of data is much more well-developed than other forms of summarizing data for our human senses. Some researchers are developing auditory versions of visual graphs, a process called **sonification**, but we aren't prepared to demonstrate that here. Instead, we will make charts, and plots, and things to look at, rather than the numbers themselves, mainly because these are tools that are easiest to get our hands on, they are the most developed, and they work really well for visual summary. If time permits, at some point I would like to come back here and do the same things with sonification. I think that would be really, really cool!

1.1 General Goals

Our general goals for this first lab are to get your feet wet, so to speak. We'll do these things:

1. Load in some data to a statistical software program
2. Talk a little bit about how the data is structured
3. Make graphs of the data so we can look at it and make sense of it.

1.1.1 Important info

1. Data for NYC film permits was obtained from the NYC open data website.
The .csv file can be found here: Film_Permits.csv
2. Gapminder data from the gapminder project (copied from the R gapminder library) can be downloaded in .csv format here: gapminder.csv

1.2 R

1.2.1 Download the lab templates

You will be completing each lab by writing your code and notes in an R Markdown document.

1. Download the RMarkdownsLab.zip to your computer.
2. Unzip the file, this will produce a new folder with three important parts
 - a. data folder (contains data files for all labs)
 - b. LabTemplates folder (contains blank templates for completing all the labs)
 - c. RMarkdownsLab.Rproj A file with a little blue cube with an R in it.
3. Double-click the RMarkdownsLab.Rproj file, this will automatically open R-studio (if you are at home, you must install R and R-studio first, or you can use R-studio Cloud through your web-browser)
4. Copy the template .Rmd file for lab 1 from the LabTemplates folder into the main folder, then open it, and use it to begin adding your code and notes for lab 1.
5. Watch this screencast to help you get started.

Your lab instructor will show you how to open R-studio on the lab computer. Just find it and double-click. Now you have R-studio. Your lab instructor will also walk you through the steps to get started completing the first lab. We also wrote down the steps here.

There are numerous resources for learning about R, we put some of them on the course website, under the resources page. You will find these resources helpful as you learn. We also have a kind of general introduction to R and Rstudio here. This shows you how to download R and R-studio at home (it's free). Throughout the labs you will be writing things called R Markdown documents. You will learn how to do this throughout the labs, but it can also be worthwhile reading other tutorials, such as the one provided by R Markdown.

When we made this course, we assumed that most students would be unfamiliar with R and R-studio, and might even be frightened of it, because it is a computer programming language (OOOOHHH NOOOOOOO, I NEED TO DROP THIS COURSE NOW)...Don't worry. It's going to be way easier than you think. Let's compare to other statistics courses where you would learn something like SPSS. That is also a limited programming language, but you would mostly learn how

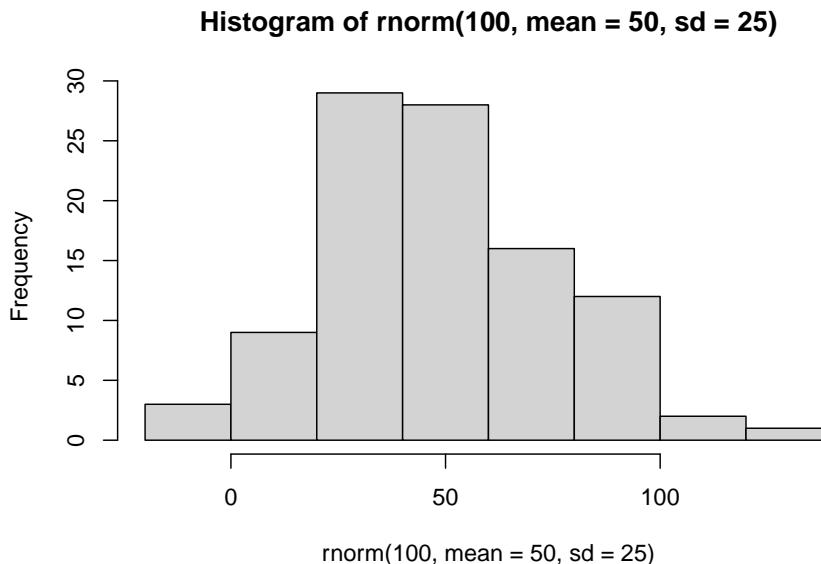
to point with a mouse, and click with button. I bet you already know how to do that. I bet you also already know how to copy and paste text, and press enter. That's mostly what we'll be doing to learn R. We will be doing statistics by typing commands, rather than by clicking buttons. However, lucky for you, all of the commands are already written for you. You just have to copy/paste them.

We know that this will seem challenging at first. But, we think that with lots of working examples, you will get the hang of it, and by the end of the course you will be able to do things you might never have dreamed you can do. It's really a fantastic skill to learn, even if you aren't planning on going on to do research in Psychology (in which case, this kind of thing is necessary skill to learn). With that, let's begin.

1.2.2 Get some data

In order to graph data, we need to have some data first...Actually, with R, that's not quite true. Run this bit of code and see what happens:

```
hist(rnorm(100, mean=50, sd=25))
```



You just made R sample 100 numbers, and then plot the results in a histogram. Pretty neat. We'll be doing some of this later in the course, where get R to make fake data for us, and then we learn to think about how data behaves under different kinds of assumptions.

For now, let's do something that might be a little bit more interesting...what

movies are going to be filming in NYC? It turns out that NYC makes a lot of data about a lot things open and free for anyone to download and look at. This is the NYC Open Data website: <https://opendata.cityofnewyork.us>. I searched through the data, and found a data file that lists the locations of film permits for shooting movies all throughout the Burroughs. There are multiple ways to load this data into R.

1. If you have downloaded the RMarkdownsLab.zip file, then you already have the data file in the data folder. Assuming you are working in your main directory (your .rmd file is saved in the main folder that contains both the data and template folders), then use the following commands to load the data.

```
library(data.table)
nyc_films <- fread("data/Film_Permits.csv")
```

2. If the above method doesn't work, you can try loading the data from the course website using:

```
library(data.table)
nyc_films <- fread("https://raw.githubusercontent.com/CrumpLab/statisticsLab/master/data/Film_Permits.csv")
```

If you are having issues getting the data loaded, then talk to your lab instructor

1.2.3 Look at the data

You will be downloading and analyzing all kinds of data files this semester. We will follow the very same steps every time. The steps are to load the data, then look at it. You want to see what you've got.

In R-studio, you will now see a variable called `nyc_films` in the top right-hand corner of the screen, in the environment tab. If you click this thing, it will show you the contents of the data in a new window. The data is stored in something we call a `data frame`. It's R lingo, for the thing that contains the data. Notice is a square, with rows going across, and columns going up and down. It looks kind of like an excel spreadsheet if you are familiar with Excel.

It's useful to know you can look at the data frame this way if you need to. But, this data frame is really big, it has 50,728 rows of data. That's a lot too much to look at.

1.2.3.1 `summarytools`

The `summarytools` packages give a quick way to summarize all of the data in a data frame. Here's how. When you run this code you will see the summary in the viewer on the bottom right hand side. There's a little browser button (arrow on top of little window) that you can click to expand and see the whole thing in a browser.

```
library(summarytools)
view(dfSummary(nyc_films))
```

That is super helpful, but it's still a lot to look at. Because there is so much data here, it's pretty much mind-boggling to start thinking about what to do with it.

1.2.4 Make Plots to answer questions

Let's walk through a couple questions we might have about this data. We can see that there were 50,728 film permits made. We can also see that there are different columns telling us information about each of the film permits. For example, the `Borough` column lists the Borough for each request, whether it was made for: Manhattan, Brooklyn, Bronx, Queen's, or Staten Island. Now we can ask our first question, and learn how to do some plotting in R.

1.2.4.1 Where are the most film permits being requested?

Do you have any guesses? Is it Manhattan, or Brooklyn, of the Bronx? Or Queen's or Staten Island? We can find out by plotting the data using a bar plot. We just need to count how many film permits are made in each borough, and then make different bars represent the counts.

First, we do the counting in R. Run the following code.

```
library(dplyr)

counts <- nyc_films %>%
  group_by(Borough) %>%
  summarize(count_of_permits = length(Borough))
```

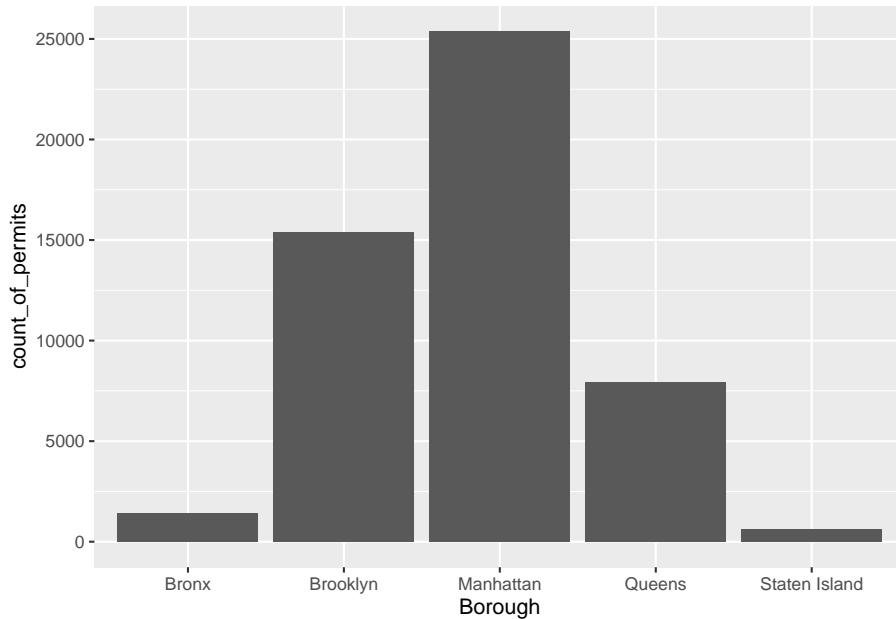
The above grouped the data by each of the five Borough's, and then counted the number of times each Borough occurred (using the `length` function). The result is a new variable called `count`. I chose to name this variable `count`. You can see that it is now displayed in the top-right hand corner in the environment tab. If you gave `count` a different name, like `muppets`, then it would be named what you called it.

If you click on the `counts` variable, you will see the five boroughs listed, along with the counts for how many film permits were requested in each Borough. These are the numbers that we want to plot in a graph.

We do the plot using a fantastic package called `ggplot2`. It is very powerful once you get the hand of it, and when you do, you will be able to make all sorts of interesting graphs. Here's the code to make the plot

```
library(ggplot2)
```

```
ggplot(counts, aes(x = Borough, y = count_of_permits )) +
  geom_bar(stat="identity")
```



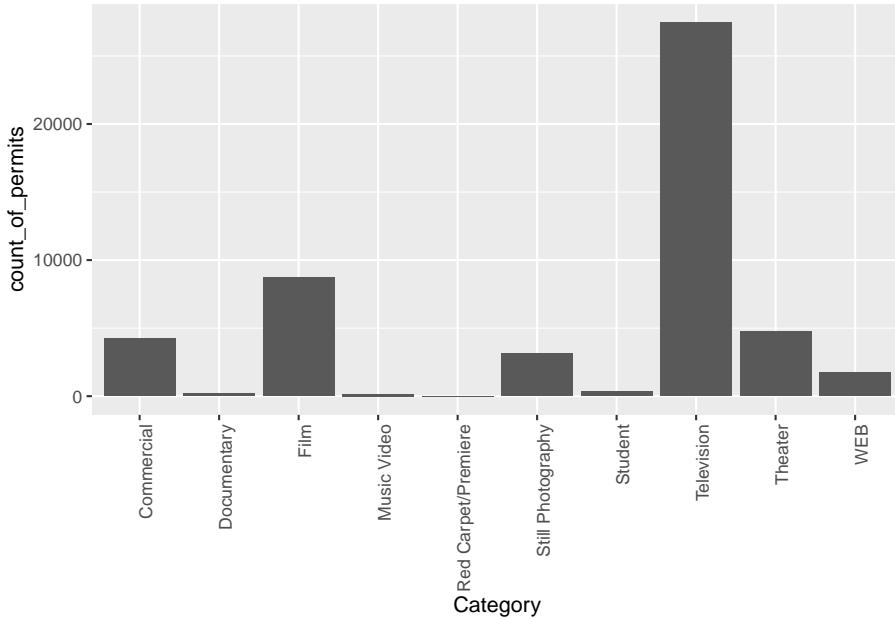
There it is, we're done here! We can easily look at this graph, and answer our question. Most of the film permits were requested in Manhattan, followed by Brooklyn, then Queen's, the Bronx, and finally Staten Island.

1.2.4.2 What kind of “films” are being made, what is the category?

We think you might be skeptical of what you are doing here, copying and pasting things. Soon you'll see just how fast you can do things by copying and pasting, and make a few little changes. Let's quickly ask another question about what kinds of films are being made. The column **Category**, gives us some information about that. Let's just copy paste the code we already made, and see what kinds of categories the films fall into. See if you can tell what I changed in the code to make this work, I'll do it all at once:

```
counts <- nyc_films %>%
  group_by(Category) %>%
  summarize(count_of_permits = length(Category))

ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



OK, so this figure might look a bit weird because the labels on the bottom are running into each other. We'll fix that in a bit. First, let's notice the changes.

1. I changed `Borough` to `Category`. That was the main thing
2. I left out a bunch of things from before. None of the `library()` commands are used again, and I didn't re-run the very early code to get the data. R already has those things in it's memory, so we don't need to do that first. If you ever clear the memory of R, then you will need to reload those things. First-things come first.

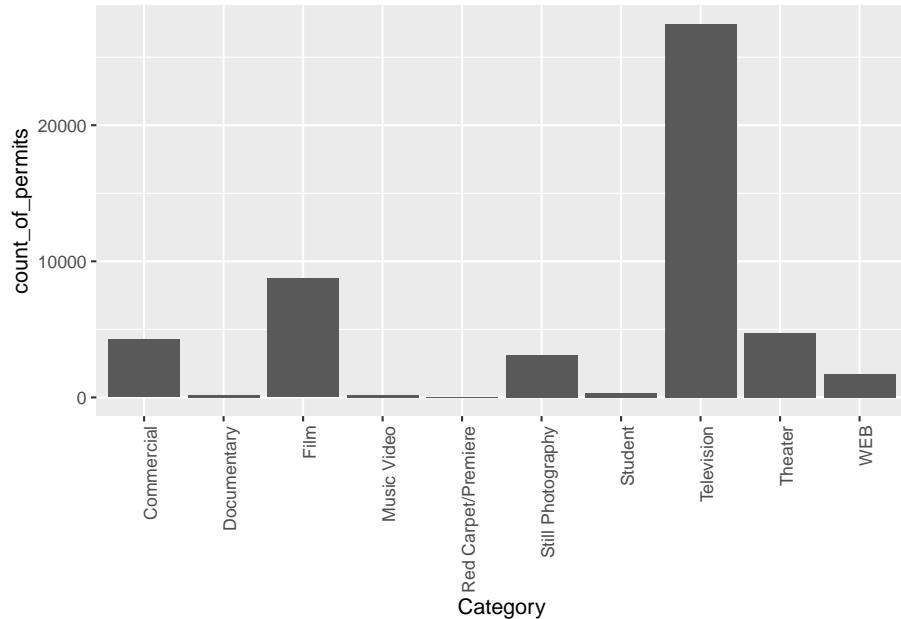
Fine, so how do we fix the graph? Good question. To be honest, I don't know right now. I totally forgot how. But, I know ggplot2 can do this, and I'm going to Google it, right now. Then I'm going to find the answer, and use it here. The googling of your questions is a fine way to learn. It's what everybody does these days....[goes to Google...].

Found it, actually found a lot of ways to do this. The trick is to add the last line. I just copy-pasted it from the solution I found on stack overflow (you will become friend's with stack overflow, there are many solutions there to all of your questions)

```
counts <- nyc_films %>%
  group_by(Category) %>%
  summarize(count_of_permits = length(Category))

ggplot(counts, aes(x = Category, y = count_of_permits)) +
```

```
geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



1.2.5 ggplot2 basics

Before we go further, I want to point out some basic properties of ggplot2, just to give you a sense of how it is working. This will make more sense in a few weeks, so come back here to remind yourself. We'll do just a bit a basics, and then move on to making more graphs, by copying and pasting.

The ggplot function uses layers. Layers you say? What are these layers? Well, it draws things from the bottom up. It lays down one layer of graphics, then you can keep adding on top, drawing more things. So the idea is something like: Layer 1 + Layer 2 + Layer 3, and so on. If you want Layer 3 to be Layer 2, then you just switch them in the code.

Here is a way of thinking about ggplot code

```
ggplot(name_of_data, aes(x = name_of_x_variable, y = name_of_y_variable)) +
  geom_layer()+
  geom_layer()+
  geom_layer()
```

What I want you to focus on in the above description is the + signs. What we are doing with the plus signs is adding layers to plot. The layers get added in the order that they are written. If you look back to our previous code, you

will see we add a `geom_bar` layer, then we added another layer to change the rotation of the words on the x-axis. This is how it works.

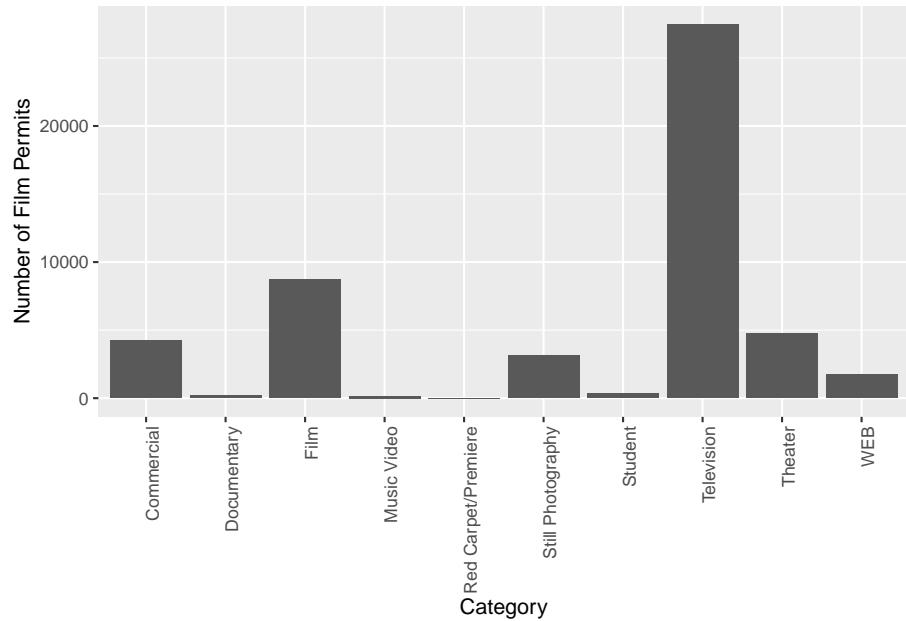
BUT WAIT? How am I supposed to know what to add? This is nuts! We know. You're not supposed to know just yet, how could you? We'll give you lots of examples where you can copy and paste, and they will work. That's how you'll learn. If you really want to read the help manual you can do that too. It's on the ggplot2 website. This will become useful after you already know what you are doing, before that, it will probably just seem very confusing. However, it is pretty neat to look and see all of the different things you can do, it's very powerful.

For now, let's get the hang of adding things to the graph that let us change some stuff we might want to change. For example, how do you add a title? Or change the labels on the axes? Or add different colors, or change the font-size, or change the background? You can change all of these things by adding different lines to the existing code.

1.2.5.1 `ylab()` changes y label

The last graph had `count_of_permits` as the label on the y-axis. That doesn't look right. ggplot2 automatically took the label from the column, and made it be the name on the y-axis. We can change that by adding `ylab("what we want")`. We do this by adding a `+` to the last line, then adding `ylab()`

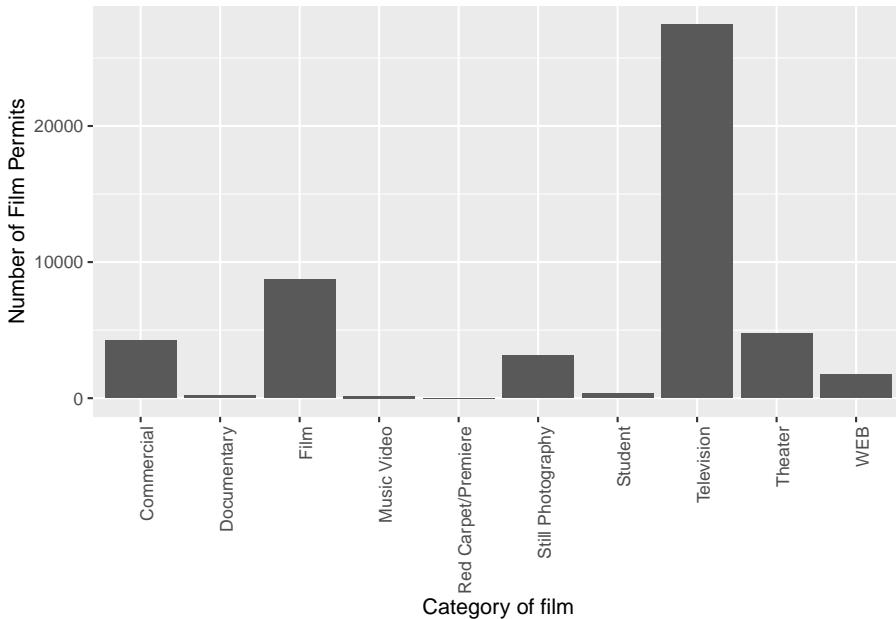
```
ggplot(counts, aes(x = Category, y = count_of_permits )) +  
  geom_bar(stat="identity") +  
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +  
  ylab("Number of Film Permits")
```



1.2.5.2 xlab() changes x label

Let's slightly modify the x label too:

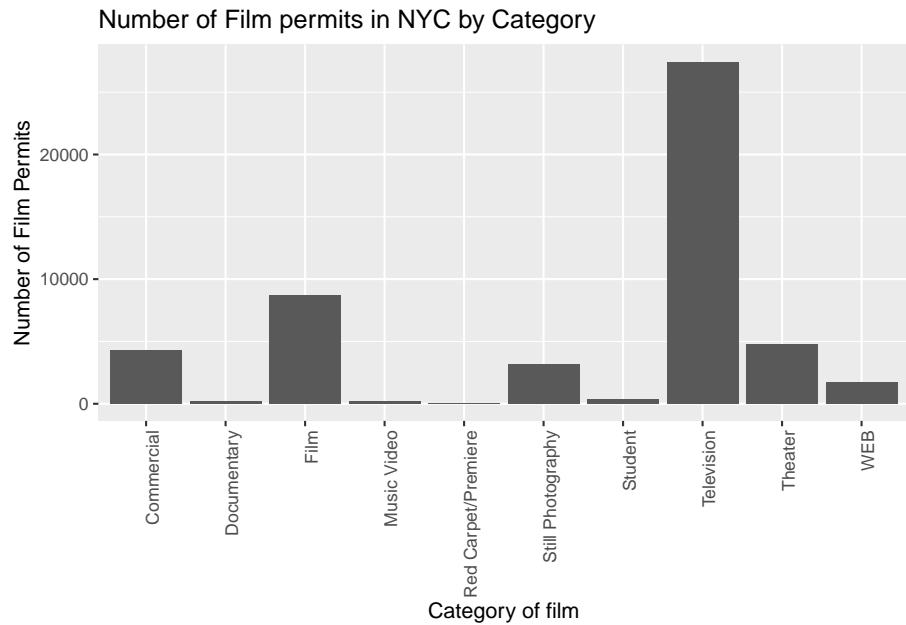
```
ggplot(counts, aes(x = Category, y = count_of_permits)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film")
```



1.2.5.3 ggtitle() adds title

Let's give our graph a title

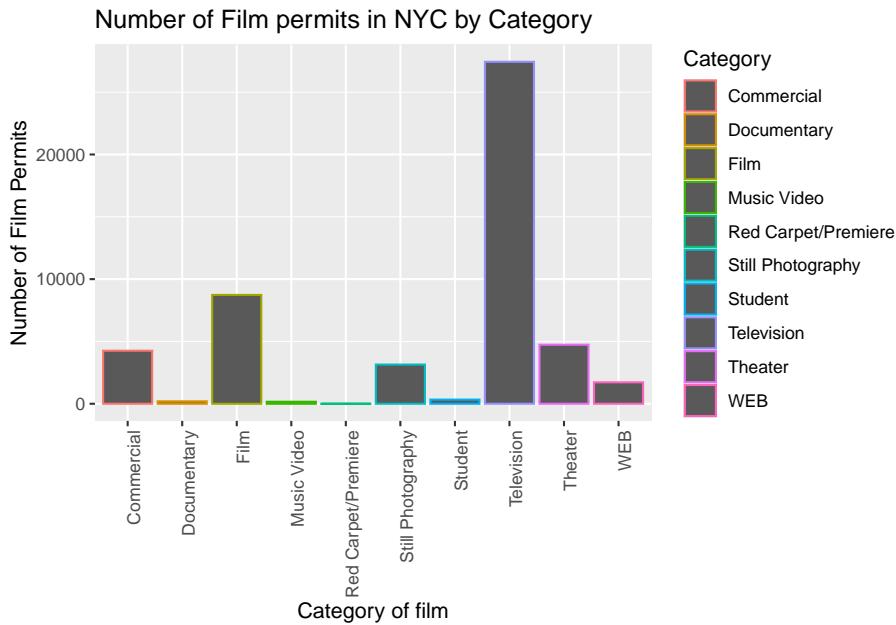
```
ggplot(counts, aes(x = Category, y = count_of_permits)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category")
```



1.2.5.4 color adds color

Let's make the bars different colors. To do this, we add new code to the inside of the `aes()` part:

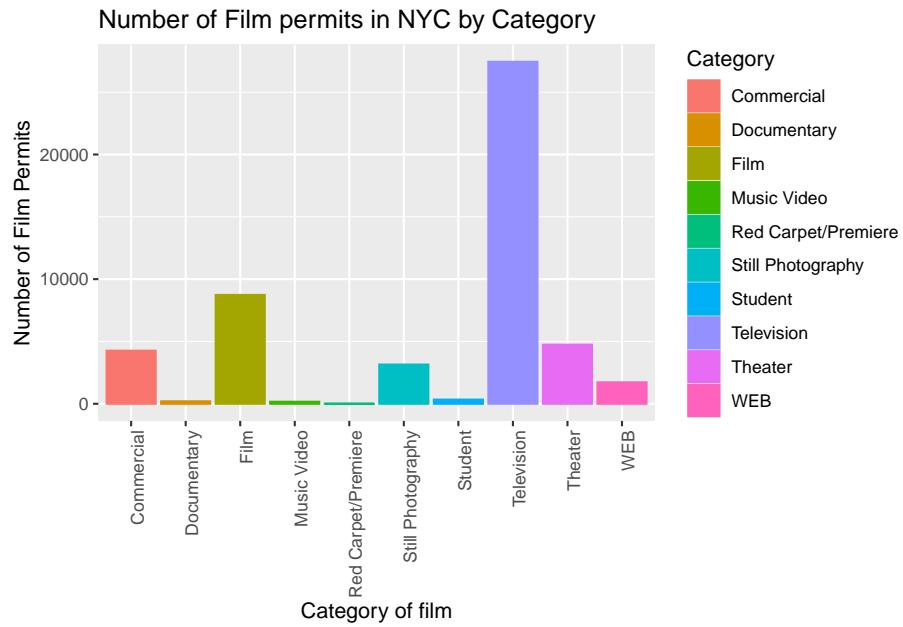
```
ggplot(counts, aes(x = Category, y = count_of_permits, color=Category)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category")
```



1.2.5.5 fill fills in color

Let's make the bars different colors. To do this, we add new code to the inside of the `aes()` part...Notice I've started using new lines to make the code more readable.

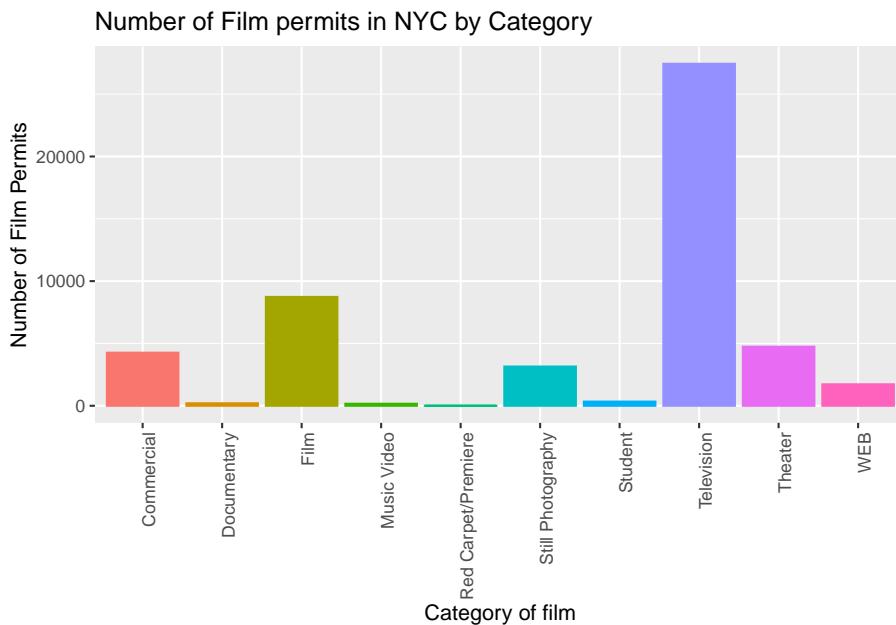
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category")
```



1.2.5.6 get rid of the legend

Sometimes you just don't want the legend on the side, to remove it add

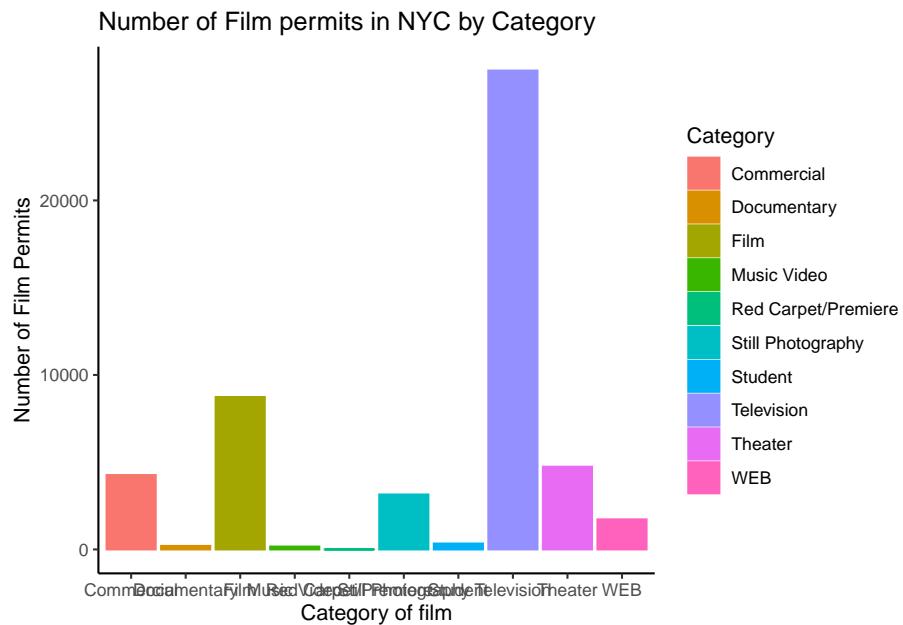
```
theme(legend.position="none")
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



1.2.5.7 theme_classic() makes white background

The rest is often just visual preference. For example, the graph above has this grey grid behind the bars. For a clean classic no nonsense look, use `theme_classic()` to take away the grid.

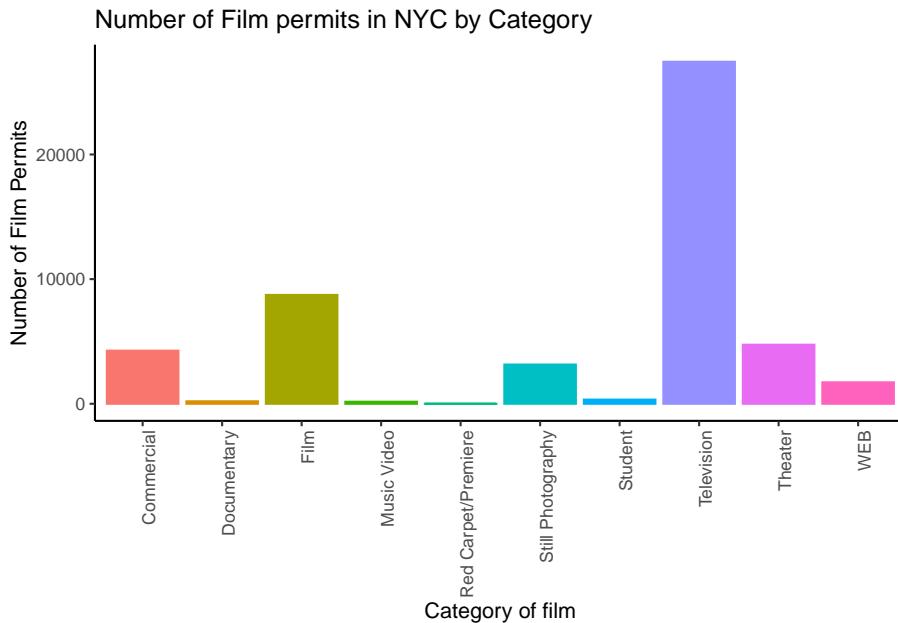
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none") +
  theme_classic()
```



1.2.5.8 Sometimes layer order matters

Interesting, `theme_classic()` is misbehaving a little bit. It looks like we have some of our layer out of order, let's re-order. I just moved `theme_classic()` to just underneath the `geom_bar()` line. Now everything get's drawn properly.

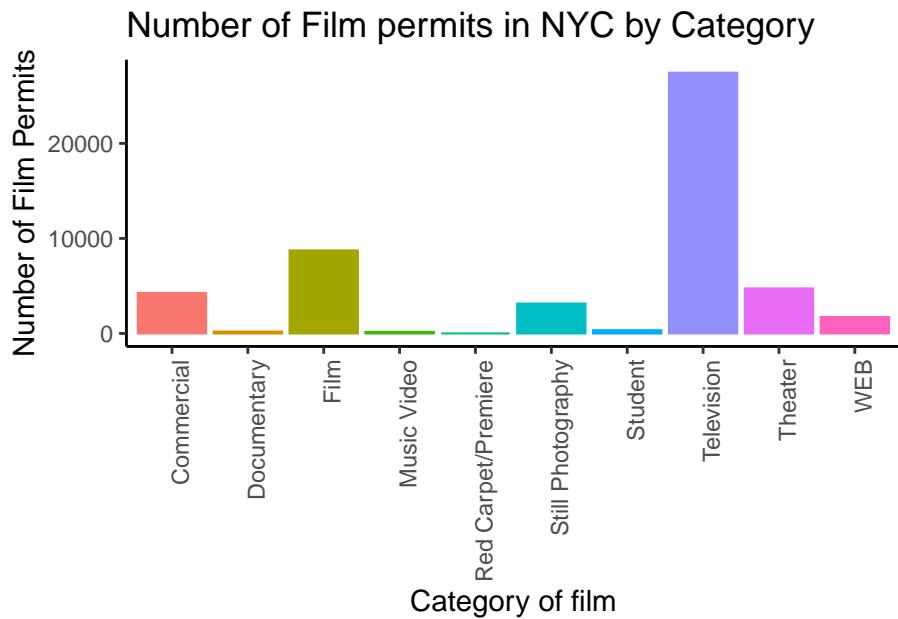
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



1.2.5.9 Font-size

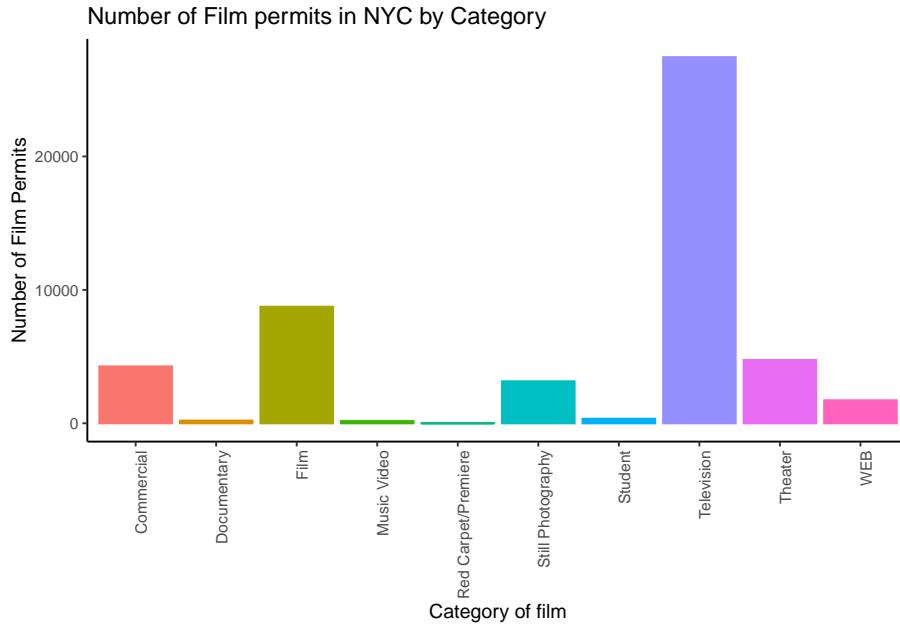
Changing font-size is often something you want to do. ggplot2 can do this in different ways. I suggest using the `base_size` option inside `theme_classic()`. You set one number for the largest font size in the graph, and everything else gets scaled to fit with that first number. It's really convenient. Look for the inside of `theme_classic()`

```
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 15) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



or make things small... just to see what happens

```
ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



1.2.5.10 ggplot2 summary

That's enough of the ggplot2 basics for now. You will discover that many things are possible with ggplot2. It is amazing. We are going to get back to answering some questions about the data with graphs. But, now that we have built the code to make the graphs, all we need to do is copy-paste, and make a few small changes, and boom, we have our graph.

1.2.6 More questions about NYC films

1.2.6.1 What are the sub-categories of films?

Notice the `nyc_films` data frame also has a column for `SubCategoryName`. Let's see what's going on there with a quick plot.

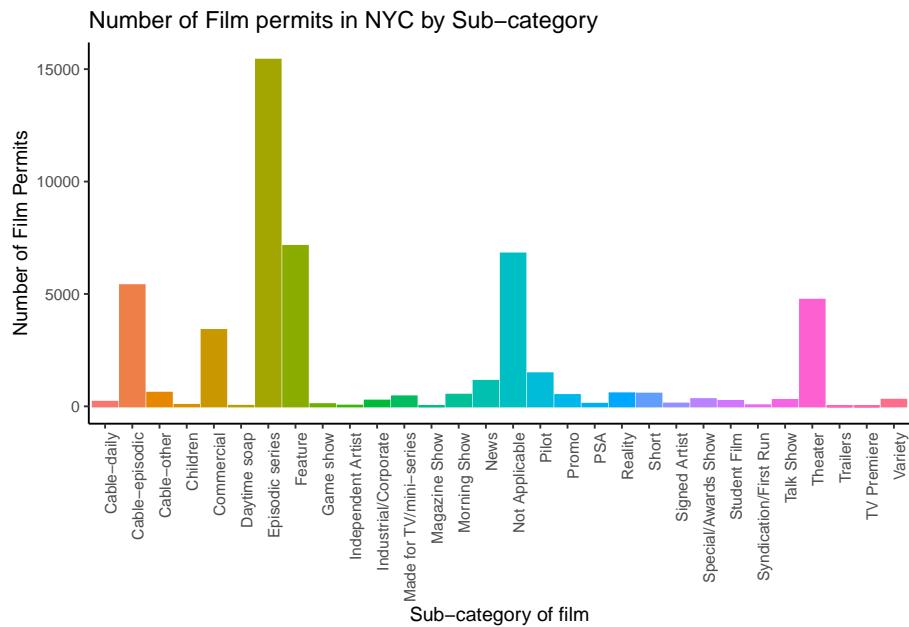
```
# get the counts (this is a comment it's just here for you to read)

counts <- nyc_films %>%
  group_by(SubCategoryName) %>%
  summarize(count_of_permits = length(SubCategoryName))

# make the plot

ggplot(counts, aes(x = SubCategoryName, y = count_of_permits,
  color=SubCategoryName,
  fill= SubCategoryName )) +
```

```
geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Sub-category of film") +
  ggtitle("Number of Film permits in NYC by Sub-category") +
  theme(legend.position="none")
```



I guess “episodic series” are the most common. Using a graph like this gave us our answer super fast.

1.2.6.2 Categories by different Boroughs

Let's see one more really useful thing about ggplot2. It's called `facet_wrap()`. It's an ugly word, but you will see that it is very cool, and you can do next-level-super-hero graph styles with `facet_wrap` that other people can't do very easily.

Here's our question. We know that some films are made in different Boroughs, and that same films are made in different categories, but do different Boroughs have different patterns for the kinds of categories of films they request permits for? Are there more TV shows in Brooklyn? How do we find out? Watch, just like this:

```
# get the counts (this is a comment it's just here for you to read)
```

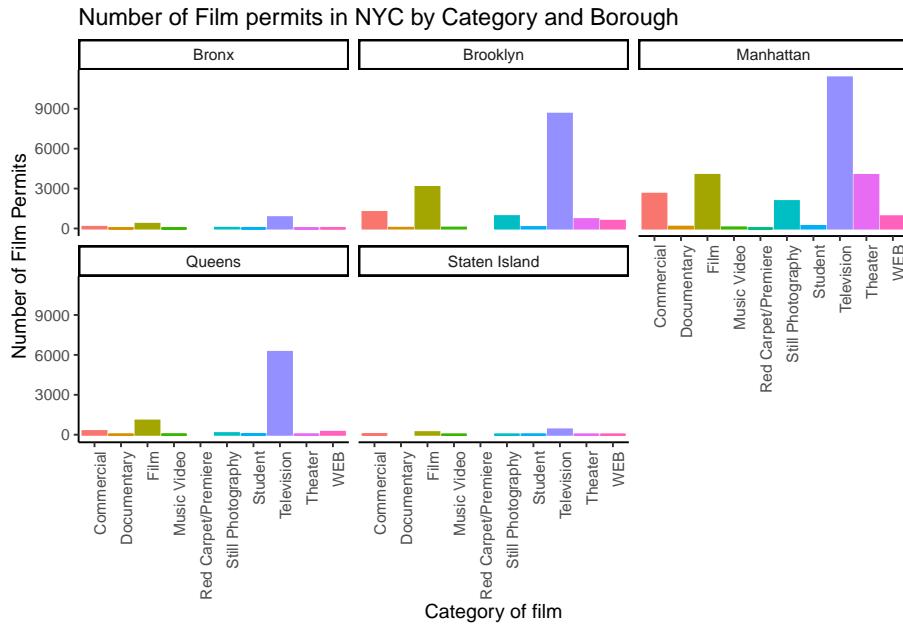
```

counts <- nyc_films %>%
  group_by(Borough, Category) %>%
  summarize(count_of_permits = length(Category))

# make the plot

ggplot(counts, aes(x = Category, y = count_of_permits,
                    color=Category,
                    fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category and Borough") +
  theme(legend.position="none") +
  facet_wrap(~Borough, ncol=3)

```

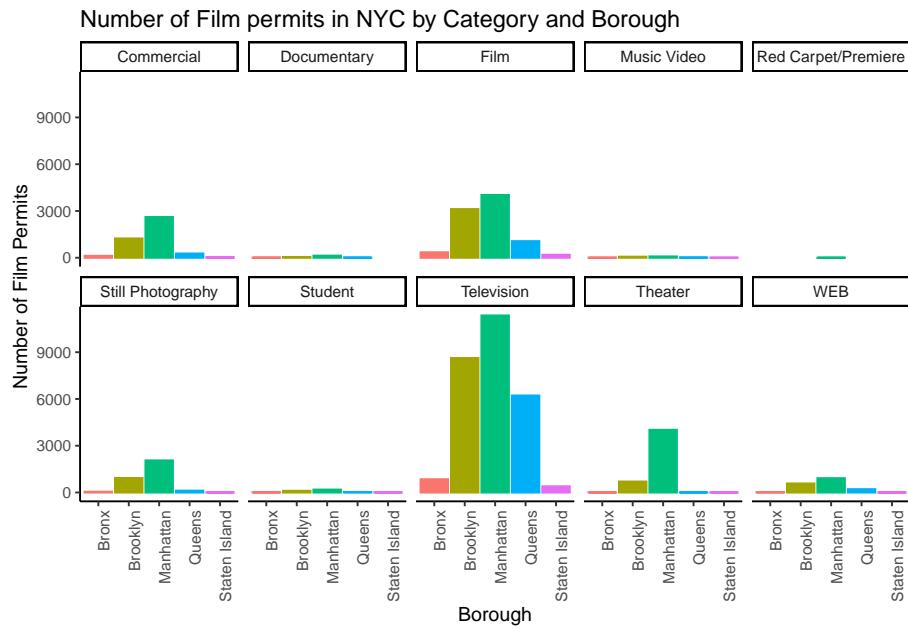


We did two important things. First we added `Borough` and `Category` into the `group_by()` function. This automatically gives separate counts for each category of film, for each Borough. Then we added `facet_wrap(~Borough, ncol=3)` to the end of the plot, and it automatically drew us 5 different bar graphs, one for each Borough! That was fast. Imagine doing that by hand.

The nice thing about this is we can switch things around if we want. For

example, we could do it this way by switching the Category with Borough, and facet-wrapping by Category instead of Borough like we did above. Do what works for you.

```
ggplot(counts, aes(x = Borough, y = count_of_permits,
                    color=Borough,
                    fill= Borough )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Borough") +
  ggtitle("Number of Film permits in NYC by Category and Borough") +
  theme(legend.position="none") +
  facet_wrap(~Category, ncol=5)
```



1.2.7 Gapminder Data

<https://www.gapminder.org> is an organization that collects some really interesting worldwide data. They also make cool visualization tools for looking at the data. There are many neat examples, and they have visualization tools built right into their website that you can play around with <https://www.gapminder.org/tools/>. That's fun check it out.

There is also an R package called `gapminder`. When you install this package, it loads in some of the data from gapminder, so we can play with it in R.

If you don't have the gapminder package installed, you can install it by running this code

```
install.packages("gapminder")
```

Once the package is installed, you need to load the new library, like this. Then, you can put the `gapminder` data into a data frame, like we do here: `gapminder_df`.

```
library(gapminder)
gapminder_df <- gapminder
```

1.2.7.1 Look at the data frame

You can look at the data frame to see what is in it, and you can use `summarytools` again to view a summary of the data.

```
view(dfSummary(gapminder_df))
```

There are 1704 rows of data, and we see some columns for country, continent, year, life expectancy, population, and GDP per capita.

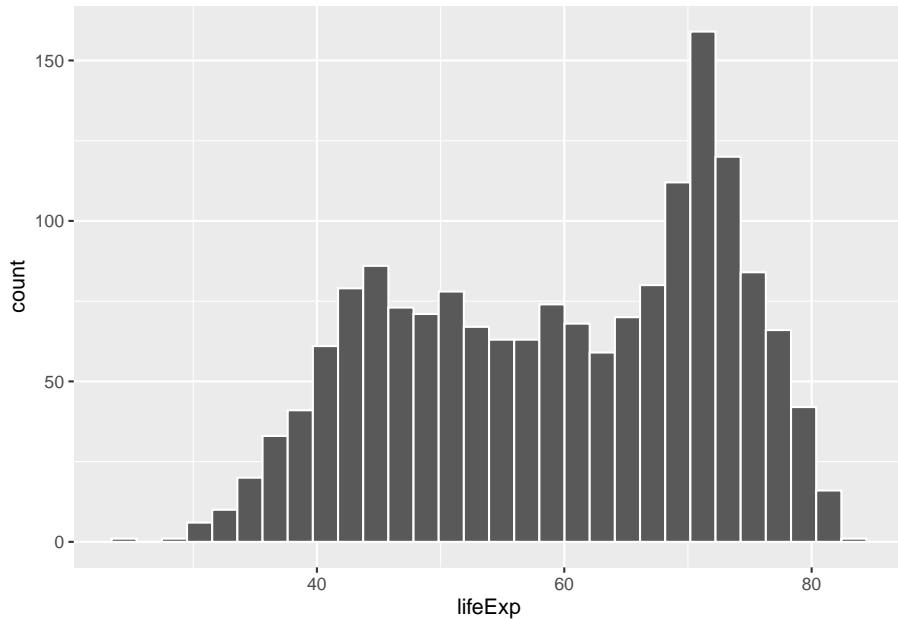
1.2.8 Asking Questions with the gap minder data

We will show you how to graph some the data to answer a few different kinds of questions. Then you will form your own questions, and see if you can answer them with `ggplot2` yourself. All you will need to do is copy and paste the following examples, and change them up a little bit

1.2.8.1 Life Expectancy histogram

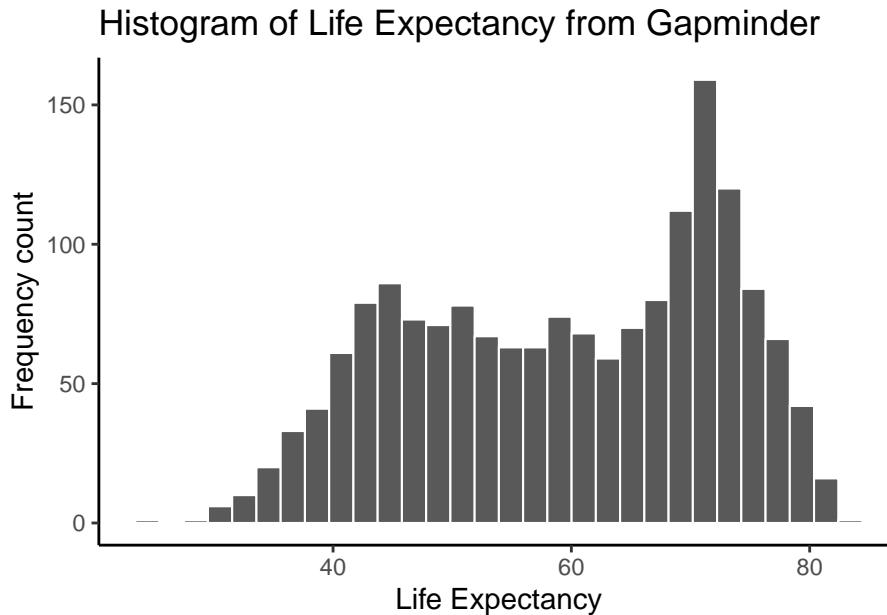
How long are people living all around the world according to this data set? There are many ways we could plot the data to find out. The first way is a histogram. We have many numbers for life expectancy in the column `lifeExp`. This is a big sample, full of numbers for 142 countries across many years. It's easy to make a histogram in `ggplot` to view the distribution:

```
ggplot(gapminder_df, aes(x=lifeExp))+
  geom_histogram(color="white")
```



See, that was easy. Next, is a code block that adds more layers and settings if you wanted to modify parts of the graph:

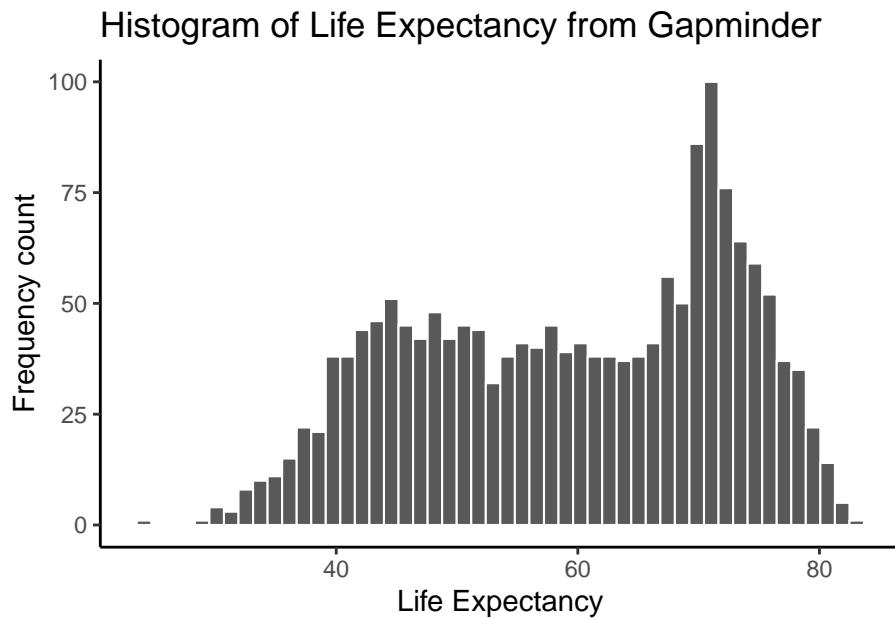
```
ggplot(gapminder_df, aes(x = lifeExp)) +  
  geom_histogram(color="white") +  
  theme_classic(base_size = 15) +  
  ylab("Frequency count") +  
  xlab("Life Expectancy") +  
  ggtitle("Histogram of Life Expectancy from Gapminder")
```



The histogram shows a wide range of life expectancies, from below 40 to just over 80. Histograms are useful, they can show you what kinds of values happen more often than others.

One final thing about histograms in ggplot. You may want to change the bin size. That controls how wide or narrow, or the number of bars (how they split across the range), in the histogram. You need to set the `bins=` option in `geom_histogram()`.

```
ggplot(gapminder_df, aes(x = lifeExp)) +
  geom_histogram(color="white", bins=50) +
  theme_classic(base_size = 15) +
  ylab("Frequency count") +
  xlab("Life Expectancy") +
  ggtitle("Histogram of Life Expectancy from Gapminder")
```



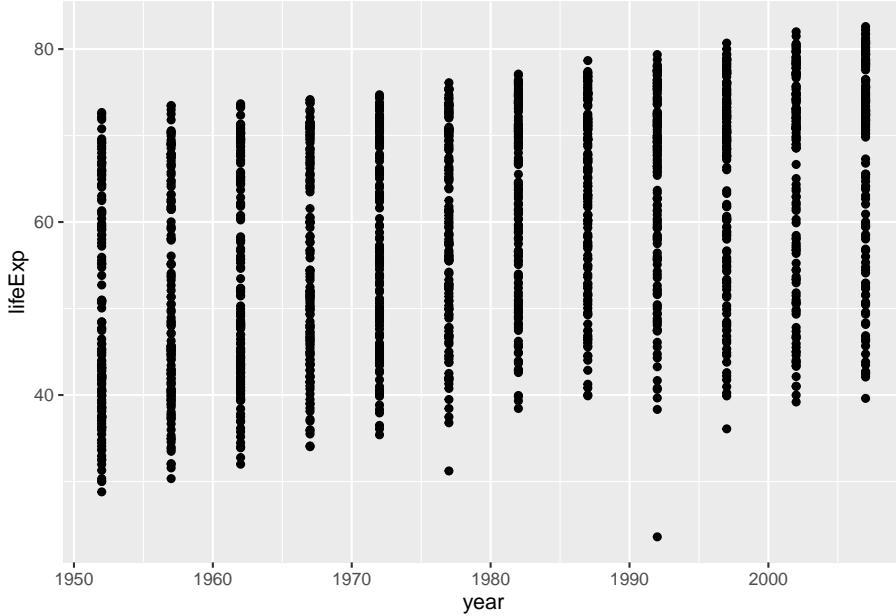
See, same basic pattern, but now breaking up the range into 50 little equal sized bins, rather than 30, which is the default. You get to choose what you want to do.

1.2.8.2 Life Expectancy by year Scatterplot

We can see we have data for life expectancy and different years. So, does worldwide life expectancy change across the years in the data set? As we go into the future, are people living longer?

Let's look at this using a scatter plot. We can set the x-axis to be year, and the y-axis to be life expectancy. Then we can use `geom_point()` to display a whole bunch of dots, and then look at them. Here's the simple code:

```
ggplot(gapminder_df, aes(y= lifeExp, x= year))+
  geom_point()
```



Whoa, that's a lot of dots! Remember that each country is measured each year. So, the bands of dots you see, show the life expectancies for the whole range of countries within each year of the database. There is a big spread inside each year. But, on the whole it looks like groups of dots slowly go up over years.

1.2.8.3 One country, life expectancy by year

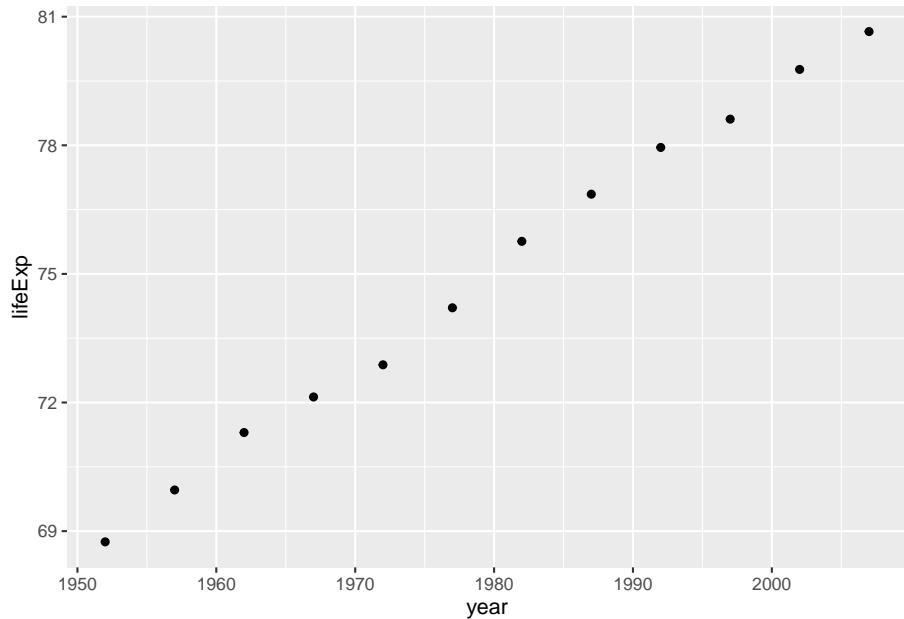
I'm (Matt) from Canada, so maybe I want to know if life expectancy for Canadians is going up over the years. To find out the answer for one country, we first need to split the full data set, into another smaller data set that only contains data for Canada. In other words, we want only the rows where the word "Canada" is found in the `country` column. We will use the `filter` function from `dplyr` for this:

```
# filter rows to contain Canada

smaller_df <- gapminder_df %>%
  filter(country == "Canada")

# plot the new data contained in smaller_df

ggplot(smaller_df, aes(y= lifeExp, x= year))+
  geom_point()
```



I would say things are looking good for Canadians, their life expectancy is going up over the years!

1.2.8.4 Multiple countries scatterplot

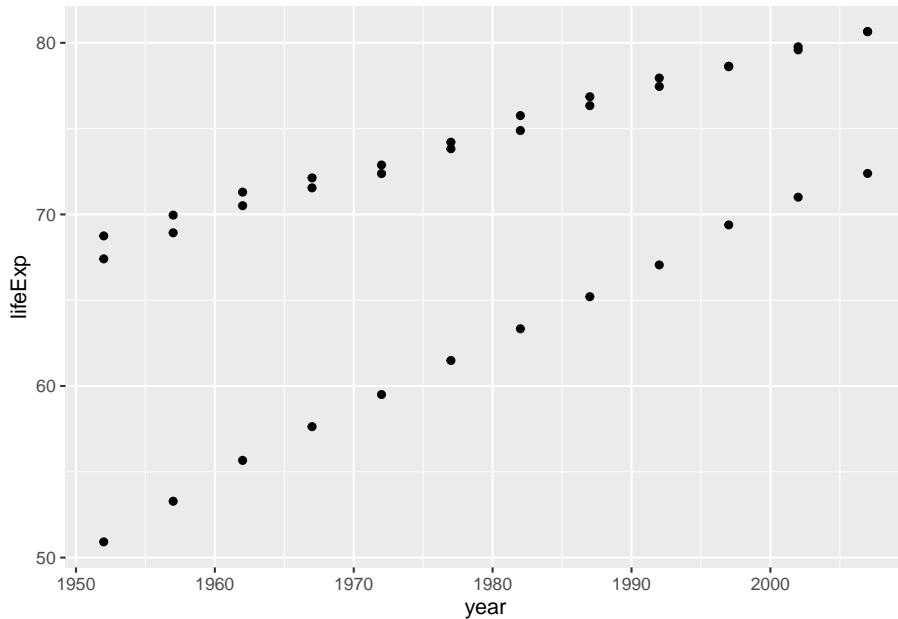
What if we want to look at a few countries altogether. We can do this too. We just change how we filter the data so more than one country is allowed, then we plot the data. We will also add some nicer color options and make the plot look pretty. First, the simple code:

```
# filter rows to contain countries of choice

smaller_df <- gapminder_df %>%
  filter(country %in% c("Canada", "France", "Brazil") == TRUE)

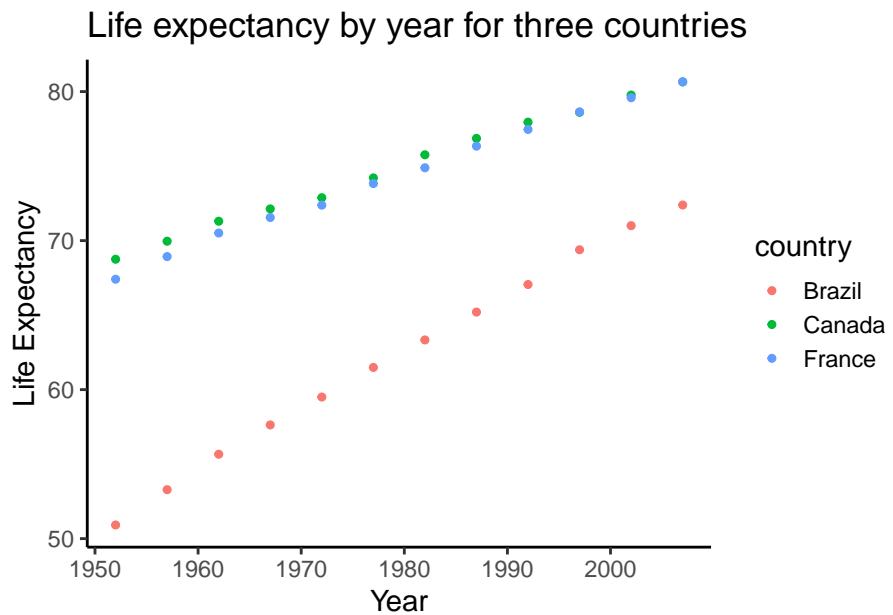
# plot the new data contained in smaller_df

ggplot(smaller_df, aes(y= lifeExp, x= year, group= country))+
  geom_point()
```



Nice, we can now see three sets of dots, but which are countries do they represent? Let's add a legend, and make the graph better looking.

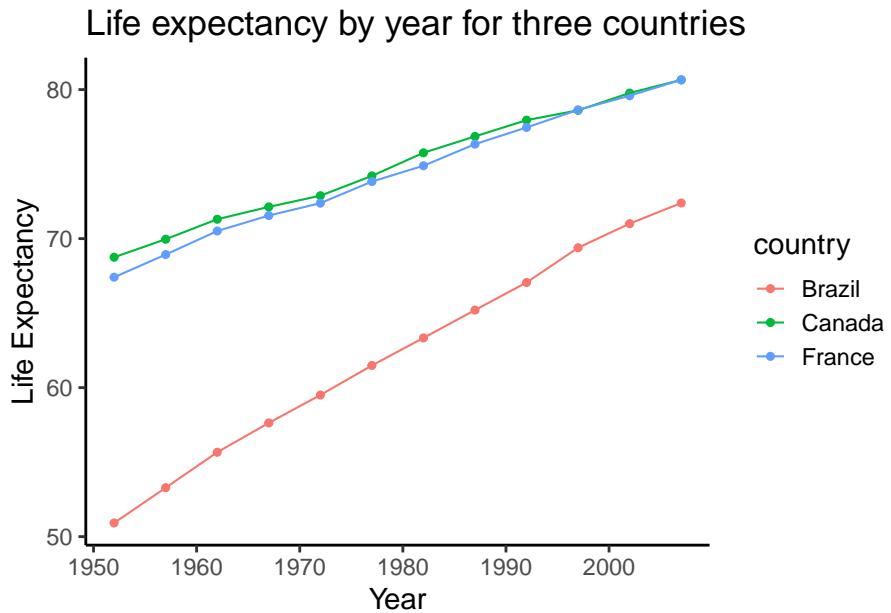
```
ggplot(smaller_df,aes(y= lifeExp, x= year,
                      group= country, color = country)) +
  geom_point()+
  theme_classic(base_size = 15) +
  ylab("Life Expectancy") +
  xlab("Year") +
  ggtitle("Life expectancy by year for three countries")
```



1.2.8.5 geom_line() connecting the dots

We might also want to connect the dots with a line, to make it easier to see the connection! Remember, ggplot2 draws layers on top of layers. So, we add in a new `geom_line()` layer.

```
ggplot(smaller_df,aes(y= lifeExp, x= year,
                      group= country, color = country)) +
  geom_point()+
  geom_line()+
  theme_classic(base_size = 15) +
  ylab("Life Expectancy") +
  xlab("Year") +
  ggtitle("Life expectancy by year for three countries")
```



1.2.9 Generalization Exercise

The following generalization exercise and writing assignment is also in your lab R Markdown document for this lab. Complete your work in that document and hand it in.

(1 point - Pass/Fail)

Use the code from above to attempt to solve the extra things we ask you do for this assignment. You generalization exercises are as follows:

1. Make a graph plotting Life Expectancy by year for the five continents, using the `continent` factor. Make sure you change the title so it reads correctly
2. Make a graph plotting GDP per capita by year for the USA, Canada, and Mexico. Use the `gdpPercap` column for the GDP per capita data
3. Make a new graph plotting anything you are interested in using the gapminder dataset. It just needs to be a plot that we have not given an example for

1.2.10 Writing assignment

Complete the writing assignment described in your R Markdown document for this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

The question for this lab is a long answer question about histograms. Here is the question:

Describe what histograms are, how to interpret them, and what they are useful for. You should answer each of these questions:

The answers to each of these questions are worth .25 points each, for a total of 2 points

- a. What do the bars on a histogram represent?
- b. How many bars can a histogram have?
- c. What do the heights of the bars tell you
- d. What is on the x-axis and y-axis of a histogram
- e. What does the tallest bar on a histogram tell you?
- f. What does the shortest bar on a histogram tell you?
- g. What are some uses for histograms, why would you want to look at a histogram of some numbers that you collected?
- h. Imagine you had two histograms, one was very wide and spread out, the other was very narrow with a very tall peak. Which histogram would you expect to contain more consistent numbers (numbers that are close to each other), explain why.

Rubric

General grading.

- You will receive 0 points for missing answers (say, if you do not answer question c, then you will receive 0 out .25 points for that question)
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points. For example, if you incorrectly describe what the x and y-axes refer to, then you will receive 0 points for that question.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

1.3 Excel

1.4 SPSS

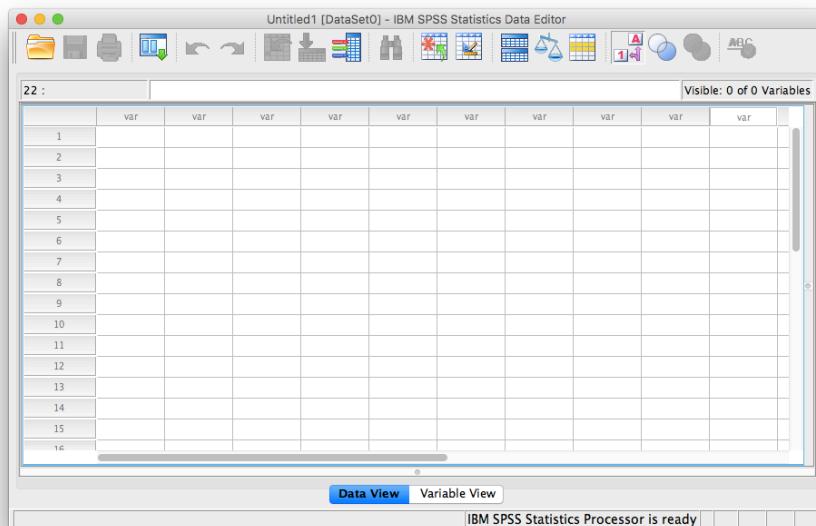
In this lab, we will get you acquainted with the SPSS software layout and graph some sample data to make sense of it. We will be doing the following:

1. Opening SPSS and the SPSS layout
2. Reviewing variable properties and the Variable View tab
3. Opening a data file and producing different types of graphs

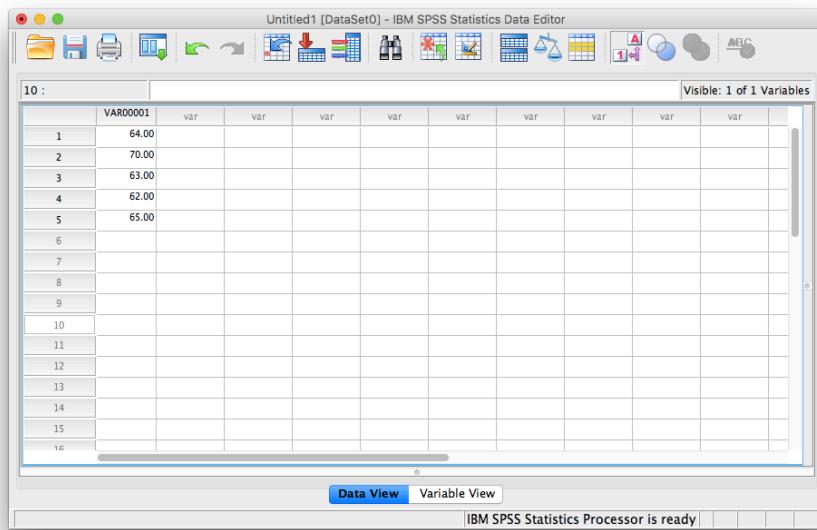
1.4.1 Opening SPSS and the SPSS layout

Your lab instructor will take you through the process of opening the SPSS program. You may double-click on its icon located on the desktop of your lab computer, or you may find it using the Start menu. Once the program loads, you will be prompted with a pop-up window that asks you which file you would like to open. For now, we will be examining the basic layout of SPSS without a data set, so you can click Cancel.

Once you do, the main SPSS spreadsheet should open. It will look like this, a basic spreadsheet:

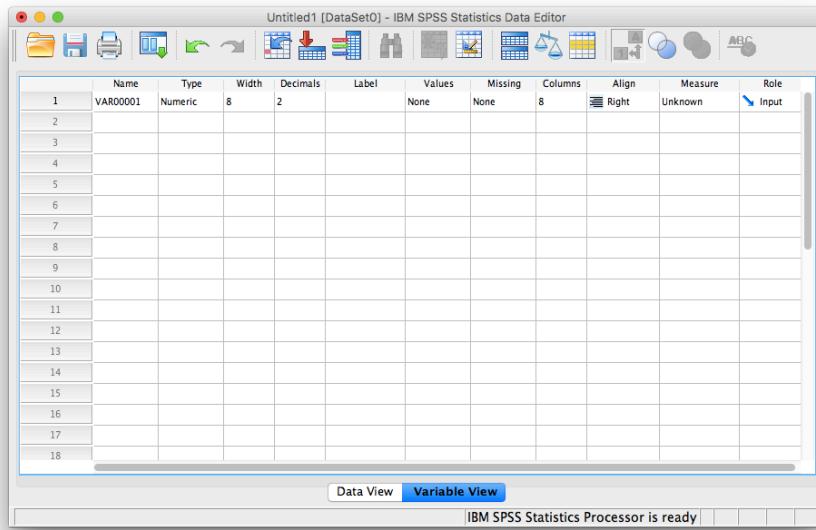


Notice at the bottom of your window there are two tabs; “Data View” and “Variable View”. In data view, we enter data into our spreadsheet. You will notice that rows are numbered on the left-hand side of the spreadsheet, while columns are labeled “var”. This is an indication of the general structure of SPSS: Variables are contained in the columns, and rows indicate individual observations. For example, if you obtained the heights (in inches) of 5 people {x= 64, 70, 63, 62, 65} and wanted to enter their data into SPSS, each person’s height would be entered in a new row, not across the columns, as seen below:



1.4.2 Reviewing variable properties and the Variable View tab

Now that we have some data entered, we might want to name our variable so that it's evident our measurements represent heights. In order to view or modify variable names and other properties, look to the bottom of your SPSS window and switch over to the "Data View" tab. Once you do this, your window will appear as follows:



Here, you can edit the name of your variables, and specify their properties. Variable names can be anything you like, with the restriction that you cannot use numbers or spaces. Next, notice several other important properties of variables you may at some point need to set or modify:

- Name: the name of your variable that will appear as a column header in Data View. No spaces or numerals.
- Type: Your data will most often be Numeric, but sometimes, as in data representing currency or data in scientific notation, you may change the data type appropriately. If your data is simply a label, word, or response (such as an open-ended response to a survey question), choose “String”: this tells SPSS not to treat this variable as a number. (Nota bene: if you select the wrong type of variable, SPSS may not be able to process your requested calculations, so always remember to check this parameter!)
- Width: This refers to how many digits will be visible by default.
- Decimals: This refers to how many decimal places will be visible by default.
- Label: This is a description of the variable. Any information too long to be included in the variable name goes here.
- Values: For nominal scale data, let’s say 1 represents male and 2 represents female, this is where you include the values and their corresponding labels.
- Measure: This variable property allows you to specify the nature of your data. Depending on the kind of scale you are using, you will choose a different measure type. Nominal and ordinal are chosen for nominal and ordinal scales, respectively. “Scale” is used when your data is measured on a ratio or interval scale. (Nota bene: this “Measure” designation is only

a marker; it does not affect the calculations as in the “Type” parameter. Even if you choose the wrong icon/label for “Measure”, SPSS will still produce the correct output)

1.4.3 Opening a data file and producing different types of graphs

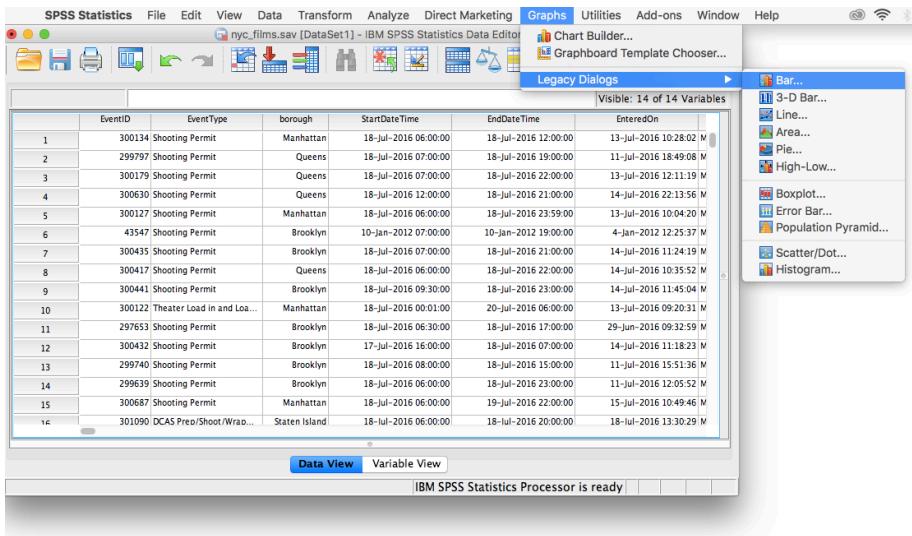
Now that we know about the properties of the SPSS spreadsheet window, let’s open a data file and learn how to make some sense of it by creating different types of graphs. Here is a link to an SPSS-based data file containing information about film permits (requests made by film companies to shoot TV shows and movies on location) filed in New York City. The file is named nyc_films.sav.

Once you open the data file, browse through to familiarize yourself with the variables that are being measured. Switch over to Variable View for details of each variable.

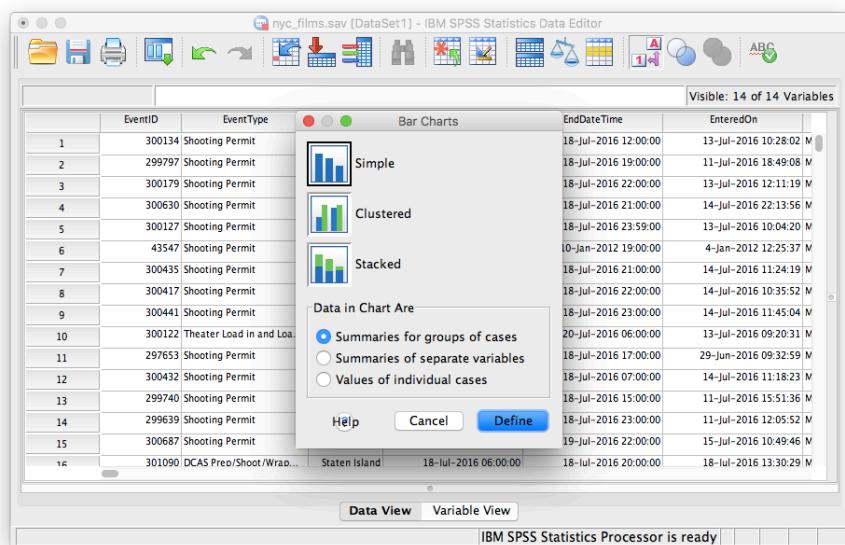
1.4.3.1 Bar Graphs

Now, back to Data View. We will not be working with every single variable in this spreadsheet, but we’ll select a few interesting ones with which to answer questions. Let’s start with `borough`. Suppose we wanted to know which borough receives the most film permits (you can probably guess which one is most popular). Let’s use SPSS to produce a graph to answer this question. With your data file open, go up to the top menu and choose Graphs, then Legacy Dialogs. You will see an entire list of possible graphs we can use to plot our data.

Let’s think about the nature of our question: we would like to know how many permits were filed for each borough. Borough is simply a label or a name for a region, and we want to know the frequency of permits for each borough. This is a nominal scale variable and so, we will appropriately choose a BAR graph to plot it. Select Bar...

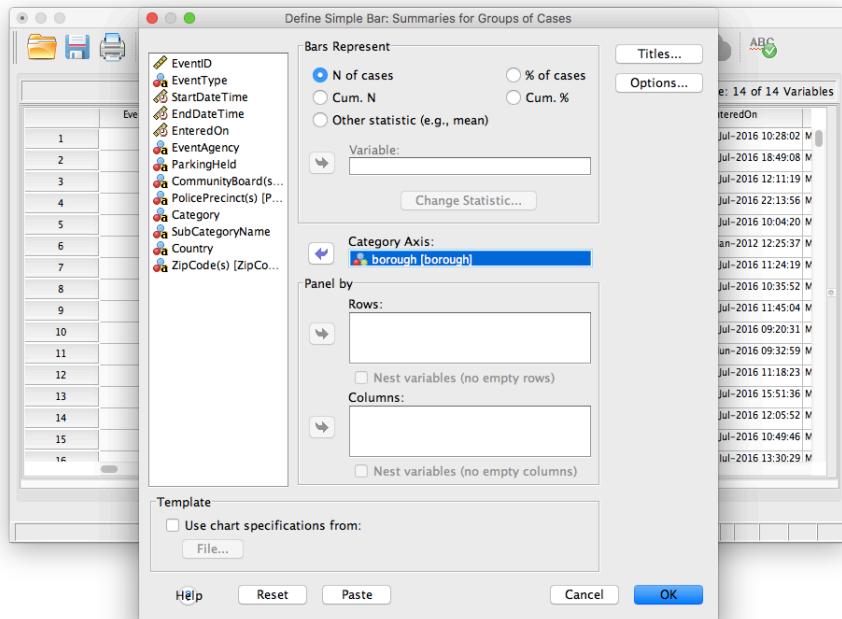


The next window will ask you to specify what kind of graph you would like. Se-

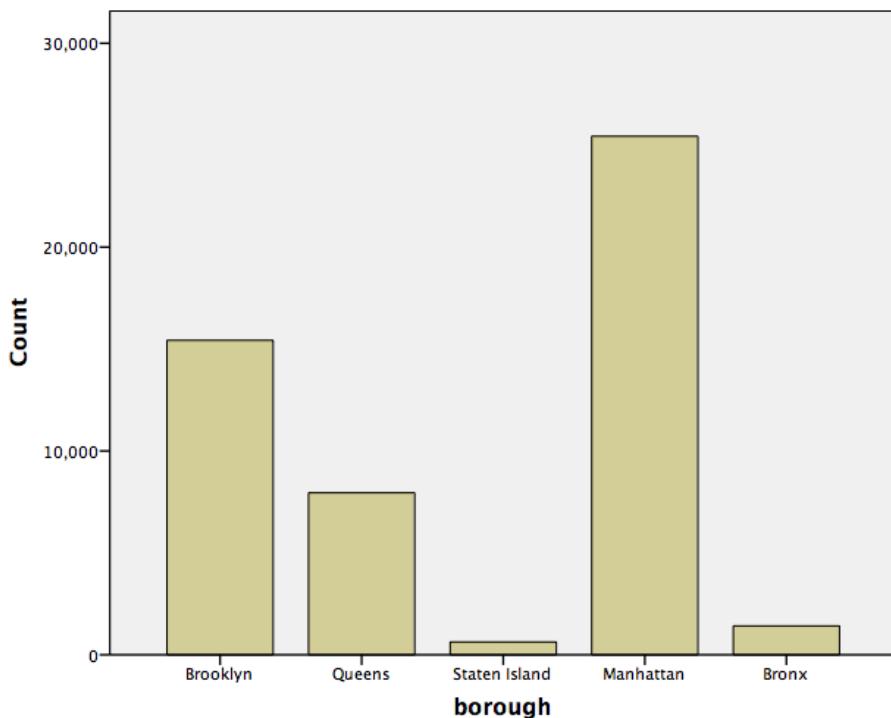


lect Simple and then Define.

The following window will ask which variable you'd like to plot. Select **borough** from the left-hand list and use the arrow to move it into the field labeled "Category Axis". Then click OK.



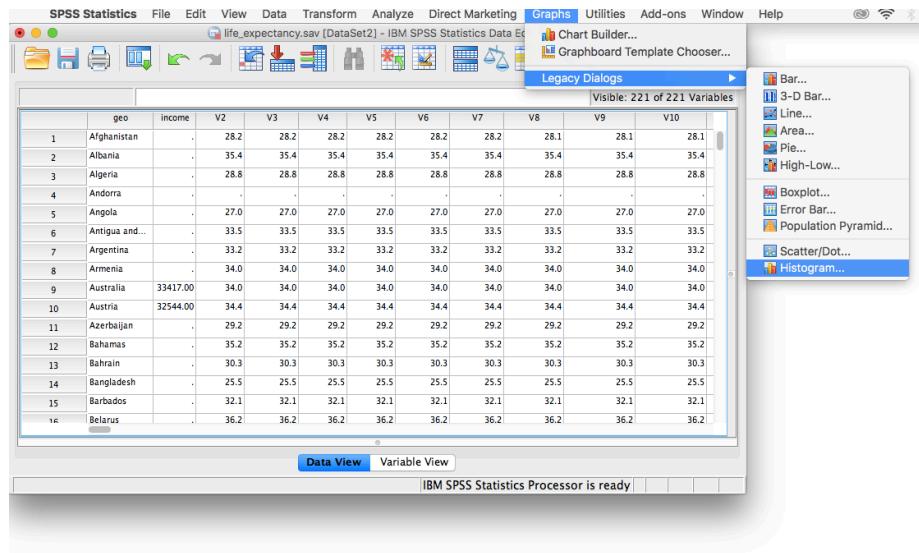
SPSS will produce a new output window which will contain the bar graph you have generated. Notice which borough receives the most film permits. Are you surprised?



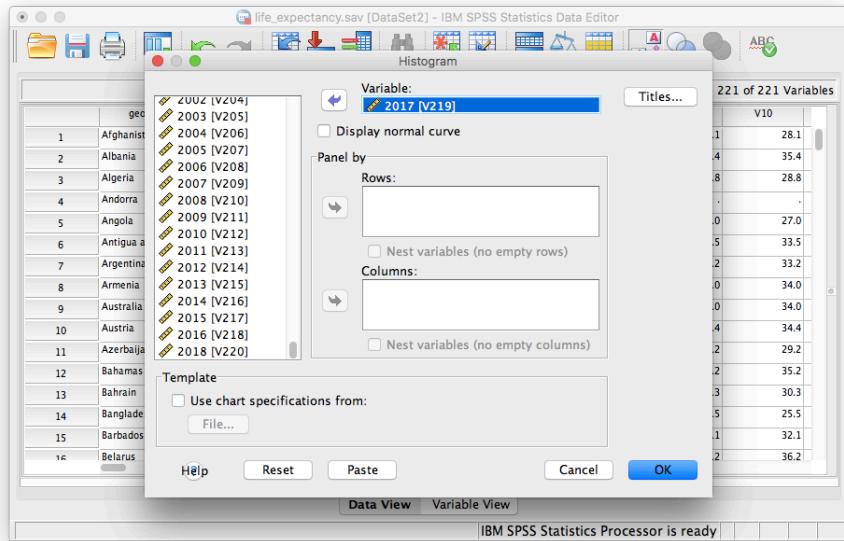
1.4.3.2 Histograms

Now, let's use a different data set to plot a histogram. The defining difference between a histogram and a bar graph (although they look very similar as they both utilize bars) is that a histogram is used to display a continuous variable (interval or ratio scale). In the previous example, boroughs were simply labels or names, so we used a nominal scale and therefore a bar graph. Here, we will deal with life expectancy (measured in years), an interval scale measure. Here is a link to the SPSS data file, [life_expectancy.sav](#). Open this file and examine its rows and columns. Each column represents a year during which life expectancy was measured. Each row represents a different country.

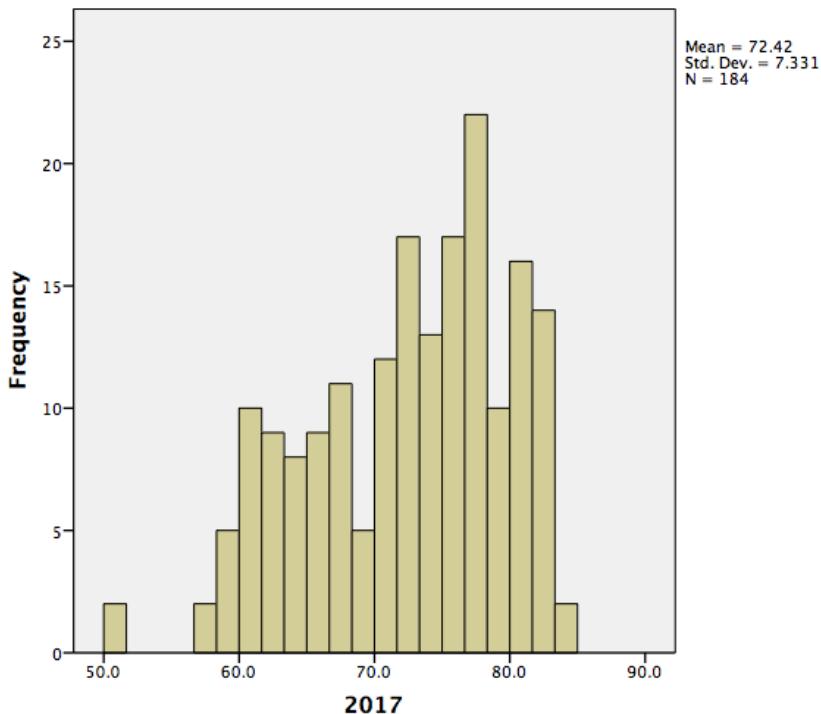
Let's first get an idea about life expectancy in general. We want to plot a histogram with life expectancy on the x-axis and frequency on the y-axis. Choose Graphs in the top menu, then Legacy Dialogs. From here, remember we want a histogram, not a bar graph, so let's select Histogram....



The window that appears contains every variable in your spreadsheet listed on the left-hand side. We can choose one variable at a time to plot. Let's scroll all the way down the list and choose 2017 [v219]. This is the variable containing life expectancies for the year 2017. Using the arrow, move that variable into the field labeled "Variable:", then click OK.



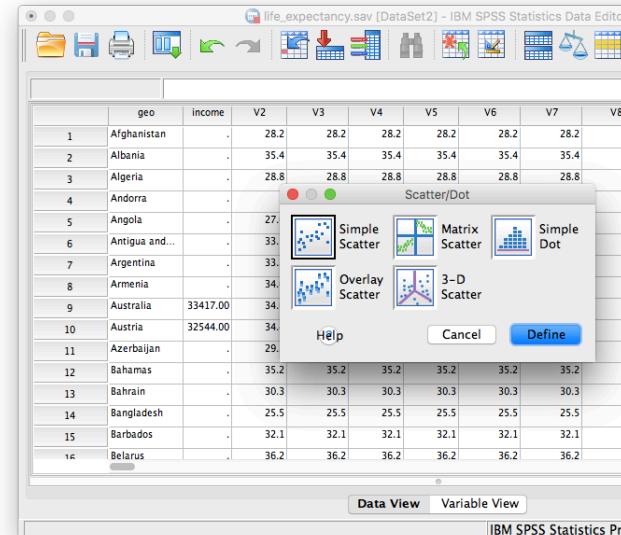
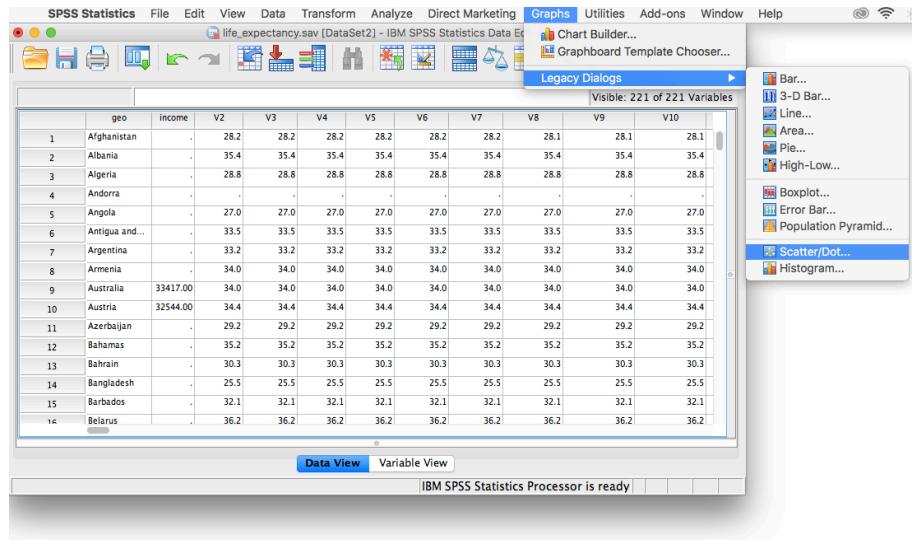
SPSS will produce an output window containing the distribution of life expectancy for the year 2017.



1.4.3.3 Scatterplots

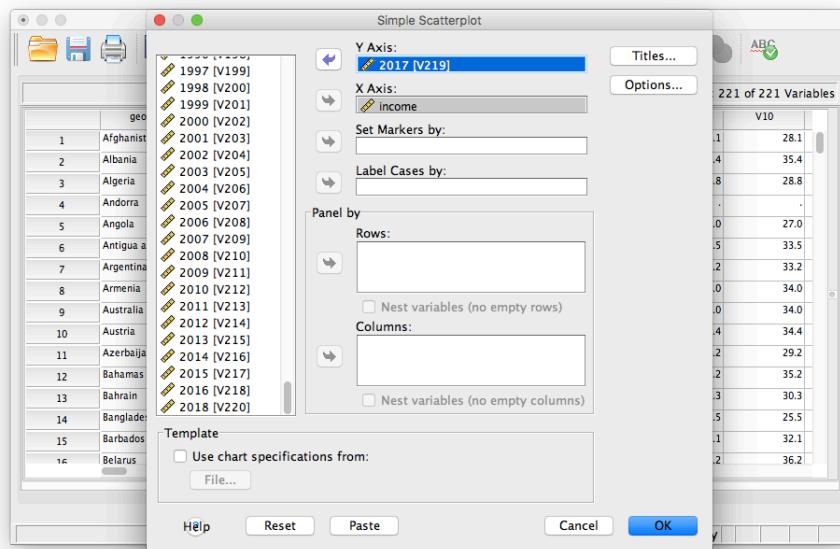
Now, we will look to a different type of data plot; the scatterplot. A scatterplot allows us to visualize bivariate data, that is, data for which there are two measurements per individual. For example, we may ask whether life expectancy in a country (or how long you live, on average) is related to the average income. Using the `life_expectancy.sav` data file, let's plot both variables: `2017` [v219] and `income`. The income variable in the spreadsheet refers to data collected in 2017 by the Better Life Initiative. Notice not all the countries listed have estimates for average annual income. For those that do, this value represents household net adjusted income (annual) in US dollars.

To create the scatterplot, let's go to Graphs in the menu toolbar, then Legacy Dialogs, then Scatter.

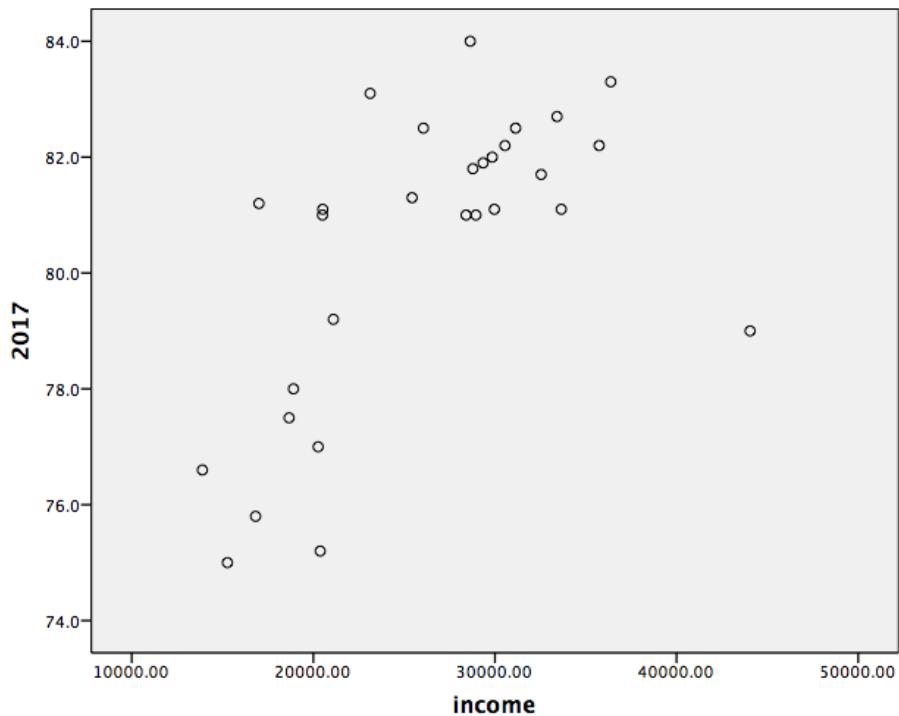


You will choose Simple scatter, then click Define.

Next, indicate which variables (there are 2 this time!) you would like in the x- and y-axes. Use the arrows to place `income` in the x-axis field, and `2017 (V219)` in the y-axis field. (For the purposes of graphing a scatterplot, it does not matter which variable goes into the y-axis and x-axis fields for now; you can reverse them if you'd like and you can still interpret the data similarly)



Then click OK. SPSS will produce output containing a scatterplot. What relationship do you notice? What happens to life expectancy the more individuals earn, on average?



1.4.4 Practice Problems

-
1. Create a histogram for life expectancy in the year 1800. Describe the distribution. How does it differ from the one we plotted for 2017?
 2. Plot the life expectancy of each country in 1800 vs. that of 2018. What does this graph show you? What are your conclusions regarding the development of nations?

1.5 JAMOVI

Chapter 2

Lab 2: Descriptive Statistics

Describing comic sensibility is near impossible. It's sort of an abstract silliness, that sometimes the joke isn't the star. —Dana Carvey

The purpose of this lab is to show you how to compute basic descriptive statistics, including measures of central tendency (mean, mode, median) and variation (range, variance, standard deviation).

2.1 General Goals

1. Compute measures of central tendency using software
2. Compute measures of variation using software
3. Ask some questions of a data set using descriptive statistics

2.1.1 Important info

We will be using data from the gapminder project. You can download a small snippet of the data in .csv format from this link (note this dataset was copied from the gapminder library for R) [gapminder.csv](#). If you are using R, then you can install the gapminder package. This method is described later in the R section.

2.2 R

2.2.1 Descriptives basics in R

We learned in lecture and from the textbook that data we want to use ask and answer questions often comes with loads of numbers. Too many numbers to

look at all at once. That's one reason we use descriptive statistics. To reduce the big set of numbers to one or two summary numbers that tell us something about all of the numbers. R can produce descriptive statistics for you in many ways. There are base functions for most of the ones that you want. We'll go over some R basics for descriptive statistics, and then use our new found skills to ask some questions about real data.

2.2.1.1 Making numbers in R

In order to do descriptive statistics we need to put some numbers in a variable. You can also do this using the `c()` command, which stands for combine

```
my_numbers <- c(1,2,3,4)
```

There are a few other handy ways to make numbers. We can use `seq()` to make a sequence. Here's making the numbers from 1 to 100

```
one_to_one_hundred <- seq(1,100,1)
```

We can repeat things, using `rep`. Here's making 10 5s, and 25 1s:

```
rep(10,5)
```

```
## [1] 10 10 10 10 10
```

```
rep(1,25)
```

```
## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
all_together_now <- c(rep(10,5),rep(1,25))
```

2.2.1.2 Sum

Let's play with the number 1 to 100. First, let's use the `sum()` function to add them up

```
one_to_one_hundred <- seq(1,100,1)
sum(one_to_one_hundred)
```

```
## [1] 5050
```

2.2.1.3 Length

We put 100 numbers into the variable `one_to_one_hundred`. We know how many numbers there are in there. How can we get R to tell us? We use `length()` for that.

```
length(one_to_one_hundred)
```

```
## [1] 100
```

2.2.2 Central Tendency

2.2.2.1 Mean

Remember the mean of some numbers is their sum, divided by the number of numbers. We can compute the mean like this:

```
sum(one_to_one_hundred)/length(one_to_one_hundred)
```

```
## [1] 50.5
```

Or, we could just use the `mean()` function like this:

```
mean(one_to_one_hundred)
```

```
## [1] 50.5
```

2.2.2.2 Median

The median is the number in the exact middle of the numbers ordered from smallest to largest. If there are an even number of numbers (no number in the middle), then we take the number in between the two (decimal .5). Use the `median` function. There's only 3 numbers here. The middle one is 2, that should be the median

```
median(c(1,2,3))
```

```
## [1] 2
```

2.2.2.3 Mode

R does not have a base function for the Mode. You would have to write one for yourself. Here is an example of writing your own mode function, and then using it. Note I searched how to do this on Google, and am using the mode defined by this answer on stack overflow

Remember, the mode is the most frequently occurring number in the set. Below 1 occurs the most, so the mode will be one.

```
my_mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

my_mode(c(1,1,1,1,1,1,2,3,4))
```

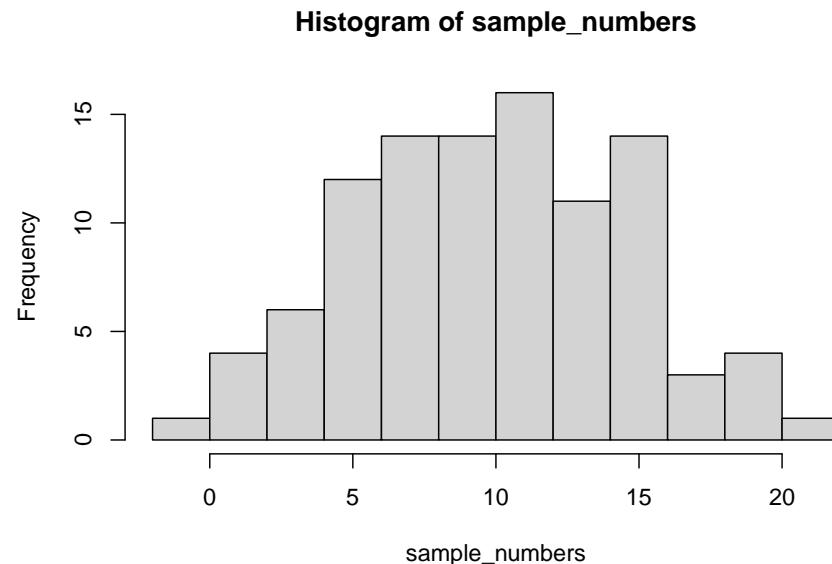
```
## [1] 1
```

2.2.3 Variation

We often want to know how variable the numbers are. We are going to look at descriptive statistics to describe this such as the **range**, **variance**, the **standard deviation**, and a few others.

First, let's remind ourselves what variation looks like (it's when the numbers are different). We will sample 100 numbers from a normal distribution (don't worry about this yet), with a mean of 10, and a standard deviation of 5, and then make a histogram so we can see the variation around 10..

```
sample_numbers <- rnorm(100,10,5)
hist(sample_numbers)
```



2.2.3.1 range

The range is the minimum and maximum values in the set, we use the **range** function.

```
range(sample_numbers)
```

```
## [1] -0.5491855 21.7348315
```

2.2.3.2 var = variance

We can find the sample variance using **var**. Note, divides by $(n-1)$

```
var(sample_numbers)
```

```
## [1] 21.84678
```

2.2.3.3 sd = standard deviation

We find the sample standard deviation us SD. Note, divides by (n-1)

```
sd(sample_numbers)
```

```
## [1] 4.674054
```

Remember that the standard deviation is just the square root of the variance, see:

```
sqrt(var(sample_numbers))
```

```
## [1] 4.674054
```

2.2.3.4 All Descriptives

Let's put all of the descriptives and other functions so far in one place:

```
sample_numbers <- rnorm(100,10,5)
```

```
sum(sample_numbers)
```

```
## [1] 1013.497
```

```
length(sample_numbers)
```

```
## [1] 100
```

```
mean(sample_numbers)
```

```
## [1] 10.13497
```

```
median(sample_numbers)
```

```
## [1] 9.984776
```

```
my_mode(sample_numbers)
```

```
## [1] 12.30226
```

```
range(sample_numbers)
```

```
## [1] -5.711123 23.625207
```

```
var(sample_numbers)
```

```
## [1] 25.19221
```

```
sd(sample_numbers)
```

```
## [1] 5.019185
```

2.2.4 Descriptives by conditions

Sometimes you will have a single variable with some numbers, and you can use the above functions to find the descriptives for that variable. Other times (most often in this course), you will have a big data frame of numbers, with different numbers in different conditions. You will want to find descriptive statistics for each the sets of numbers inside each of the conditions. Fortunately, this is where R really shines, it does it all for you in one go.

Let's illustrate the problem. Here I make a date frame with 10 numbers in each condition. There are 10 conditions, each labelled, A, B, C, D, E, F, G, H, I, J.

```
scores <- rnorm(100,10,5)
conditions <- rep(c("A","B","C","D","E","F","G","H","I","J"), each =10)
my_df <- data.frame(conditions,scores)
```

If you look at the `my_df` data frame, you will see it has 100 rows, there are 10 rows for each condition with a label in the `conditions` column, and 10 scores for each condition in the `scores` column. What if you wanted to know the mean of the scores in each condition? You would want to find 10 means.

The slow way to do it would be like this:

```
mean(my_df [my_df$conditions=="A",]$scores)
```

```
## [1] 8.102554
```

```
mean(my_df [my_df$conditions=="B",]$scores)
```

```
## [1] 10.85379
```

```
mean(my_df [my_df$conditions=="C",]$scores)
```

```
## [1] 10.93324
```

and then keep going

Nobody wants to do that! Not, me I stopped doing it that way, you should to.

2.2.4.1 group_by and summarise

We can easily do everything all at once using the `group_by` and `summarise` function from the `dplyr` package. Just watch

```
library(dplyr)
```

```
my_df %>%
```

```
group_by(conditions) %>%
  summarise(means = mean(scores))

## # A tibble: 10 x 2
##   conditions means
##   <chr>      <dbl>
## 1 A          8.10 
## 2 B          10.9 
## 3 C          10.9 
## 4 D          10.2 
## 5 E          10.7 
## 6 F          7.77 
## 7 G          10.1 
## 8 H          11.7 
## 9 I          9.48 
## 10 J         8.09
```

A couple things now. First, the print out of this was ugly. Let's fix that. we put the results of our code into a new variable, then we use `knitr::kable` to print it out nicely when we `knit` the document

```
summary_df <- my_df %>%
  group_by(conditions) %>%
  summarise(means = mean(scores))

knitr::kable(summary_df)
```

conditions	means
A	8.102554
B	10.853789
C	10.933236
D	10.154181
E	10.685167
F	7.769095
G	10.139262
H	11.708035
I	9.481652
J	8.092515

2.2.4.2 multiple descriptives

The best thing about the `dplyr` method, is that we can add more than one function, and we'll get more than one summary returned, all in a nice format, let's add the standard deviation:

```
summary_df <- my_df %>%
  group_by(conditions) %>%
```

```

    summarise(means = mean(scores),
              sds = sd(scores))

knitr::kable(summary_df)

```

conditions	means	sds
A	8.102554	4.311988
B	10.853789	5.323515
C	10.933236	4.054272
D	10.154181	4.546303
E	10.685167	6.066749
F	7.769095	5.992941
G	10.139262	6.462827
H	11.708035	8.066638
I	9.481652	3.944036
J	8.092515	6.318956

We'll add the min and the max too:

```

summary_df <- my_df %>%
  group_by(conditions) %>%
  summarise(means = mean(scores),
            sds = sd(scores),
            min = min(scores),
            max = max(scores))

knitr::kable(summary_df)

```

conditions	means	sds	min	max
A	8.102554	4.311988	1.0527033	14.26046
B	10.853789	5.323515	1.6533462	18.60518
C	10.933236	4.054272	3.8850972	16.94131
D	10.154181	4.546303	1.3844868	16.69458
E	10.685167	6.066749	1.4260706	19.50597
F	7.769095	5.992941	-3.4099284	13.88533
G	10.139262	6.462827	-2.1657601	18.18280
H	11.708035	8.066638	-0.0460183	25.33241
I	9.481652	3.944036	3.8785103	14.57990
J	8.092515	6.318956	-5.4123091	18.43951

2.2.5 Describing gapminder

Now that we know how to get descriptive statistics from R, we can do this will some real data. Let's quickly ask a few questions about the gapminder data.

```

library(gapminder)
gapminder_df <- gapminder

```

Note: The above code will only work if you have installed the gapminder package. Make sure you are connected to the internet, then choose the Packages tab from the bottom right panel, and choose install. Then search for gapminder, choose it, and install it.

2.2.5.1 What are some descriptive for Life expectancy by continent?

Copy the code from the last part of descriptives using dplyr, then change the names like this:

```
summary_df <- gapminder_df %>%
  group_by(continent) %>%
  summarise(means = mean(lifeExp),
            sds = sd(lifeExp),
            min = min(lifeExp),
            max = max(lifeExp))

knitr::kable(summary_df)
```

continent	means	sds	min	max
Africa	48.86533	9.150210	23.599	76.442
Americas	64.65874	9.345088	37.579	80.653
Asia	60.06490	11.864532	28.801	82.603
Europe	71.90369	5.433178	43.585	81.757
Oceania	74.32621	3.795611	69.120	81.235

2.2.6 Generalization Exercise

(1 point - Pass/Fail)

Complete the generalization exercise described in your R Markdown document for this lab.

- What is the mean, standard deviation, minimum and maximum life expectancy for all the gapminder data (across all the years and countries). Hint: do not use `group_by`
- What is the mean, standard deviation, minimum and maximum life expectancy for all of the continents in 2007, the most recent year in the dataset. Hint: add another pipe using `filter(year==2007) %>%`

2.2.7 Writing assignment

(2 points - Graded)

Complete the writing assignment described in your R Markdown document for this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

Your writing assignment is to answer these questions in full sentences using simple plain language:

1. Define the mode.
2. Explain what would need to happen in order for a set of numbers to have two modes
3. Define the median
4. Define the mean
5. Define the range
6. When calculating the standard deviation, explain what the difference scores represent
7. Explain why the difference scores are squared when calculating the standard deviation
8. If one set of numbers had a standard deviation of 5, and another had a standard deviation of 10, which set of numbers would have greater variance, explain why.

Rubric

General grading.

- You will receive 0 points for missing answers (say, if you do not answer question c, then you will receive 0 out .25 points for that question)
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

2.3 Excel

How to do it in Excel

2.4 SPSS

In this lab, we will use SPSS to calculate a variety of descriptive statistics. SPSS allows us to specify which statistics we would like calculated and produce them all in one output table. Here, we will learn to:

1. Calculate descriptive statistics
2. Graph data using a histogram
3. Editing graphs

Let's begin with a short data set $\{x = 1, 1, 4, 1, 2, 5, 7\}$

Suppose we want to calculate the measures of central tendency (mean, median, and mode) as well as variability (range, standard deviation, and variance). First,

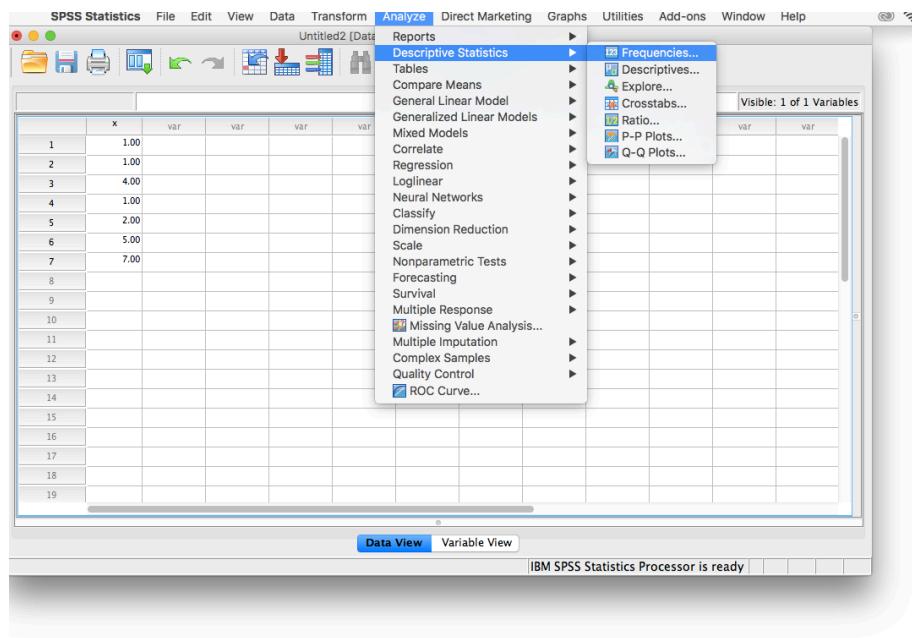
we will have to enter our data into the SPSS spreadsheet. There are 7 measurements, so we will need 7 rows of data (I have also changed the name of this variable in Variable View to **x**:

The screenshot shows the IBM SPSS Statistics Data Editor window titled "Untitled2 [DataSet3] - IBM SPSS Statistics Data Editor". The window has a toolbar at the top with various icons. Below the toolbar is a menu bar. The main area is a data grid with 19 rows and 12 columns. The first column is labeled "x" and contains numerical values: 1.00, 1.00, 4.00, 1.00, 2.00, 5.00, and 7.00. The other columns are labeled "var" and are mostly empty. A status bar at the bottom indicates "Visible: 1 of 1 Variables". The bottom of the window shows tabs for "Data View" and "Variable View", with "Data View" being selected. A message at the bottom right says "IBM SPSS Statistics Processor is ready".

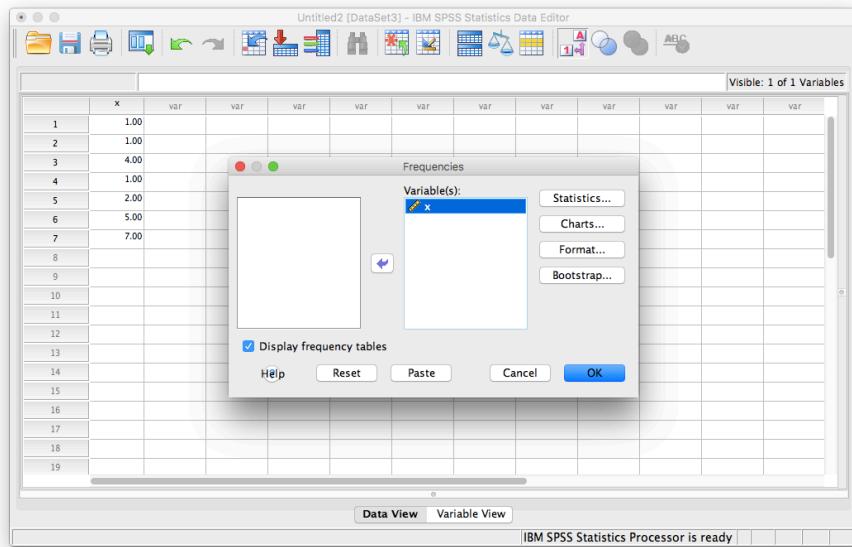
	x	var									
1	1.00										
2	1.00										
3	4.00										
4	1.00										
5	2.00										
6	5.00										
7	7.00										
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											

2.4.1 Calculating Descriptive Statistics

From here, go to the top menu and choose Analyze, then Descriptive Statistics and then Frequencies:

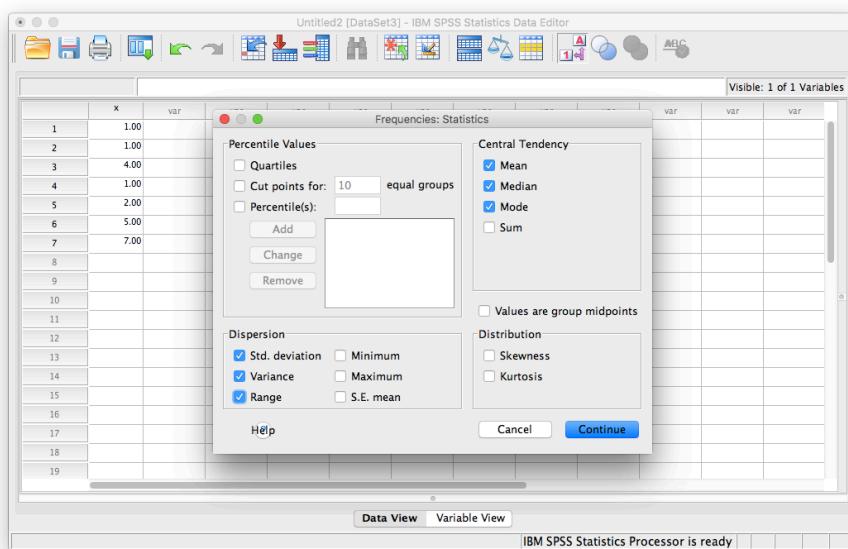


A new window will ask you to specify for which variables you want descriptives statistics calculated. Use the arrow to move the **x** variable from the left-hand to the right-hand field.



Now, click the Statistics button. This will open a new window containing a list of statistics. You can choose as many as you want to be calculated. We will

choose mean, median mode, range, standard deviation, and variance.



Then click Continue, and then OK. SPSS will produce a set of two output tables: one containing the descriptive statistics we have chosen, and the other a frequency table—a list of all the possible values in our data set and their corresponding frequencies.

Statistics

x	
N	Valid 7
	Missing 0
Mean	3.0000
Median	2.0000
Mode	1.00
Std. Deviation	2.38048
Variance	5.667
Range	6.00

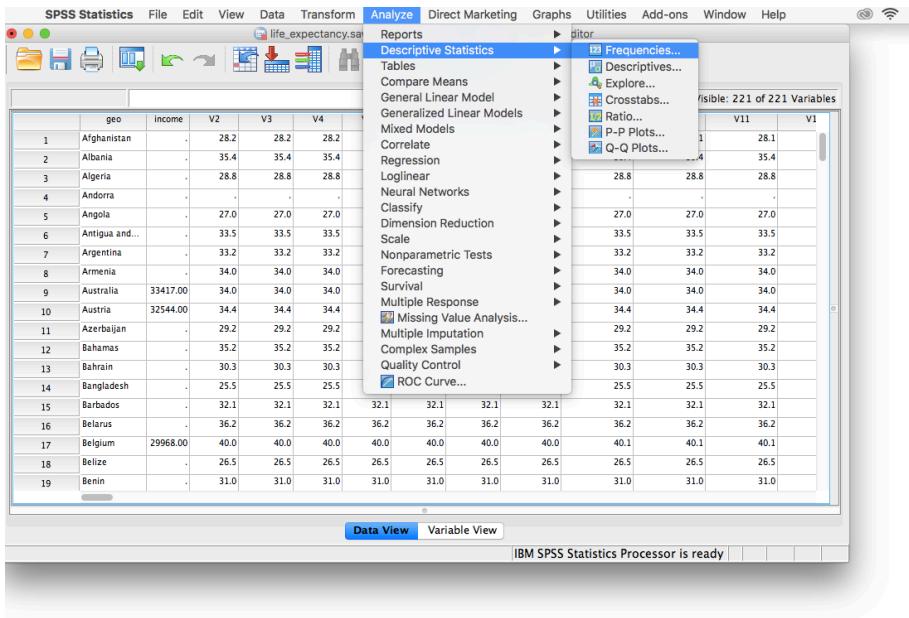
x		Frequency	Percent	Valid Percent	Cumulative Percent
Valid 1.00		3	42.9	42.9	42.9
	2.00	1	14.3	14.3	57.1
	4.00	1	14.3	14.3	71.4
	5.00	1	14.3	14.3	85.7
	7.00	1	14.3	14.3	100.0
	Total	7	100.0	100.0	

2.4.2 Descriptive Statistics and Histograms

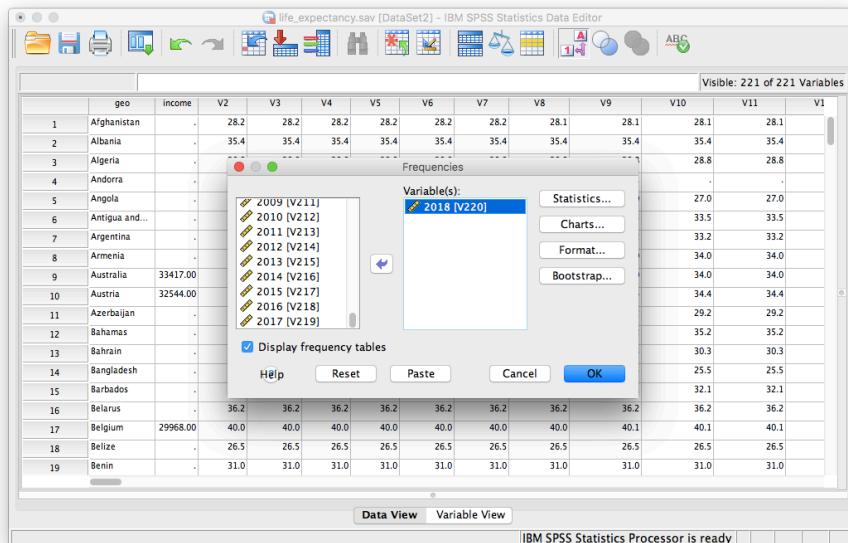
Now let's use a real dataset to calculate the same measures of central tendency and variability as in the last example, but with the addition of a histogram to visualize a distribution to relate back to the descriptive statistics. Here is a link to the life expectancy dataset we used for our graphing tutorial. It is named [life_expectancy.sav](#).

Suppose we wanted to know about life expectancy (around the world) in 2018. This will include calculating descriptive statistics, as well as graphing a histogram to examine the distribution of our data. SPSS often has neat shortcuts to graph data within other menu options. For example, in the process of asking SPSS to produce a table of descriptive statistics, we can also add a histogram to the output without having to go to the Graphs menu.

First, we go to Analyze, then Descriptive Statistics, then Frequencies.

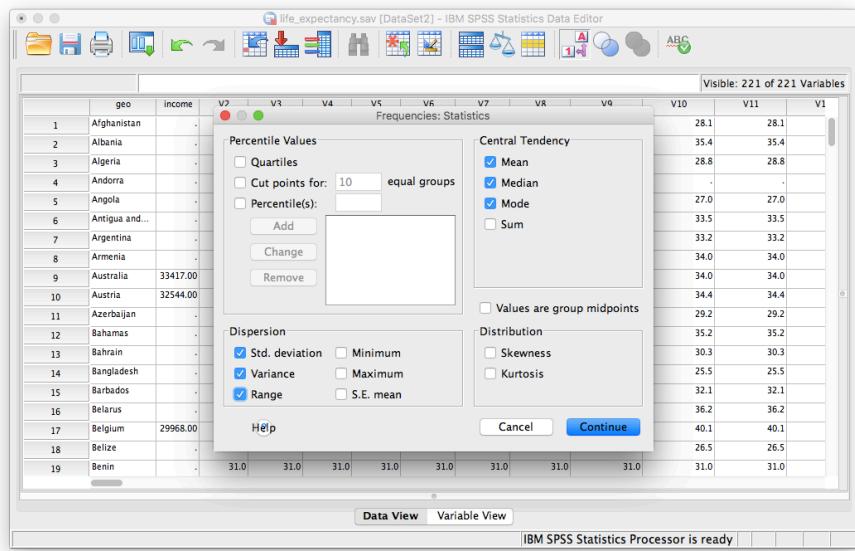


A window will appear asking us to indicate which variable to use. We will scroll all the way down in the list of variables on the left, choose 2018 [V220], and then move it into the field on the right using the arrow.

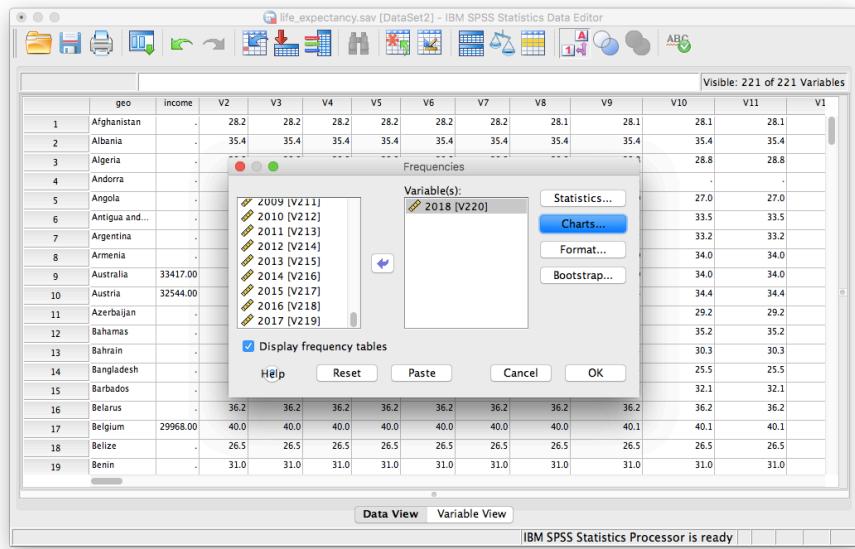


As in the previous example, we will click the Statistics button next, and choose

our measures of central tendency (mean, median, mode) as well as variability (range, standard deviation, variance).

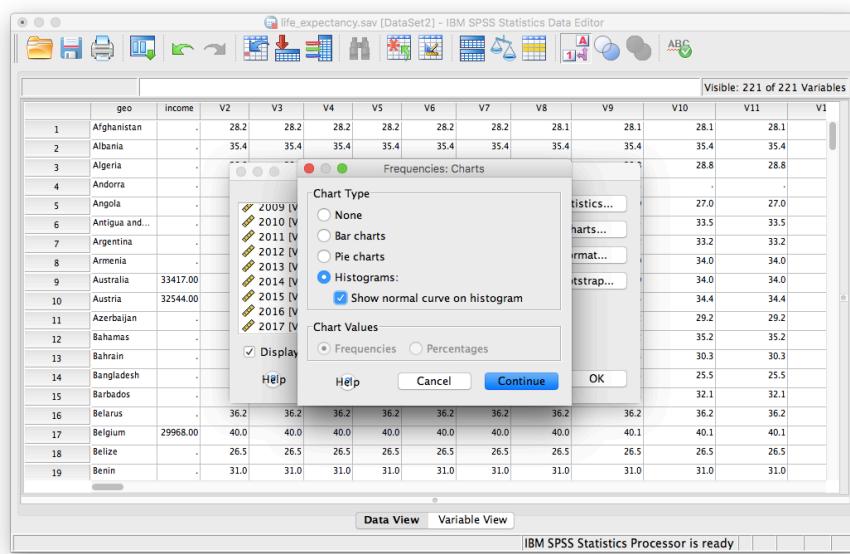


Once we click continue, we are back to the previous window. Here, we can conveniently ask SPSS to insert a histogram. Let's click on the Charts button:



This will open a new window that asks us to specify the type of chart we would

like. Let's select Histogram, and check off Show normal curve on histogram:



Click Continue, and then OK.

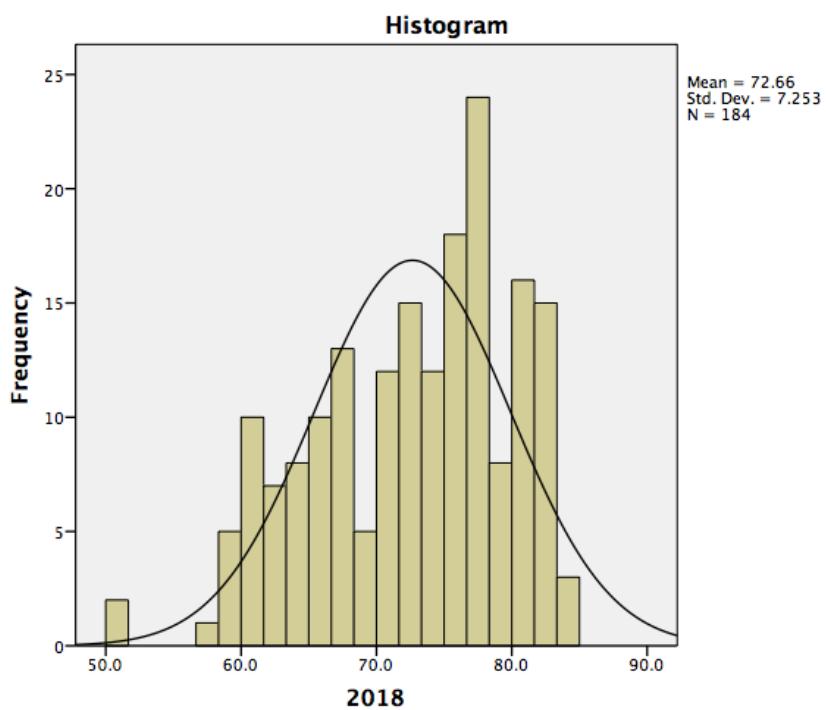
SPSS will produce a table of the requested descriptive statistics, a frequency table (we will ignore this for now because it is long and not relevant to this exercise), and finally, a histogram showing the distribution of life expectancy (in years) with a normal distribution superimposed for comparison.

Statistics

2018

N	Valid	184
	Missing	3
Mean		72.658
Median		74.050
Mode		81.3^a
Std. Deviation		7.2528
Variance		52.603
Range		33.1

a. Multiple modes
exist. The smallest
value is shown

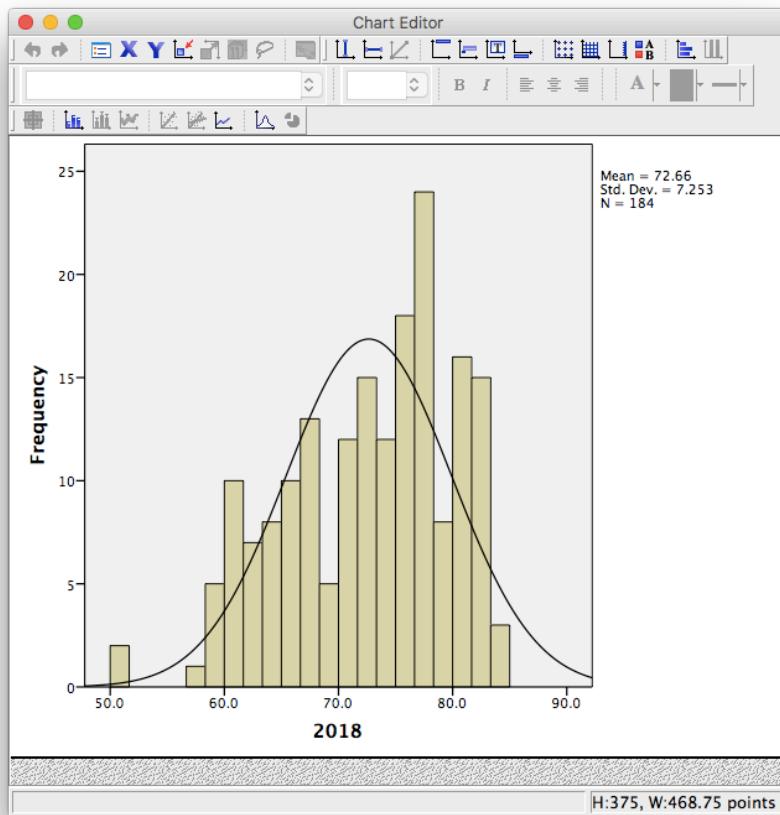


Something to think about: What do the mean, median, and mode indicate about the shape of the distribution? Is it confirmed when we look at the histogram? How does the shape of this distribution compare to the symmetrical normal distribution which is superimposed over it?

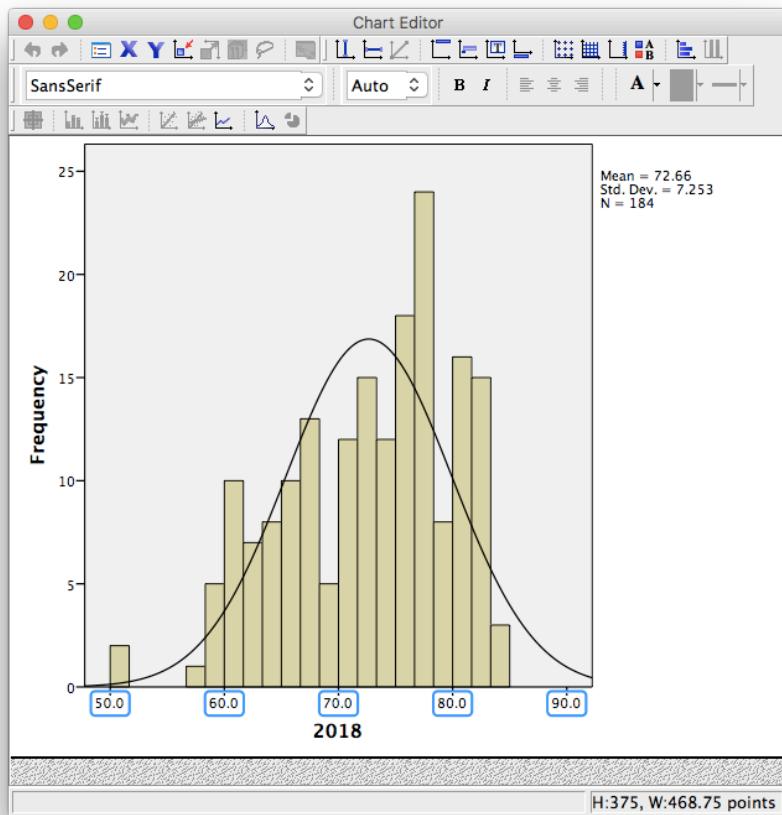
2.4.3 Editing graphs

For future reference, we will take this opportunity to make edits to the appearance of this graph. It is helpful to know how to manipulate not just the superficial, cosmetic appearance of data visualizations, but also the components of graphs that communicate important information. One feature to consider is the scale of our axes.

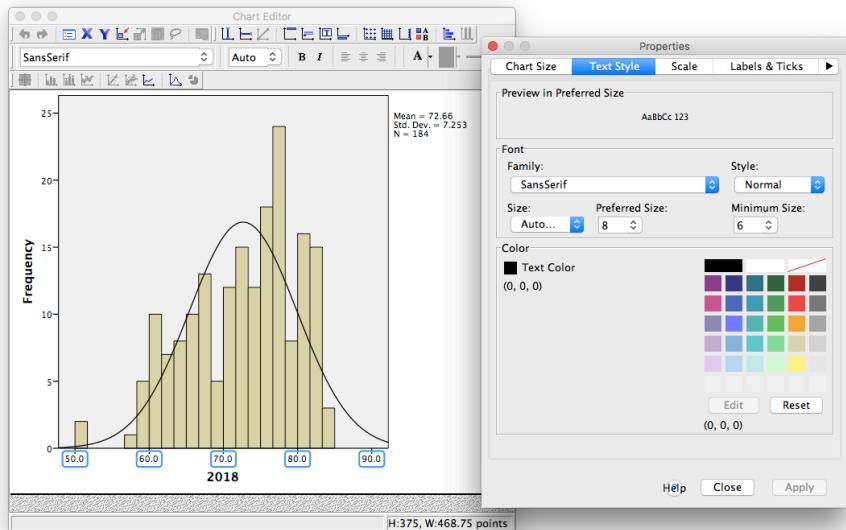
To modify the scale in a graph, first double-click on the graph itself in the output window. A Chart Editor should pop up.



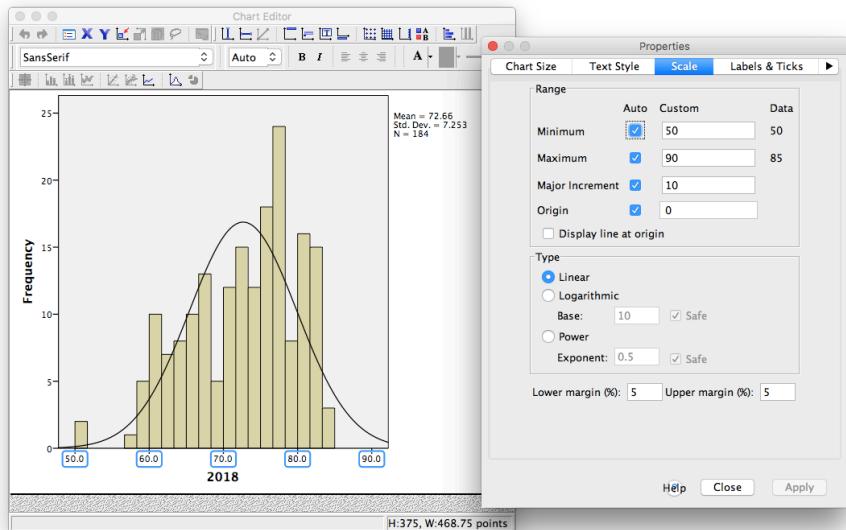
Now, hover your mouse over one of the values on the x-axis, and click down once. You will see all the x-axis values suddenly highlight:



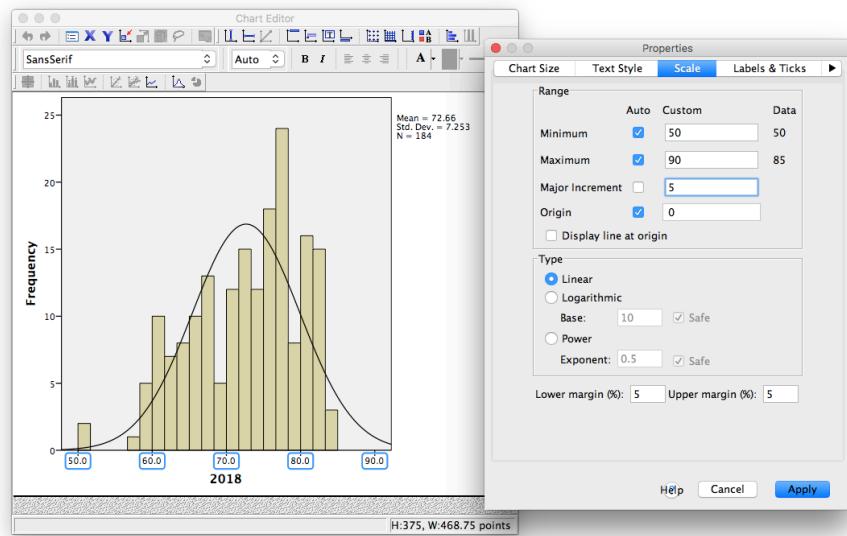
Now that you know how to select the entire axis, double click on it. A properties window will pop up:



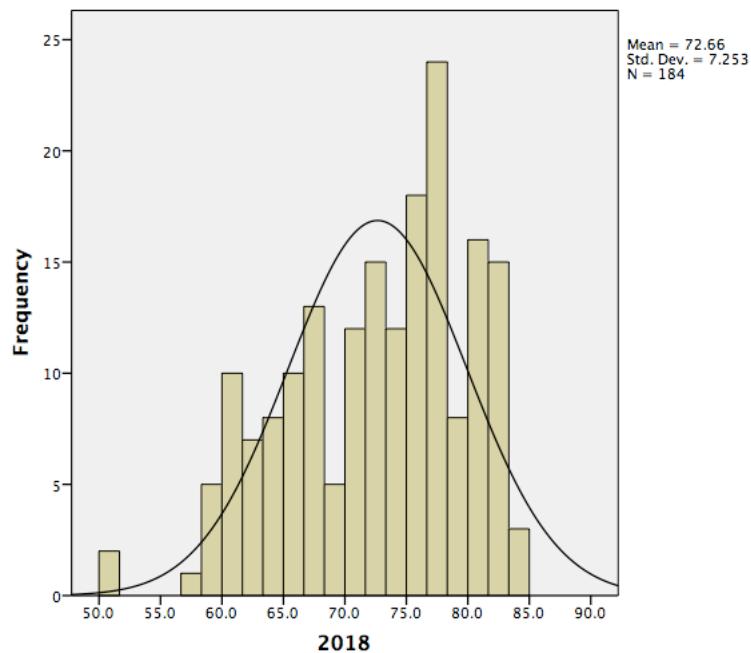
Notice that in the default tab, you can change the fill color of components of your graph. For now, we will select the Scale tab instead:



Notice in this tab, we can change the minimum, maximum, and increments of our scale. Lets change the major increment to 5:



Click Apply. Then Close. Now, close your Chart Editor window by simply x-ing it out. The SPSS output window will now contain your edited graph:

Graph**2.4.4 Practice Problems**

-
1. Using the life expectancy data set, produce a table of output showing the descriptive statistics (measures of central tendency and variability) for both years 1800 and 1934 (during the Great Depression).
 2. Plot histograms of life expectancy for both years. How are these distributions different? (Hint: Plot these on the same axes so that they are comparable).

2.5 JAMOVI

How to do it in JAMOVI

Chapter 3

Lab 3: Correlation

If ... we choose a group of social phenomena with no antecedent knowledge of the causation or absence of causation among them, then the calculation of correlation coefficients, total or partial, will not advance us a step toward evaluating the importance of the causes at work. —Sir Ronald Fisher

In lecture and in the textbook, we have been discussing the idea of correlation. This is the idea that two things that we measure can be somehow related to one another. For example, your personal happiness, which we could try to measure say with a questionnaire, might be related to other things in your life that we could also measure, such as number of close friends, yearly salary, how much chocolate you have in your bedroom, or how many times you have said the word Nintendo in your life. Some of the relationships that we can measure are meaningful, and might reflect a causal relationship where one thing causes a change in another thing. Some of the relationships are spurious, and do not reflect a causal relationship.

In this lab you will learn how to compute correlations between two variables in software, and then ask some questions about the correlations that you observe.

3.1 General Goals

1. Compute Pearson's r between two variables using software
2. Discuss the possible meaning of correlations that you observe

3.1.1 Important Info

We use data from the World Happiness Report. A .csv of the data can be found here: WHR2018.csv

3.2 R

In this lab we use explore to explore correlations between any two variables, and also show how to do a regression line. There will be three main parts. Getting R to compute the correlation, and looking at the data using scatter plots. We'll look at some correlations from the World Happiness Report. Then you'll look at correlations using data we collect from ourselves. It will be fun.

3.2.1 cor for correlation

R has the `cor` function for computing Pearson's r between any two variables. In fact this same function computes other versions of correlation, but we'll skip those here. To use the function you just need two variables with numbers in them like this:

```
x <- c(1,3,2,5,4,6,5,8,9)
y <- c(6,5,8,7,9,7,8,10,13)
cor(x,y)
```

```
## [1] 0.76539
```

Well, that was easy.

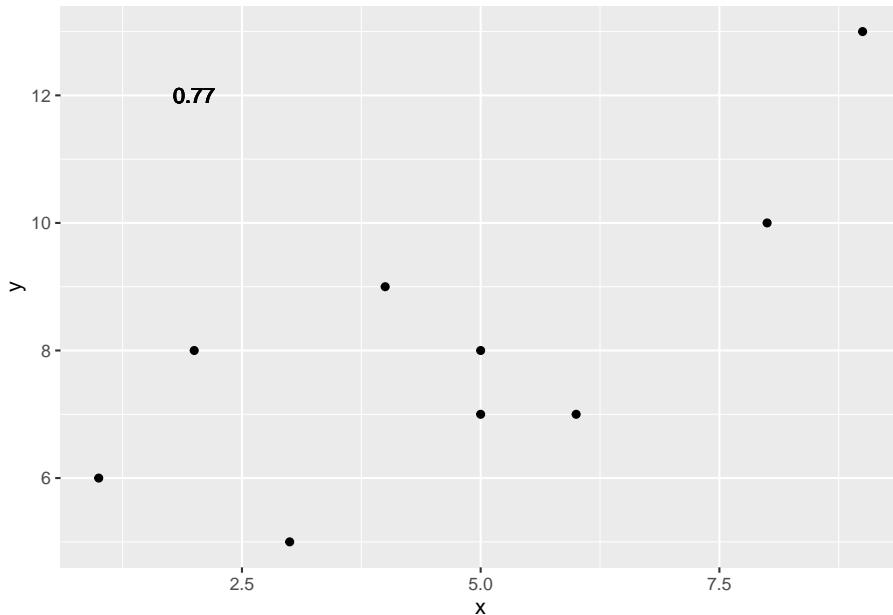
3.2.1.1 scatterplots

Let's take our silly example, and plot the data in a scatter plot using ggplot2, and let's also return the correlation and print it on the scatter plot. Remember, ggplot2 wants the data in a data.frame, so we first put our x and y variables in a data frame.

```
library(ggplot2)

# create data frame for plotting
my_df <- data.frame(x,y)

# plot it
ggplot(my_df, aes(x=x,y=y))+
  geom_point()+
  geom_text(aes(label = round(cor(x,y), digits=2), y=12, x=2))
```



Wow, we're moving fast here.

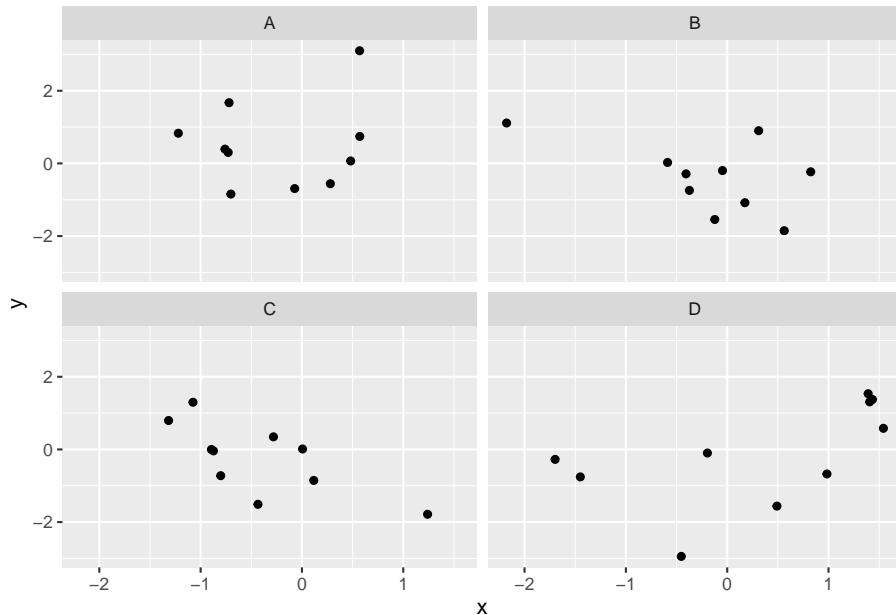
3.2.1.2 lots of scatterplots

Before we move on to real data, let's look at some fake data first. Often we will have many measures of X and Y, split between a few different conditions, for example, A, B, C, and D. Let's make some fake data for X and Y, for each condition A, B, C, and D, and then use facet_wrapping to look at four scatter plots all at once

```
x<-rnorm(40,0,1)
y<-rnorm(40,0,1)
conditions<-rep(c("A","B","C","D"), each=10)

all_df <- data.frame(conditions, x, y)

ggplot(all_df, aes(x=x,y=y))+
  geom_point()+
  facet_wrap(~conditions)
```



3.2.1.3 computing the correlations all at once

We've seen how we can make four graphs at once. `Facet_wrap` will always try to make as many graphs as there are individual conditions in the column variable. In this case there are four, so it makes four.

Notice, the scatter plots don't show the correlation (r) values. Getting these numbers on there is possible, but we have to calculate them first. We'll leave it to you to Google how to do this, if it's something you want to do. Instead, what we will do is make a table of the correlations in addition to the scatter plot. We again use `dplyr` to do this:

OK, we are basically ready to turn to some real data and ask if there are correlations between interesting variables... You will find that there are some... But before we do that, we do one more thing. This will help you become a little bit more skeptical of these "correlations".

3.2.1.4 Chance correlations

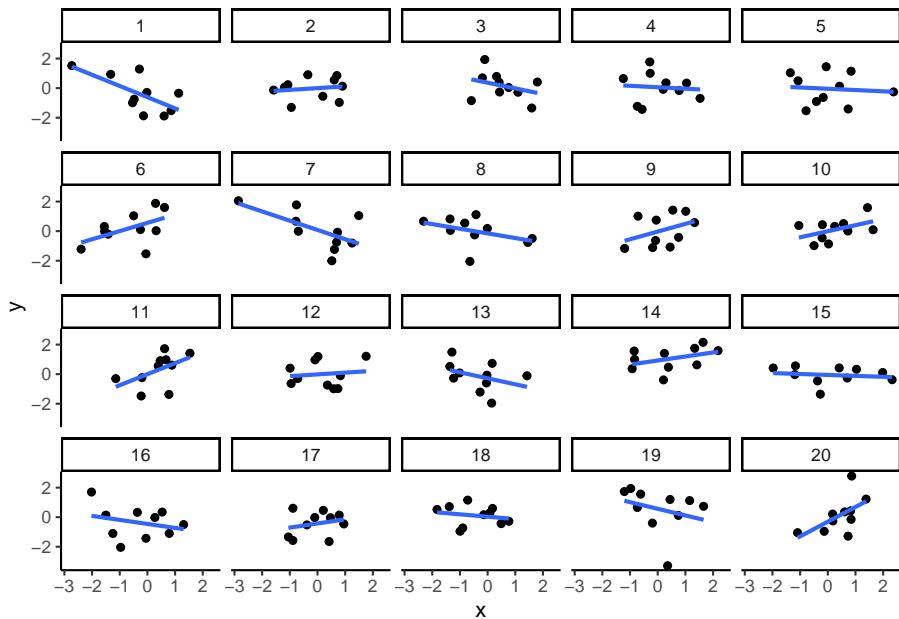
As you learned from the textbook. We can find correlations by chance alone, even when there is no true correlation between the variables. For example, if we sampled randomly into x , and then sampled some numbers randomly into y . We know they aren't related, because we randomly sampled the numbers. However, doing this creates some correlations some of the time just by chance. You can demonstrate this to yourself with the following code. It's a repeat of what we already saw, just with a few more conditions added. Let's look at 20

conditions, with random numbers for x and y in each. For each, sample size will be 10. We'll make the fake data, then make a big graph to look at all. And, even though we get to regression later in the lab, I'll put the best fit line onto each scatter plot, so you can "see the correlations".

```
x<-rnorm(10*20,0,1)
y<-rnorm(10*20,0,1)
conditions<-rep(1:20, each=10)

all_df <- data.frame(conditions, x, y)

ggplot(all_df, aes(x=x,y=y))+
  geom_point()+
  geom_smooth(method=lm, se=FALSE)+
  facet_wrap(~conditions)+
  theme_classic()
```



You can see that the slope of the blue line is not always flat. Sometimes it looks like there is a correlation, when we know there shouldn't be. You can keep re-doing this graph, by re-knitting your R Markdown document, or by pressing the little green play button. This is basically you simulating the outcomes as many times as you press the button.

The point is, now you know you can find correlations by chance. So, in the next section, you should always wonder if the correlations you find reflect meaningful association between the x and y variable, or could have just occurred by chance.

3.2.2 World Happiness Report

Let's take a look at some correlations in real data. We are going to look at responses to a questionnaire about happiness that was sent around the world, from the world happiness report

3.2.2.1 Load the data

We load the data into a data frame. Reminder, the following assumes that you have downloaded the RMarkdownsLab.zip file which contains the data file in the data folder.

```
library(data.table)
whr_data <- fread('data/WHR2018.csv')
```

You can also load the data using the following URL

```
library(data.table)
whr_data <- fread("https://raw.githubusercontent.com/CrumpLab/statisticsLab/master/data/WHR2018.csv")
```

3.2.2.2 Look at the data

```
library(summarytools)
view(dfSummary(whr_data))
```

You should be able to see that there is data for many different countries, across a few different years. There are lots of different kinds of measures, and each are given a name. I'll show you some examples of asking questions about correlations with this data, then you get to ask and answer your own questions.

3.2.2.3 My Question #1

For the year 2017 only, does a countries measure for "freedom to make life choices" correlate with that countries measure for "Confidence in national government"?

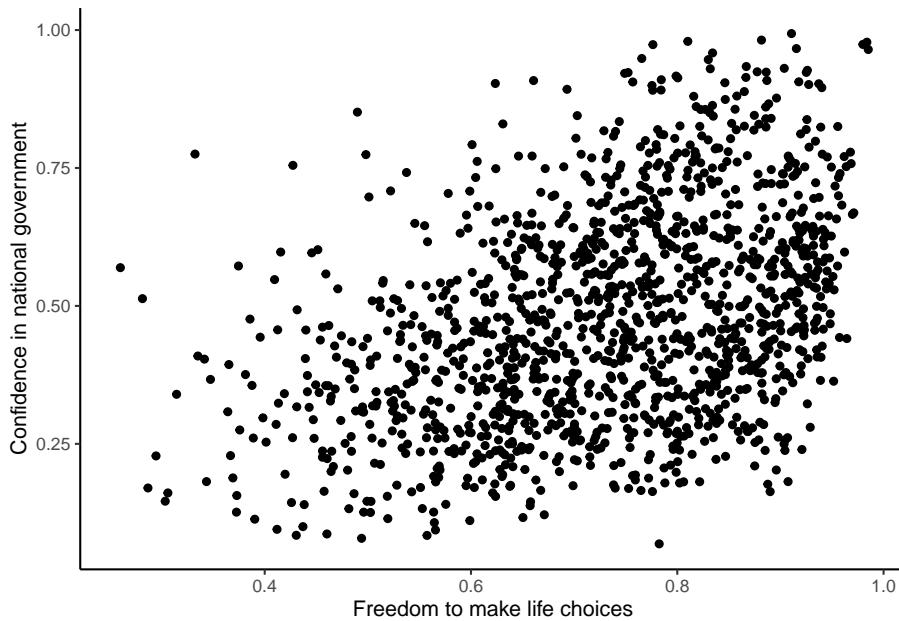
Let's find out. We calculate the correlation, and then we make the scatter plot.

```
cor(whr_data$`Freedom to make life choices`,
    whr_data$`Confidence in national government`)

## [1] NA

ggplot(whr_data, aes(x=`Freedom to make life choices`,
    y=`Confidence in national government`))+
```

geom_point()+
 theme_classic()



Interesting, what happened here? We can see some dots, but the correlation was NA (meaning undefined). This occurred because there are some missing data points in the data. We can remove all the rows with missing data first, then do the correlation. We will do this a couple steps, first creating our own data.frame with only the numbers we want to analyse. We can select the columns we want to keep using `select`. Then we use `filter` to remove the rows with NAs.

```
library(dplyr)

smaller_df <- whr_data %>%
  select(country,
         `Freedom to make life choices`,
         `Confidence in national government`) %>%
  filter(!is.na(`Freedom to make life choices`),
         !is.na(`Confidence in national government`))

cor(smaller_df$`Freedom to make life choices`,
     smaller_df$`Confidence in national government`)

## [1] 0.4080963
```

Now we see the correlation is .408.

Although the scatter plot shows the dots are everywhere, it generally shows that as Freedom to make life choices increases in a country, that countries confidence in their national government also increase. This is a positive correlation. Let's do this again and add the best fit line, so the trend is more clear, we use

`geom_smooth(method=lm, se=FALSE)`. I also change the alpha value of the dots so they blend it bit, and you can see more of them.

```
# select DVs and filter for NAs

smaller_df <- whr_data %>%
  select(country,
         `Freedom to make life choices`,
         `Confidence in national government`) %>%
  filter(!is.na(`Freedom to make life choices`),
         !is.na(`Confidence in national government`))

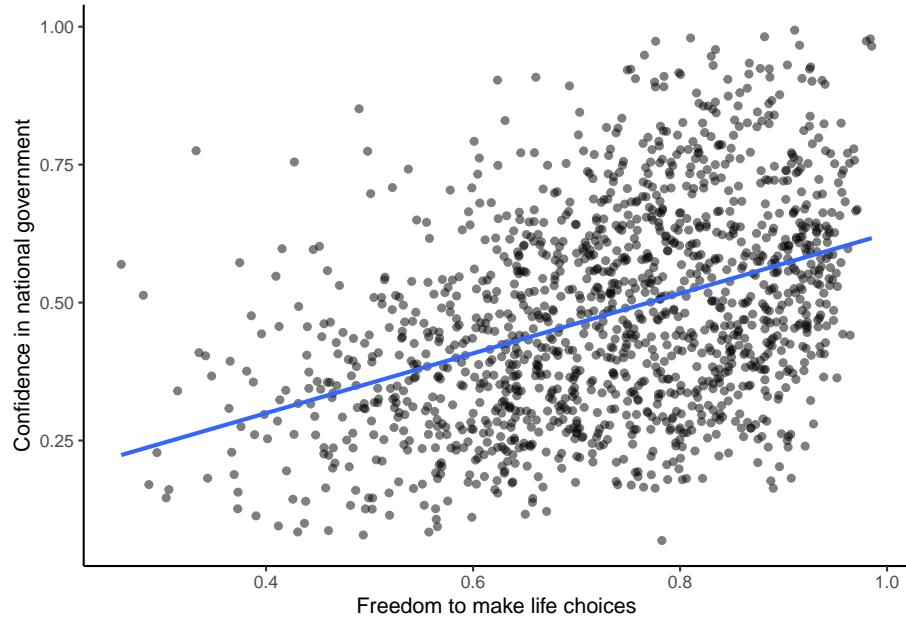
# calcualte correlation

cor(smaller_df$`Freedom to make life choices`,
     smaller_df$`Confidence in national government`)

## [1] 0.4080963

# plot the data with best fit line

ggplot(smaller_df, aes(x=`Freedom to make life choices`,
                        y=`Confidence in national government`)) +
  geom_point(alpha=.5) +
  geom_smooth(method=lm, se=FALSE) +
  theme_classic()
```



3.2.2.4 My Question #2

After all that work, we can now speedily answer more questions. For example, what is the relationship between positive affect in a country and negative affect in a country. I wouldn't be surprised if there was a negative correlation here: when positive feelings generally go up, shouldn't negative feelings generally go down?

To answer this question, we just copy paste the last code block, and change the DVs to be Positive affect, and Negative affect

```
# select DVs and filter for NAs

smaller_df <- whr_data %>%
  select(country,
         `Positive affect`,
         `Negative affect`) %>%
  filter(!is.na(`Positive affect`),
         !is.na(`Negative affect`))

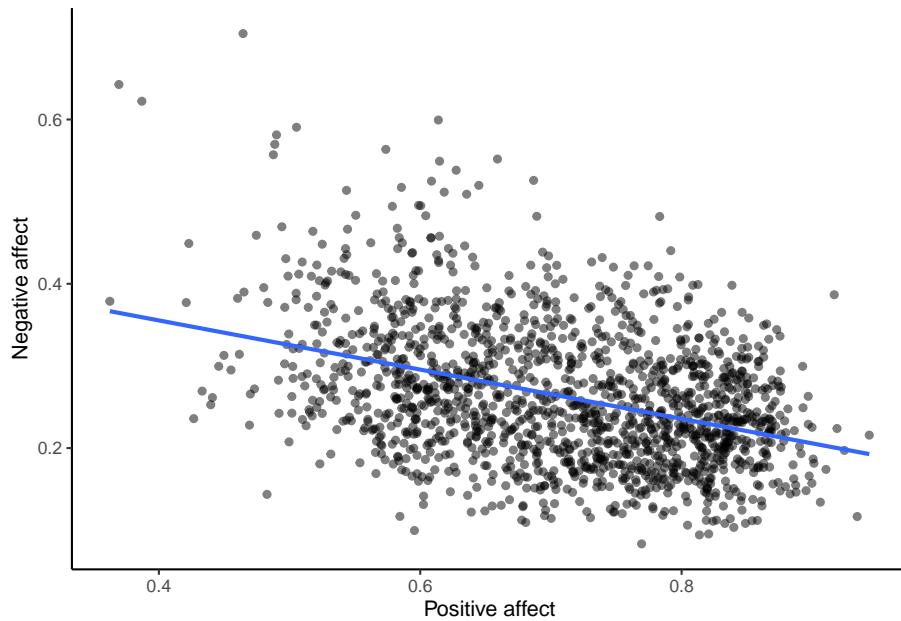
# calcualte correlation

cor(smaller_df$`Positive affect`,
     smaller_df$`Negative affect`)
```

```
## [1] -0.3841123

# plot the data with best fit line

ggplot(smaller_df, aes(x=`Positive affect`,
                        y=`Negative affect`)) +
  geom_point(alpha=.5) +
  geom_smooth(method=lm, se=FALSE) +
  theme_classic()
```



Bam, there we have it. As positive affect goes up, negative affect goes down. A negative correlation.

3.2.3 Generalization Exercise

This generalization exercise will explore the idea that correlations between two measures can arise by chance alone. There are two questions to answer. For each question you will be sampling random numbers from uniform distribution. To conduct the estimate, you will be running a simulation 100 times. The questions are:

1. Estimate the range (minimum and maximum) of correlations (using pearson's r) that could occur by chance between two variables with $n=10$.
2. Estimate the range (minimum and maximum) of correlations (using pearson's r) that could occur by chance between two variables with $n = 100$.

Use these tips to answer the question.

Tip 1: You can use the `runif()` function to sample random numbers between a minimum value, and maximum value. The example below sample 10 ($n=10$) random numbers between the range 0 (min = 0) and 10 (max=10). Everytime you run this code, the 10 values in `x` will be re-sampled, and will be 10 new random numbers

```
x <- runif(n=10, min=0, max=10)
```

Tip 2: You can compute the correlation between two sets of random numbers, by first sampling random numbers into each variable, and then running the `cor()` function.

```
x <- runif(n=10, min=0, max=10)
y <- runif(n=10, min=0, max=10)
cor(x,y)
```

```
## [1] 0.3124014
```

Running the above code will give different values for the correlation each time, because the numbers in `x` and `y` are always randomly different. We might expect that because `x` and `y` are chosen randomly that there should be a 0 correlation. However, what we see is that random sampling can produce “fake” correlations just by chance alone. We want to estimate the range of correlations that chance can produce.

Tip 3: One way to estimate the range of correlations that chance can produce is to repeat the above code many times. For example, if you ran the above code 100 times, you could save the correlations each time, then look at the smallest and largest correlation. This would be an estimate of the range of correlations that can be produced by chance. How can you repeat the above code many times to solve this problem?

We can do this using a `for` loop. The code below shows how to repeat everything inside the `for` loop 100 times. The variable `i` is an index, that goes from 1 to 100. The `saved_value` variable starts out as an empty variable, and then we put a value into it (at index position `i`, from 1 to 100). In this code, we put the sum of the products of `x` and `y` into the `saved_value` variable. At the end of the simulation, the `saved_value` variable contains 100 numbers. The `min()` and `max()` functions are used to find the minimum and maximum values for each of the 100 simulations. You should be able to modify this code by replacing `sum(x*y)` with `cor(x,y)`. Doing this will allow you to run the simulation 100 times, and find the minimum correlation and maximum correlation that arises by chance. This will be estimate for question 1. To provide an estimate for question 2, you will need to change `n=10` to `n=100`.

```
saved_value <- c() #make an empty variable
for (i in 1:100){
  x <- runif(n=10, min=0, max=10)
  y <- runif(n=10, min=0, max=10)
  saved_value[i] <- sum(x*y)
}

min(saved_value)

## [1] 100.2798
```

```
max(saved_value)  
## [1] 399.6434
```

3.2.4 Writing assignment

Answer the following questions with complete sentences. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

1. Imagine a researcher found a positive correlation between two variables, and reported that the r value was $.3$. One possibility is that there is a true correlation between these two variables. Discuss one alternative possibility that would also explain the observation of $.3$ value between the variables.
2. Explain the difference between a correlation of $r = .3$ and $r = .7$. What does a larger value of r represent?
3. Explain the difference between a correlation of $r = .5$, and $r = -.5$.

3.3 Excel

How to do it in Excel

3.4 SPSS

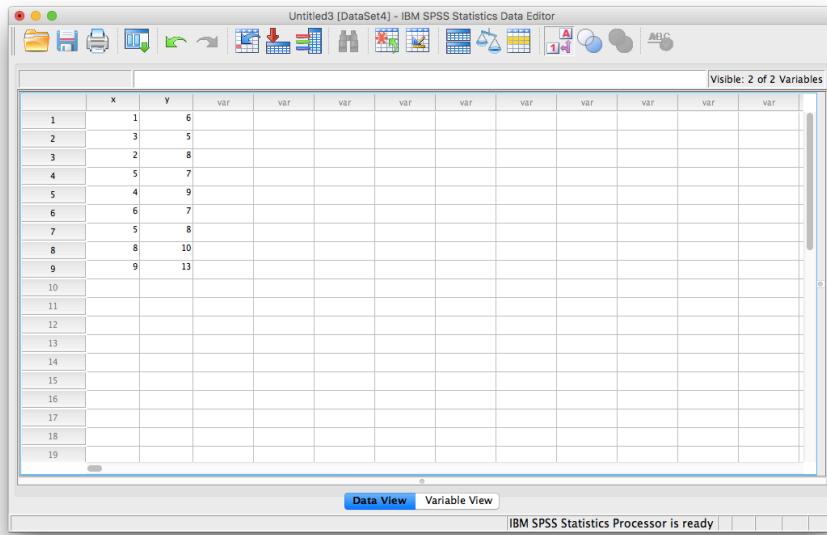
In this lab, we will use SPSS to calculate the correlation coefficient. We will focus on the most-commonly used Pearson's coefficient, r . We will learn how to:

1. Calculate the Pearson's r correlation coefficient for bivariate data
2. Produce a correlation matrix, reporting Pearson's r for more than two variables at a time
3. Produce a scatterplot
4. Split a data file for further analysis

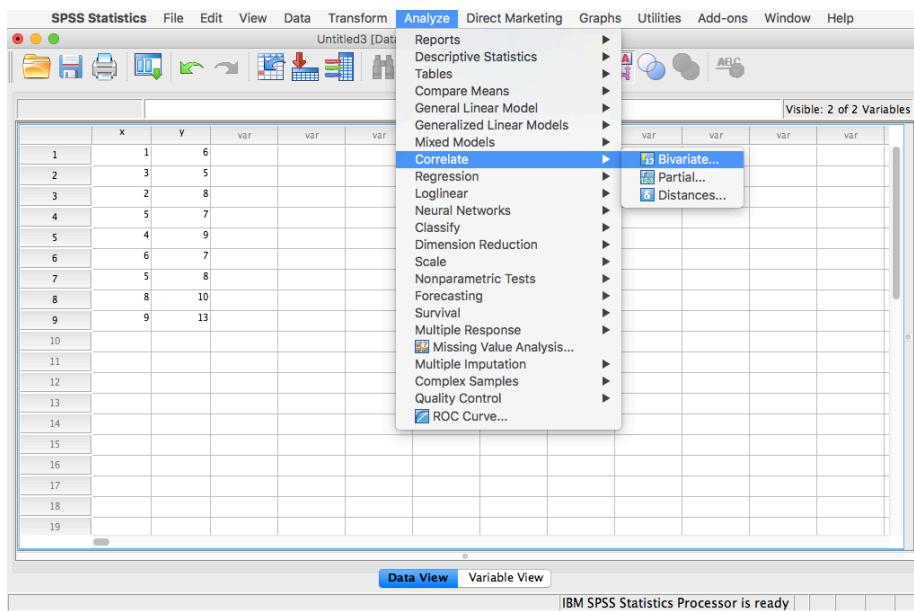
Let's first begin with a short data set we will enter into a new SPSS data spreadsheet. Remember, in order to calculate a correlation, you need to have bivariate data; that is, you must have at least two variables, x and y . You can have more than two variables, in which case we can calculate a correlation matrix, as indicated in the section that follows.

3.4.1 Correlation Coefficient for Bivariate Data: Two Variables

Let's use the following data set: $\{x = 1, 3, 2, 5, 4, 6, 5, 8, 9\}$ $\{y = 6, 5, 8, 7, 9, 7, 8, 10, 13\}$. Notice there are two variables, x and y . Enter these into SPSS and name them appropriately.

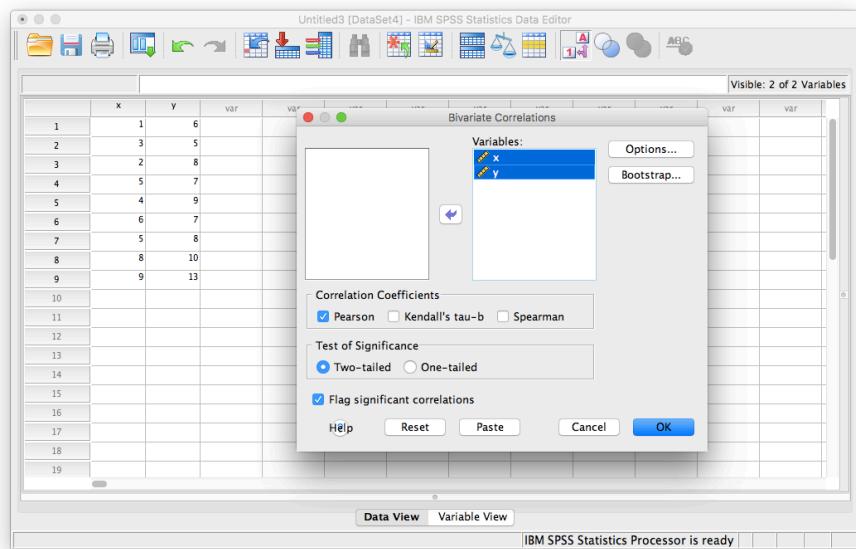


Next, click Analyze, then Correlate, then Bivariate:



The next window will ask you to select variables to correlate. Since we have two

(x and y) move them both from the left-hand field to the right-hand field using the arrow. Notice that in this window, Pearson is selected. This is the default setting (and the one we want), but notice there are other ways to calculate the correlation between variables. We will stick with Pearson's correlation coefficient for this course.



Now, click OK.

SPSS will produce an output table containing the correlation coefficient requested. Notice that the table is redundant; it gives us the correlation between x and y, the correlation between y and x, the correlation between x and itself, and the correlation between y and itself. Any variable correlated with itself will result in an r of 1. The Pearson r correlation between variables x and y is .765.

Correlations

[DataSet4]

Correlations

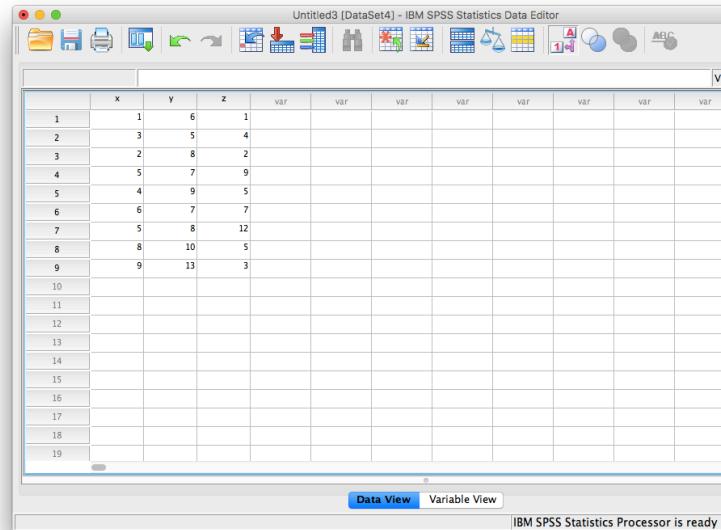
		x	y
x	Pearson Correlation	1	.765*
x	Sig. (2-tailed)		.016
x	N	9	9
y	Pearson Correlation	.765*	1
y	Sig. (2-tailed)	.016	
y	N	9	9

*. Correlation is significant at the 0.05 level
(2-tailed).

3.4.2 Correlation Matrix

In the event that you have more than two variables in your spreadsheet, and would like to evaluate correlations between several variables taken two at a time, you need not re-run the correlations in SPSS repeatedly. You can, in fact, enter multiple variables into the correlation window and obtain a correlation matrix—a table showing every possible bivariate correlation amongst a group of variables.

To illustrate how this is done, let's add a new variable to our existing spreadsheet:

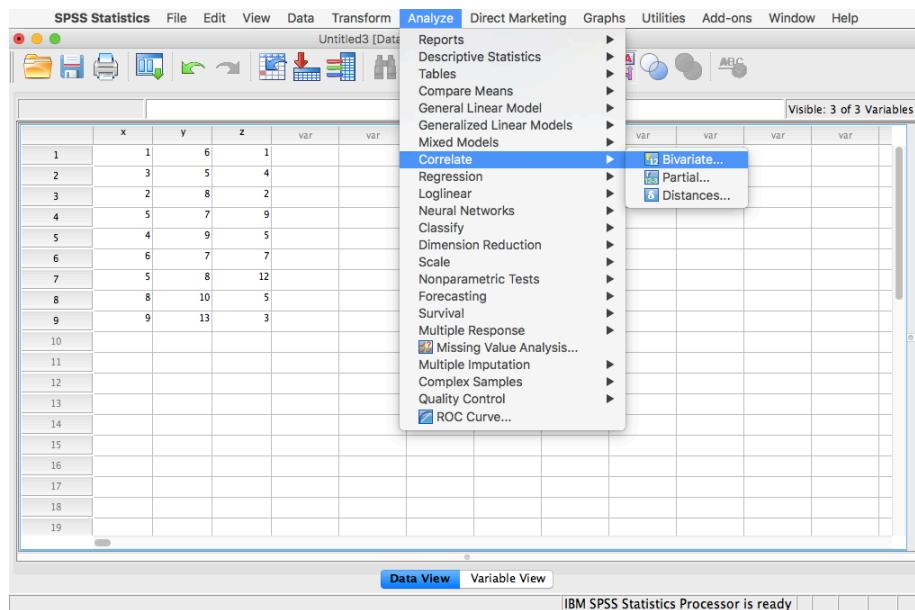


The screenshot shows the SPSS Statistics Data Editor window titled "Untitled3 [DataSet4] - IBM SPSS Statistics Data Editor". The Data View tab is active. The data table contains the following rows:

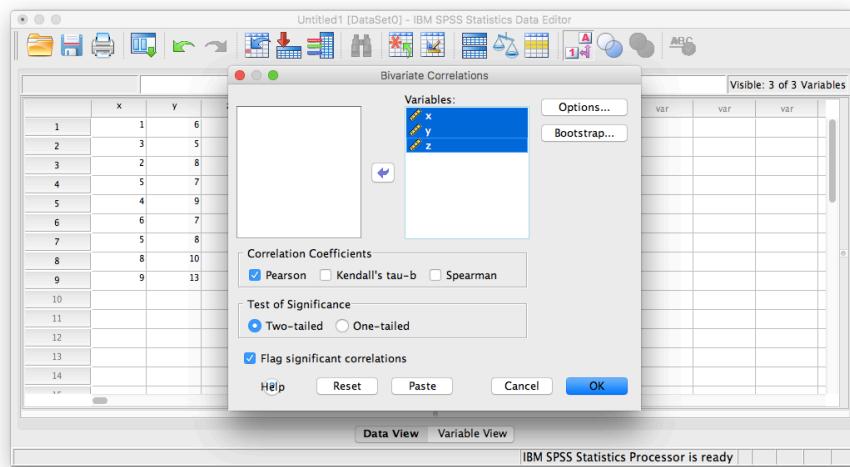
	x	y	z	var															
1	1	6	1																
2	3	5	4																
3	2	8	2																
4	5	7	9																
5	4	9	5																
6	6	7	7																
7	5	8	12																
8	8	10	5																
9	9	13	3																
10																			
11																			
12																			
13																			
14																			
15																			
16																			
17																			
18																			
19																			

variable z, {z= 1, 4, 2, 9, 5, 7, 12, 5, 3}

From here, go to Analyze, then Correlate, then Bivariate:



Next, you will encounter the window that asks you to indicate which variables to correlate. Select all three variables (x, y, and z) and move them to the right-hand field using the arrow.



Click OK. SPSS will produce an output table that contains correlations for every pairing of our three variables, along with the correlations of each variable with itself.

Correlations

[DataSet4]

Correlations

		x	y	z
x	Pearson Correlation	1	.765*	.294
	Sig. (2-tailed)		.016	.443
	N	9	9	9
y	Pearson Correlation	.765*	1	-.080
	Sig. (2-tailed)	.016		.837
	N	9	9	9
z	Pearson Correlation	.294	-.080	1
	Sig. (2-tailed)	.443	.837	
	N	9	9	9

*. Correlation is significant at the 0.05 level (2-tailed).

According to this output:

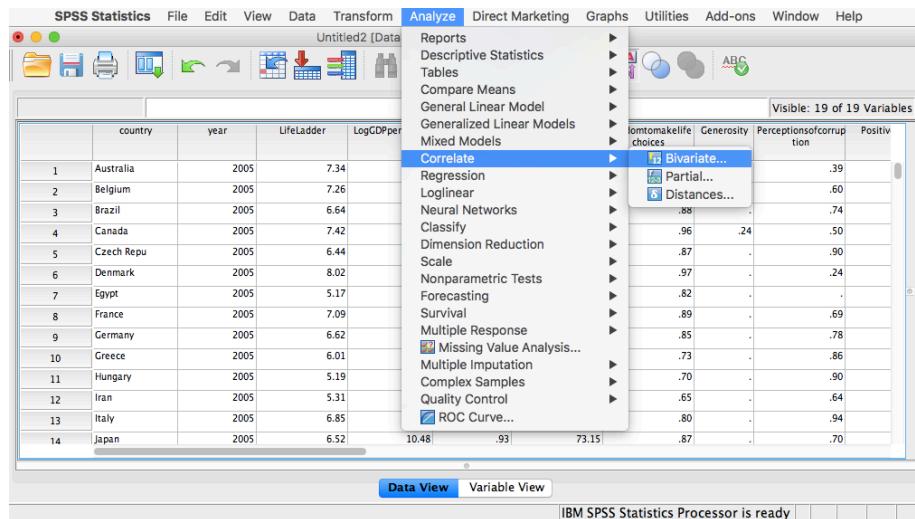
1. The correlation coefficient between variables x and y is .765
2. The correlation coefficient between variables x and z is .294
3. The correlation coefficient between variables y and z is -.080

3.4.3 Correlation and Scatterplots

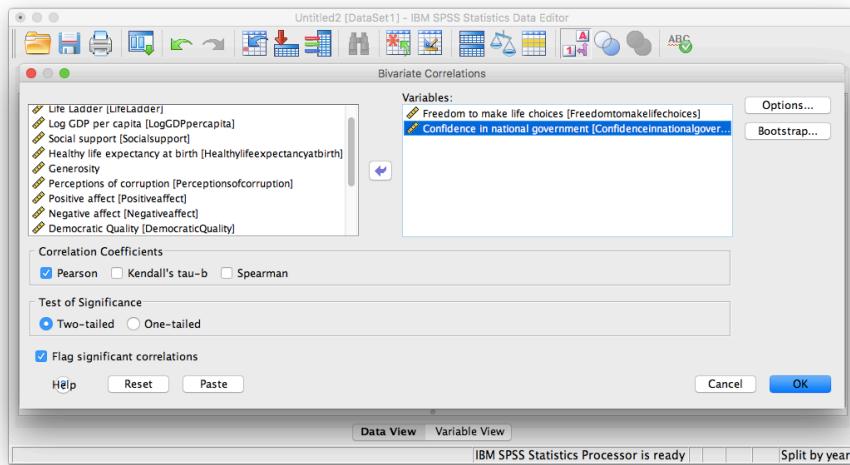
To accompany the calculation of the correlation coefficient, the scatterplot is the relevant visualization tool. Let's use data from The World Happiness Report, a questionnaire about happiness. Here is a link to the file named WHR2018.sav.

Using this data, let's answer the following question: does a country's measure for **freedom to make life choices** correlate with that country's measure for **Confidence in national government**?

Let's find the correlation coefficient between these variables first. Go to Analyze, then Correlate, then Bivariate:



Next, a window will appear asking for the variables to be correlated. Go through the list on the left and find **Freedom to make life choices** as well as **Confidence in national government**. Move both of these variables to the field on the right using the arrow.



Click OK. SPSS will produce a correlation table.

Correlations

[DataSet1]

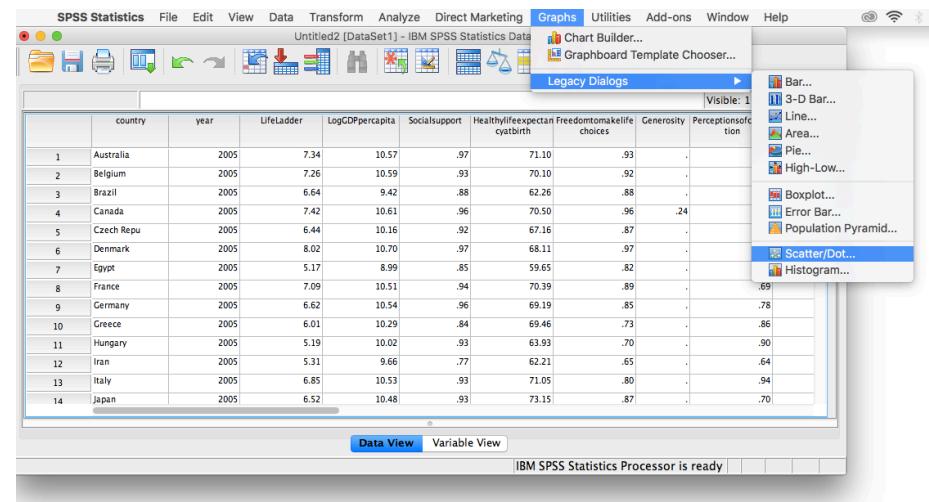
Correlations

		Freedom to make life choices	Confidence in national government
Freedom to make life choices	Pearson Correlation	1	.408**
	Sig. (2-tailed)		.000
	N	1533	1395
Confidence in national government	Pearson Correlation	.408**	1
	Sig. (2-tailed)	.000	
	N	1395	1401

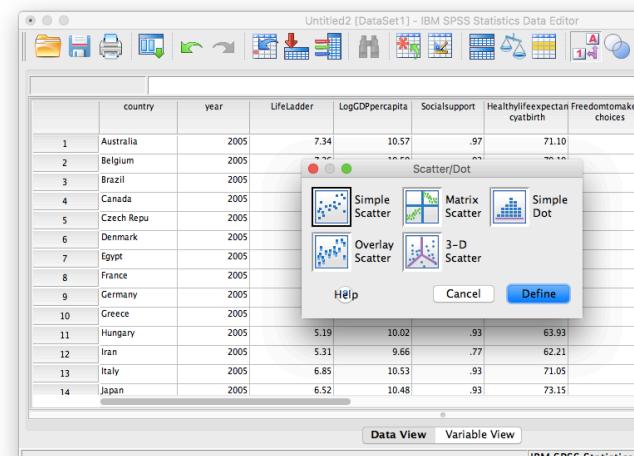
**. Correlation is significant at the 0.01 level (2-tailed).

Based on this output, the correlation between Freedom to make life choices and Confidence in national government is .408.

Let's continue to create the scatterplot for this data. Go to Graphs, then Legacy

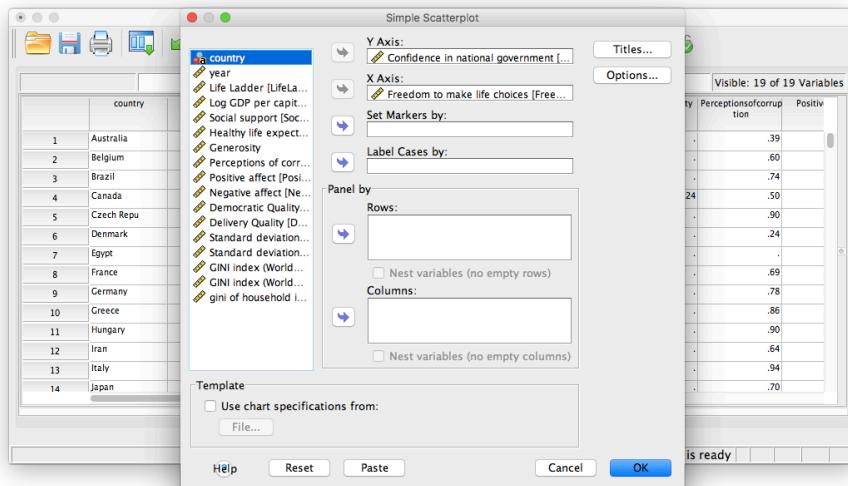


Dialogs, then Scatter...



In the next window, choose Simple, then Define:

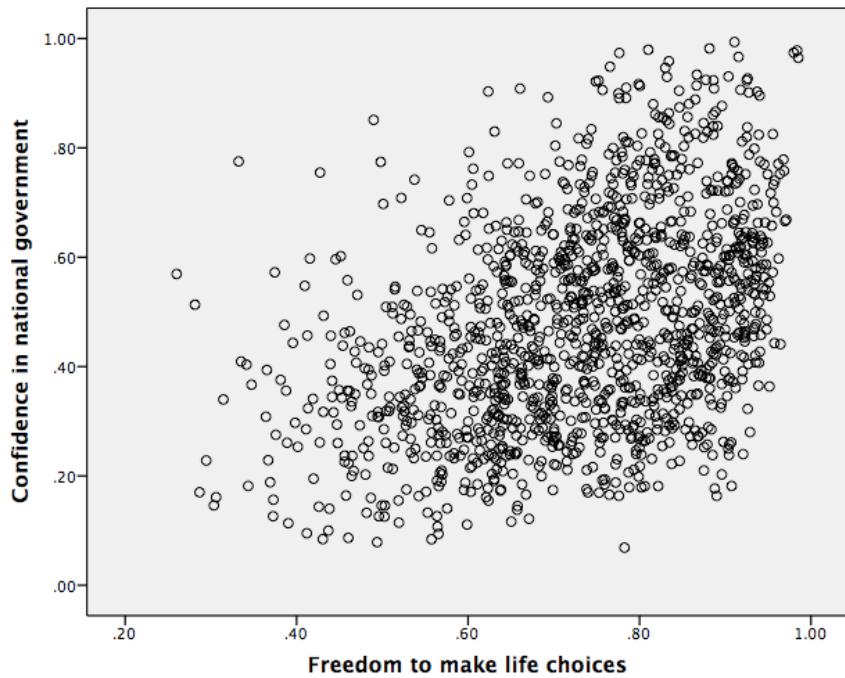
Next, move your two variables (**freedom to make life choices** and **confidence in national government**) into the x-axis and y-axis fields. Again, it does not matter which variable goes where, for now.



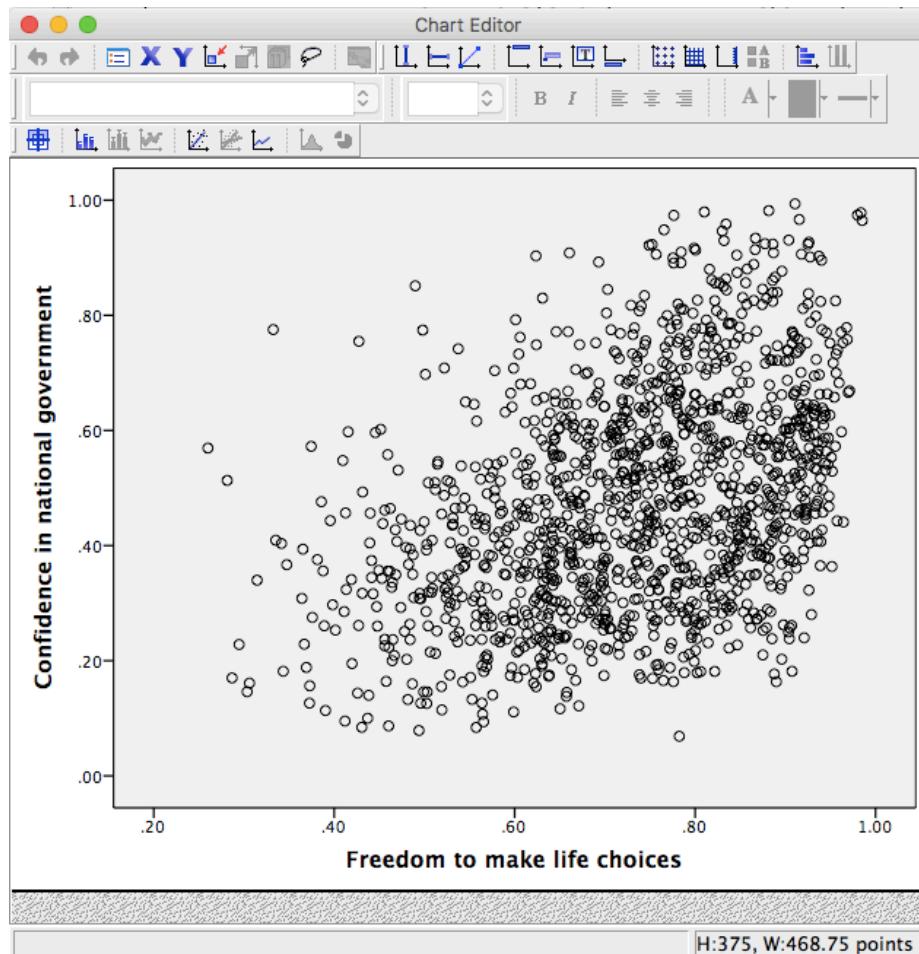
Click OK. SPSS will produce a scatterplot of your data, as follows:

Graph

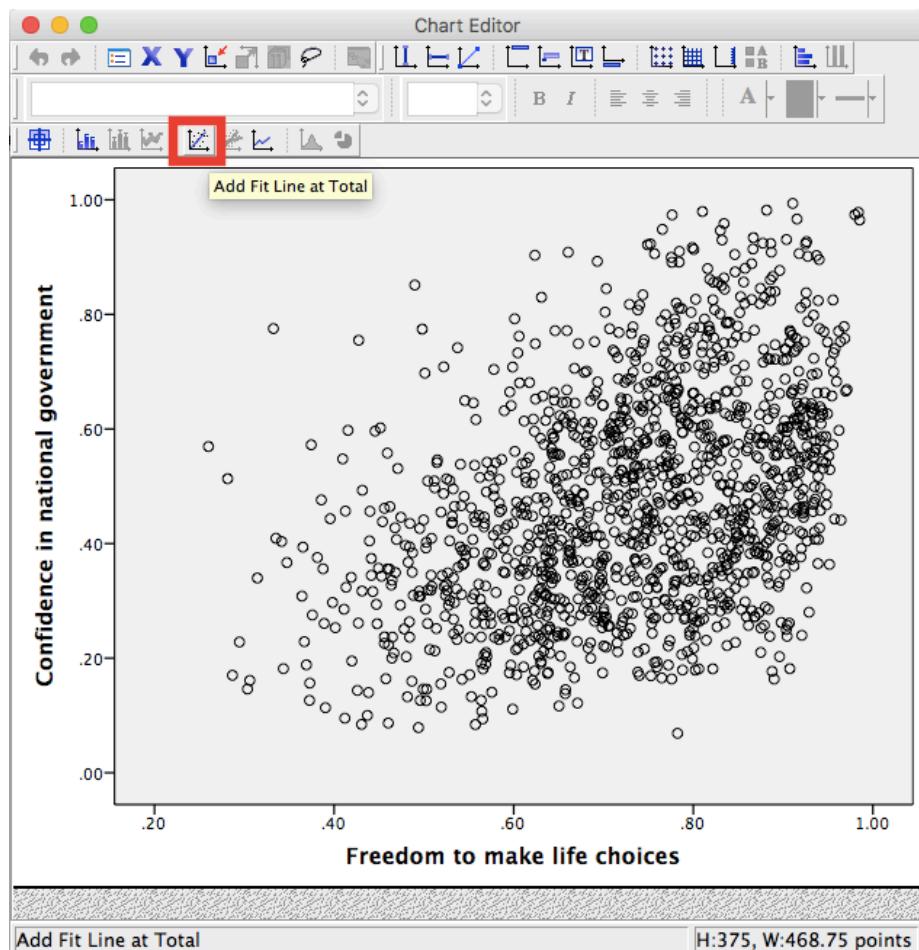
[DataSet1]



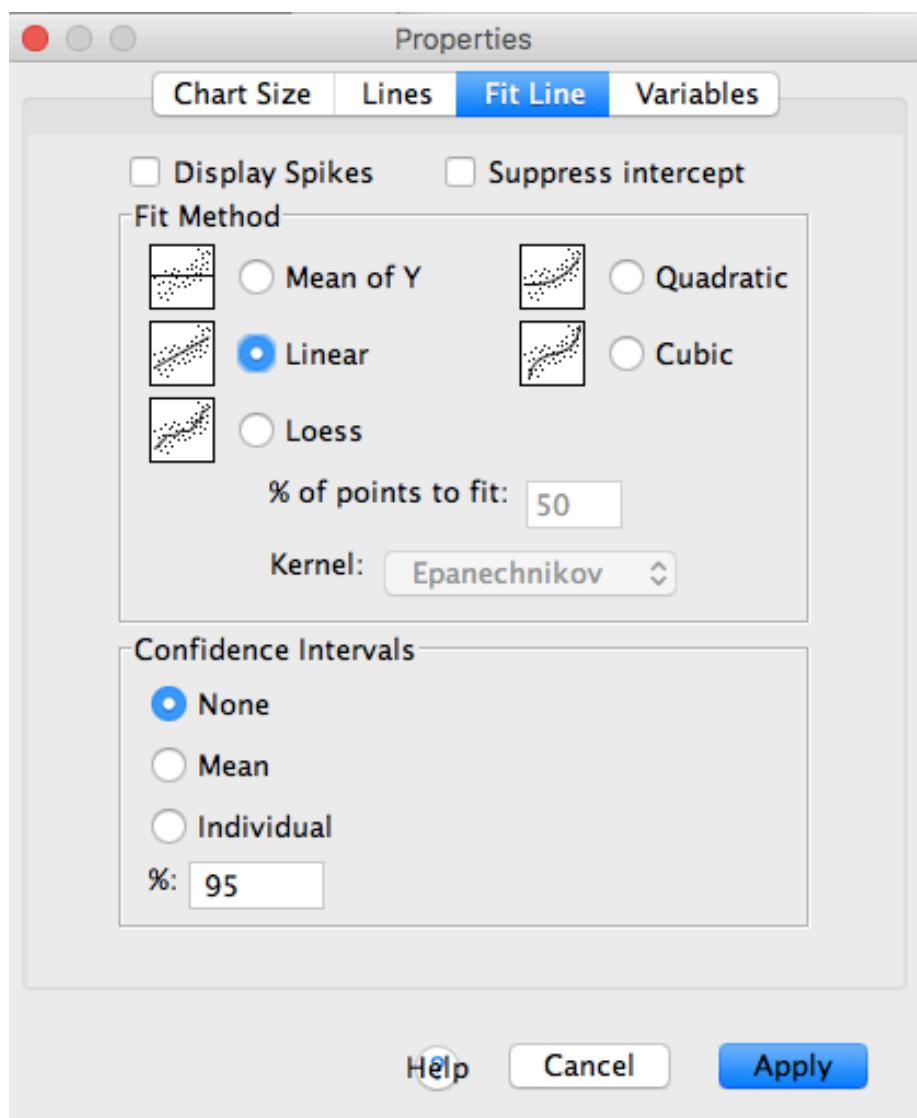
You can keep this scatterplot as it is, or, you can edit it to include a straight line that best fits the data points. This line is known as the best-fitting line as it minimizes the distance from it to all the data. To edit the scatterplot double click on the graph and a window labeled Chart Editor should appear:



In this window, find the button at the top that reads Fit Line at Total when you hover your mouse over it. Below, I have highlighted it for clarity:



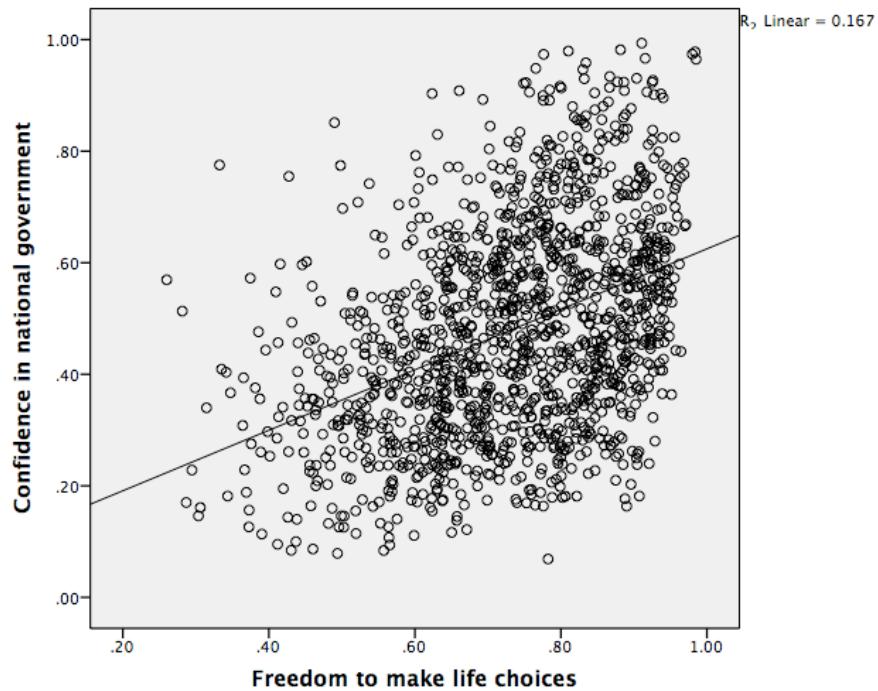
Press this button and you will see a new menu. Make sure Linear is selected and click Apply.



Next, exit from the Chart Editor. This means you will hit the X in the corner of the window. You will find that the graph in your output window has now updated and has a line drawn on it.

Graph

[DataSet1]



This scatterplot is very important. The distance between the line and the data points is indicative of the strength of the correlation coefficient; they are directly related. For example, if the data were more clustered or tighter to the line, the correlation would be stronger. If the data points are more spread out and far from the line, the correlation is weaker.

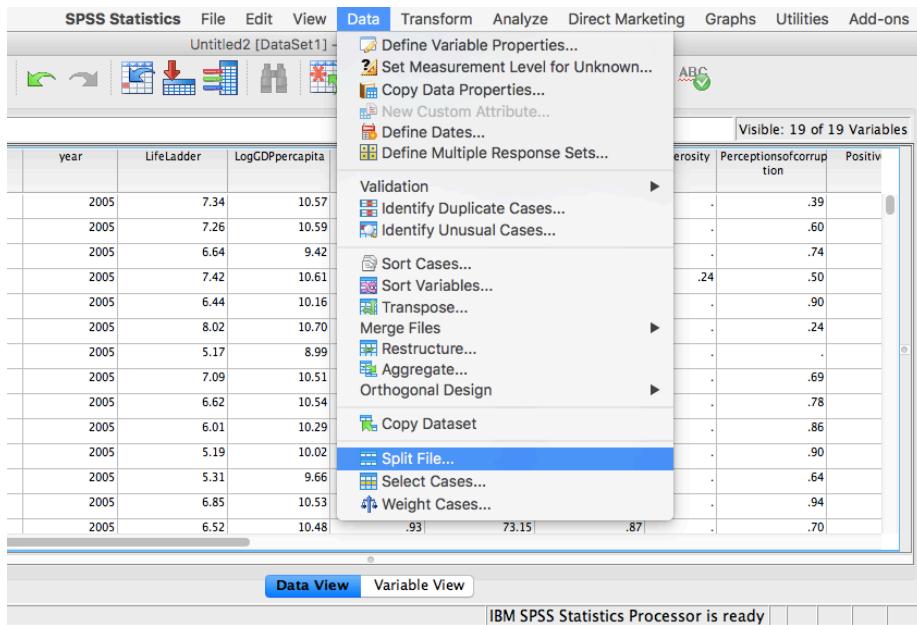
3.4.4 Splitting a File

What if we asked the question: for the year 2017 only, does a countries measure for `freedom to make life choices` correlate with that countries measure for `Confidence in national government`?

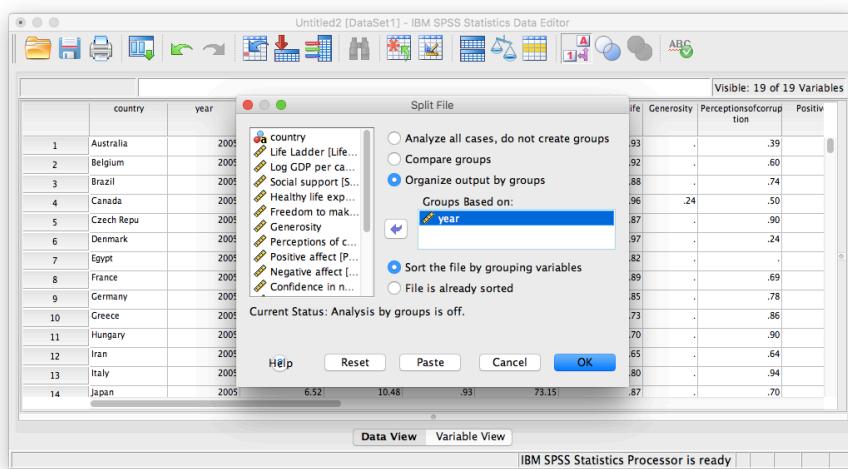
Notice that this question is asking us to find the correlation between the same two variables we used in the previous example, but only in the case where the year is 2017. To achieve this, we're going to utilize a function called splitting. Splitting takes the file as a whole, and sets it up so that every analysis is done on some subset of the data. For example, if we split our data by year and calculate

a correlation coefficient, SPSS will find Pearson r for only 2017, and another for only 2016, and so on.

In order to split the data, we go to the top menu and choose Data, then Split file...

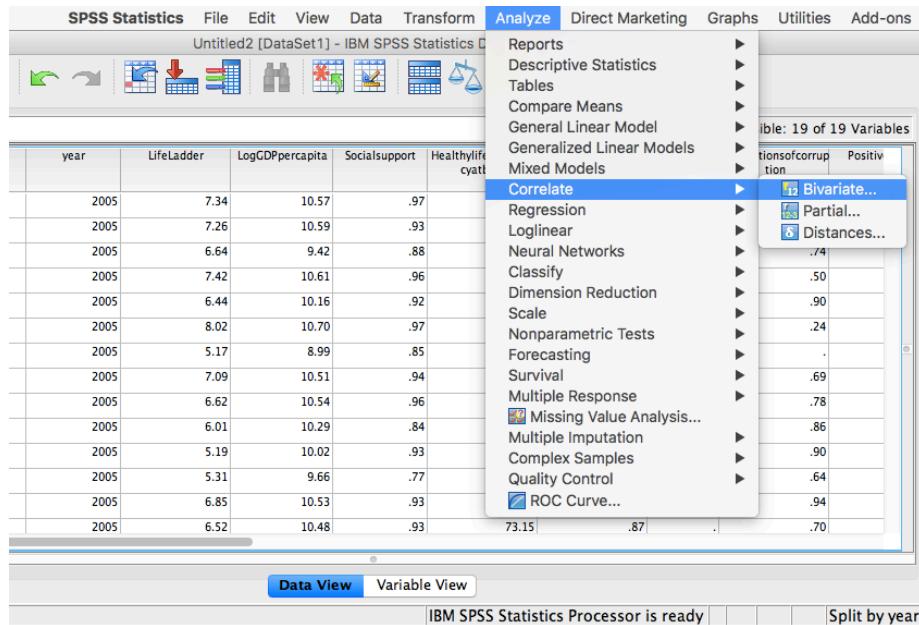


In the next window, you must select Organize output by groups and then specify which variable will be used to split the data. Select **year** and move it to the right-hand field using the arrow.

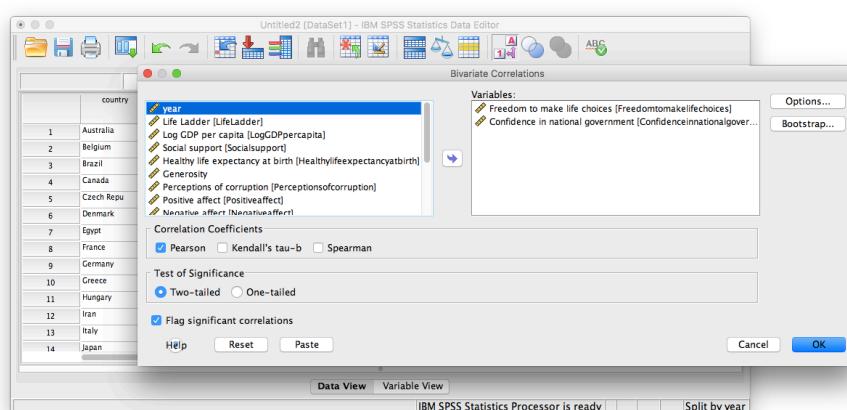


Click OK. Notice that this will cause the output window to produce some next indicating that you have split your file. You can ignore this and go back to your data window.

From here, any analysis you choose to do will be done separately for each year's worth of data. Let's calculate the correlation coefficient, as usual. Click Analyze, then Correlate, then Bivariate:



In the next window, select the variables to be used (they will be the same as in the last example).



Click OK. Notice that in the output window you will see a bunch of correlation

tables (13 of them to be exact); one for each year. Scroll down and find the table with the heading “year = 2017”. That’s the table we need in order to answer our question:

year = 2017

Correlations^a

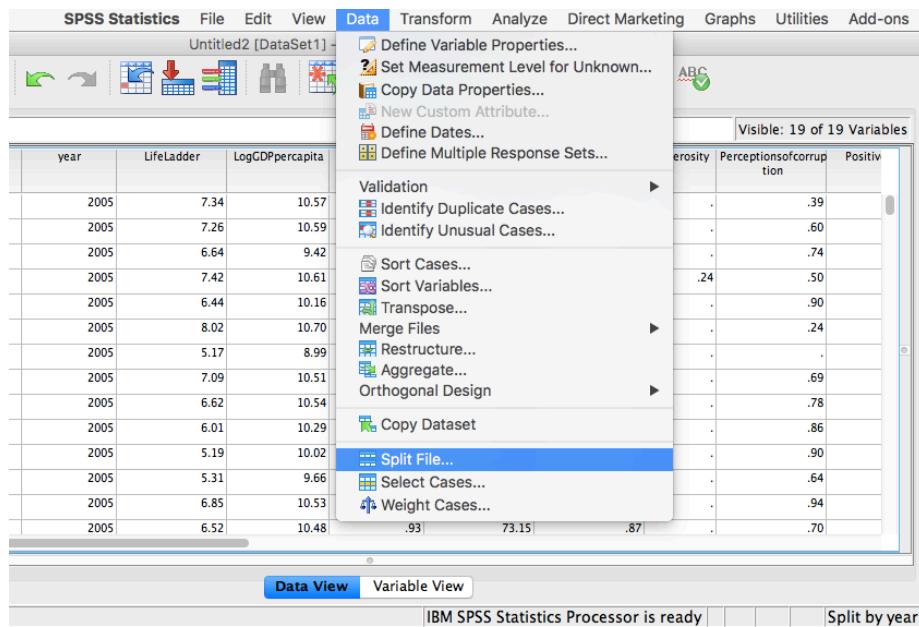
		Freedom to make life choices	Confidence in national government
Freedom to make life choices	Pearson Correlation	1	.442 **
	Sig. (2-tailed)		.000
	N	140	128
Confidence in national government	Pearson Correlation	.442 **	1
	Sig. (2-tailed)	.000	
	N	128	128

**. Correlation is significant at the 0.01 level (2-tailed).

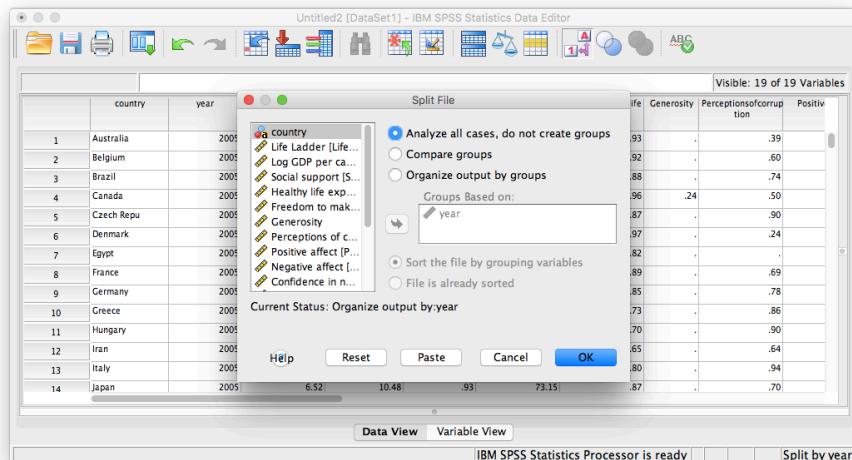
a. year = 2017

This table indicates that, if we only look at the year 2017, the correlation coefficient between **freedom to make life choices** and **confidence in national government** is .442.

It is VERY important to remember that once you have split a file, every analysis that follows the split will be done on the split variable. If you want to go back to performing analyses and calculating statistics for the data as a whole, you must UNSPLIT your data file (or undo the split). To do this, go to Data, then Split file...



Then make sure to select Analyze all cases, do not create groups and click OK.



3.4.5 Practice Problems

1. For the year 2005 ONLY, find the correlation between “perceptions of corruption” and “positive affect”. Create a scatterplot to visualize this relationship. What are your conclusions about the relationship between

affect and perceived corruption? Is this surprising to you?

2. What has happened to log GDP (consider this a measure of GDP) in the United States ONLY with time (as the year has increased)? Explain this relationship and provide a scatterplot.
3. Which country (or countries) have seen a more consistent and strong increase in log GDP over time? Which country (or countries) have seen a decrease over time?

3.5 JAMOVI

How to do it in JAMOVI

Chapter 4

Lab 4: Normal Distribution & Central Limit Theorem

By a small sample, we may judge of the whole piece. —Miguel de Cervantes from Don Quixote

4.1 General Goals

1. Distributions
2. Sampling from distributions
3. Sampling distribution of the mean
4. Sampling statistics (statistics of many samples)
5. Central limit theorem
6. Normal Distribution
7. z-scores

4.2 R

This is one of two special labs where we don't use too much real data. We will mostly fake everything. Yes, you will learn how to fake data in this course. Be a superhero, and only use these skills for good and not for evil.

As we progress through the course, you will learn that generating simulated data can be very useful to help you understand real data. In fact, I will say this right now. If you can't simulate the data you expect to find, then you probably can't understand the data that you do find very well. That's a bold statement. It's probably partly true.

4.2.1 Generating Numbers in R

There are many ways to make R generate numbers for you. In all cases you define how the numbers are generated. We'll go through a few of the many ways.

4.2.1.1 sample

The sample function is like an endless gumball machine. You put the gumballs inside with different properties, say As and Bs, and then you let sample endlessly take gumballs out. Check it out:

```
gumballs <- c("A", "B")
sample_of_gumballs <- sample(gumballs, 10, replace=TRUE)
sample_of_gumballs
```

```
## [1] "A" "B" "A" "A" "B" "A" "A" "A" "B" "A"
```

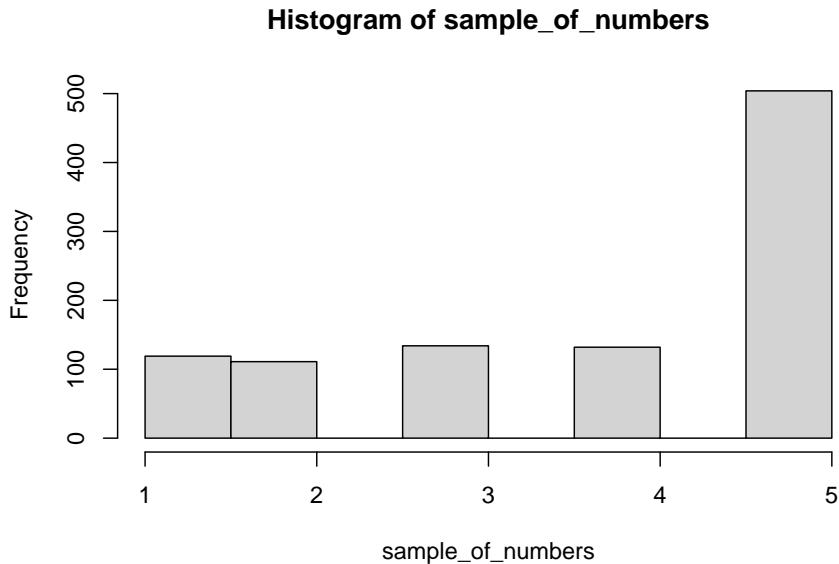
Here the sample function randomly picks A or B each time. We set it do this 10 times, so our sample has 10 things in it. We set `replace=TRUE` so that after each sample, we put the item back into the gumball machine and start again. Here's another example with numbers

```
some_numbers <- c(1,2,3,4,5,5,5,5)
sample_of_numbers <- sample(some_numbers, 20, replace=TRUE)
sample_of_numbers
```

```
## [1] 5 4 5 1 2 3 1 2 3 5 1 1 4 4 4 5 5 2 2 5
```

Let's do one more thing with sample. Let's sample 1000 times from our `some_numbers` variable, and then look at the histogram

```
library(ggplot2)
some_numbers <- c(1,2,3,4,5,5,5,5)
sample_of_numbers <- sample(some_numbers, 1000, replace=TRUE)
hist(sample_of_numbers)
```

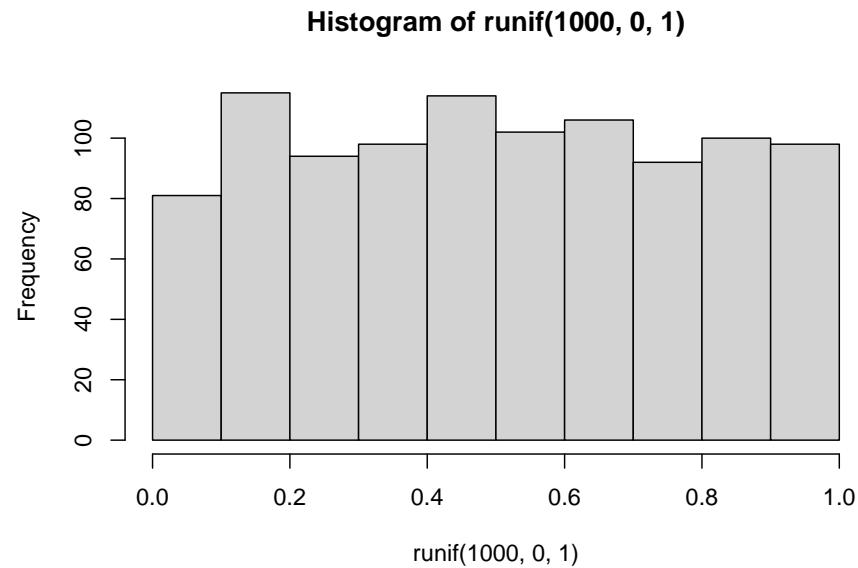


We are looking at lots of samples from our little gumball machine of numbers. We put more 5s in, and voila, more 5s come out of in our big sample of 1000.

4.2.1.2 runif uniform distribution

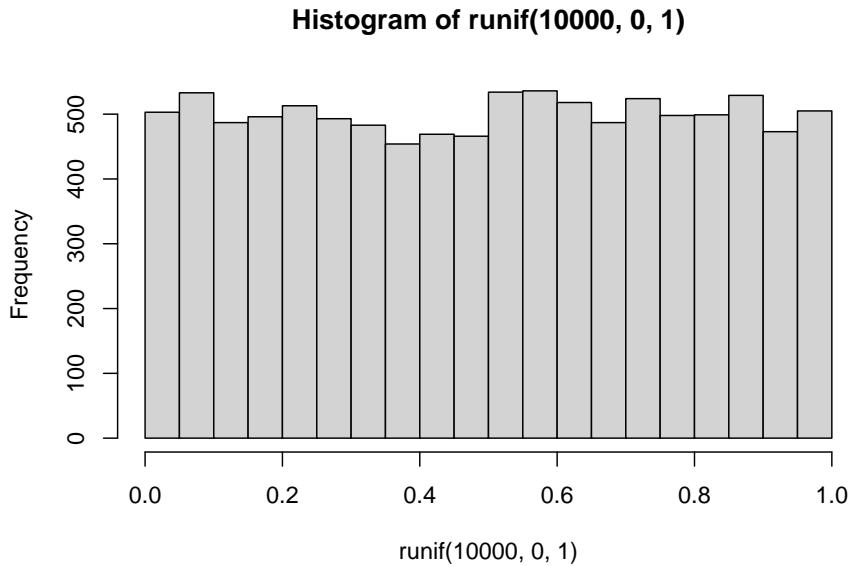
We can sample random numbers between any range using the `runif(n, min=0, max = 1)` function for the uniform distribution. We discussed this in the textbook. A uniform distribution is flat, and all the numbers between the min and max should occur roughly equally frequently. Let's take 1000 random numbers between 0 and 1 and plot the histogram. We'll just do it all in one line for speed.

```
hist(runif(1000,0,1))
```



This histogram is flattish. Not perfectly flat, after all we only took 1000 samples. What if we took many more, say 10,000 total samples? Now it looks more flat, each bin is occurring about 500 times each, which is pretty close to the same amount.

```
hist(runif(10000, 0, 1))
```



4.2.1.3 rbinom the binomial distribution

The binomial distribution sounds like a scary word... binomial (AAGGGGH-HHHH, stay away!). The binomial can be a coin flipping distribution. You use `rbinom(n, size, prob)`. `n` gives the number of samples you want to take. We'll keep `size = 1` for now, it's the number of trials (forget this for now, it's more useful for more complicated things than what we are doing, if you want to know what it does, try it out, and see if you figure it out). `prob` is a little list you make of probabilities, that define how often certain things happen.

For example, consider flipping a coin. It will be heads or tails, and the coin, if it is fair, should have a 50% chance of being heads or tails. Here's how we flip a coin 10 times using `rbinom`.

```
coin_flips <- rbinom(10, 1, .5)
coin_flips
```

```
## [1] 0 1 1 0 1 0 0 1 1 0
```

We get a bunch of 0s, and 1s. We can pretend 0 = tails, and 1 = heads. Great, now we can do coin flipping if we want. For example, if you flip 10 coins, how many heads do you get? We can do the above again, and then `sum(coin_flips)`. All the 1s are heads, so it will work out.

```
coin_flips <- rbinom(10, 1, .5)
sum(coin_flips)
```

```
## [1] 5
```

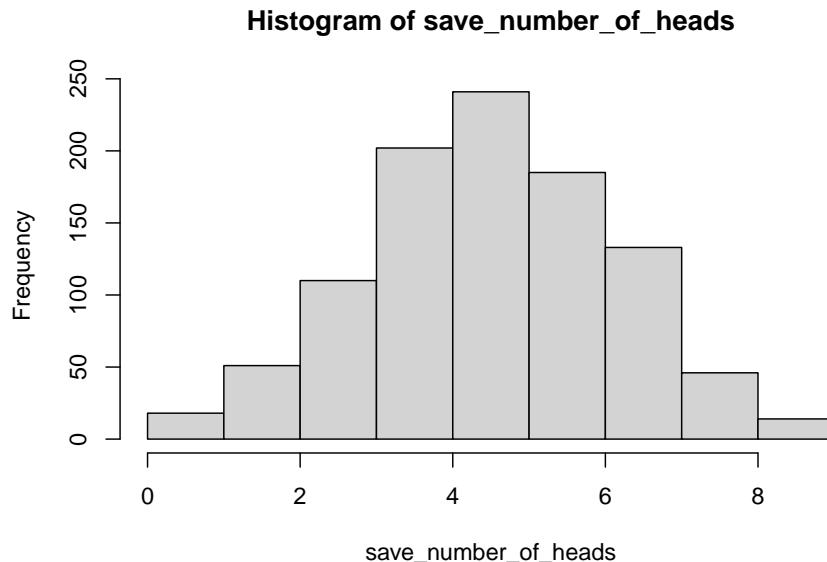
Alright, so we get the sum, which tells us the number of heads. But, should we always get that number of heads if we flipped a coin 10 times? If you keep redoing the above, you'll get different answers. 5 heads will be the most frequent answer, but you will get lots of other answers too.

Hold on to your seats for this next one. With R, we can simulate the flipping of a coin 10 times (you already know that, you just did it), and we can do that over and over as many times as we want. For example, we could do it 100 times over, saving the number of heads for each set of 10 flips. Then we could look at the distribution of those sums. That would tell us about the range of things that can happen when we flip a coin 10 times. We can do that in loop like this:

```
save_number_of_heads<-length(1000) # make an empty variable to save things in

for(i in 1:1000){
  save_number_of_heads[i] <- sum(rbinom(10,1,.5))
}

hist(save_number_of_heads)
```

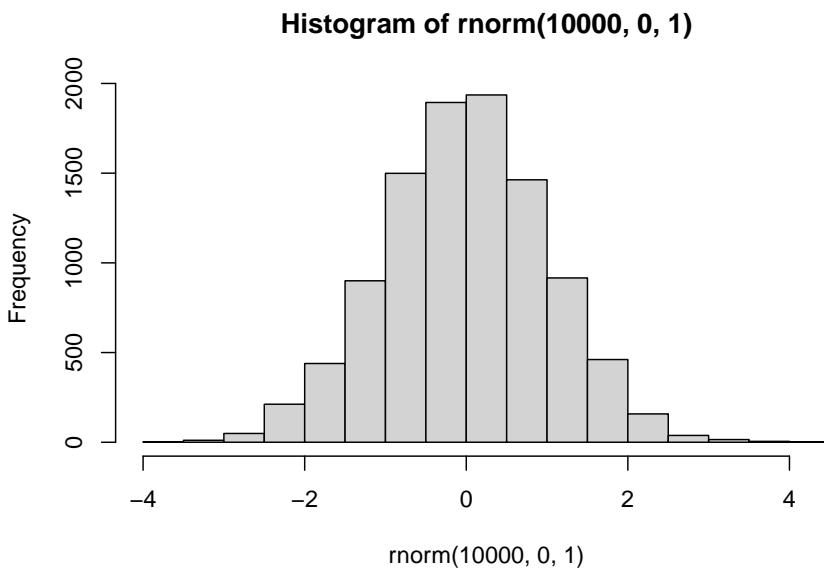


See, that wasn't too painful. Now we see another histogram. The histogram shows us the frequency observing different numbers of heads (for 10 flips) across the 1000 simulations. 5 happens the most, but 2 happens sometimes, and so does 8. All of the possibilities seem to happen sometimes, some more than others.

4.2.1.4 rnorm the normal distribution

We'll quickly show how to use `rnorm(n, mean=0, sd=1)` to sample numbers from a normal distribution. And, then we'll come back to the normal distribution later, because it is so important.

```
hist(rnorm(10000, 0, 1))
```

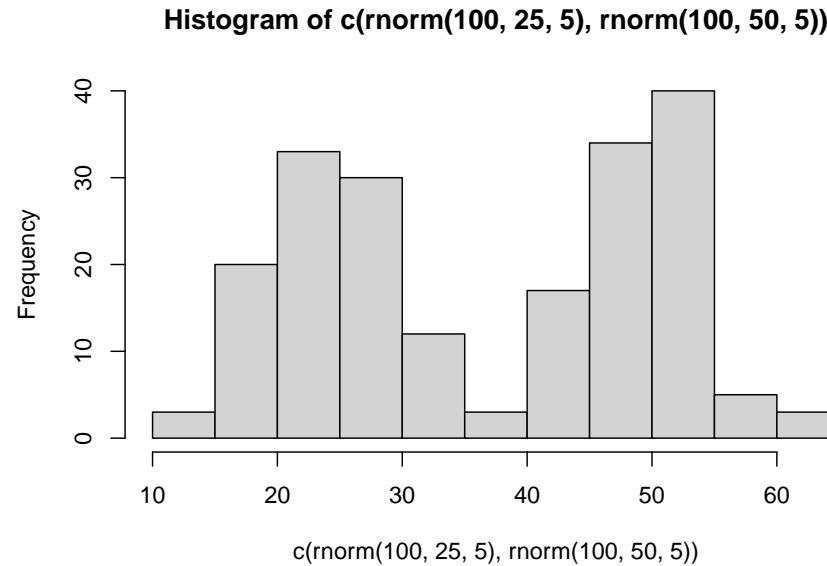


There it is, a bell-shaped normal distribution with a mean of 0, and a standard deviation of 1. You've probably seen things like this before. Now you can sample numbers from normal distributions with any mean or standard deviation, just by changing those parts of the `rnorm` function.

4.2.1.5 mixing it up

The r functions are like Legos, you can put them together and come up with different things. What if wanted to sample from a distribution that looked like a two-humped camel's back? Just sample from `rnorm` twice like this... mix away.

```
hist( c( rnorm(100,25,5), rnorm(100,50,5)) )
```



4.2.1.6 summary

You can generate as many numbers as your computer can handle with R. PSA: Don't ask R to generate a bajillion numbers or it will explode (or more likely just crash, probably won't explode, that's a metaphor).

4.2.2 sampling distribution of the mean.

Remember the sampling distribution of the sample means from the textbook? Now, you will see the R code that made the graphs from before. As we've seen, we can take samples from distributions in R. We can take as many as we want. We can set our sample-size to be anything we want. And, we can take multiple samples of the same size as many times as we want.

4.2.2.1 Taking multiple samples of the same size

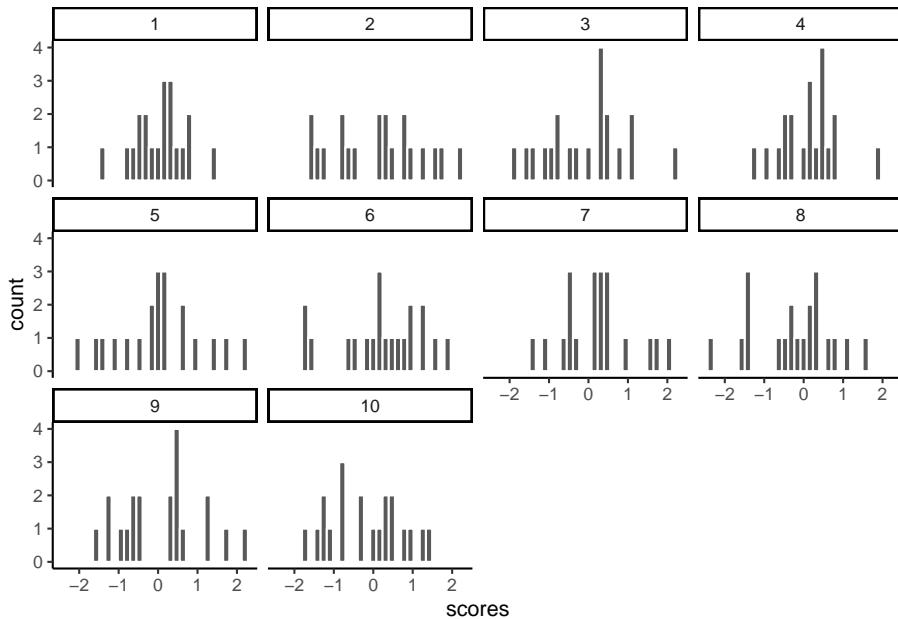
Let's take 10 samples from a normal distribution (mean = 0, and SD = 1). Let's set the sample-size for each to be 20. Then, we'll put them all in a data frame and look at 10 different histograms, one for each sample.

```
scores <- rnorm(10*20,0,1)
samples <- rep(1:10,each=20)
my_df <- data.frame(samples,scores)
```

First, look at the new my_df data frame. You can see there is a column with numbers 1 to 10, these are the sample names. There are also 20 scores for each

in the scores column. Let's make histograms for each sample, so we can see what all of the samples look like:

```
ggplot(my_df, aes(x=scores))+
  geom_histogram(color="white")+
  facet_wrap(~samples)+
  theme_classic()
```



Notice, all of the samples do not have the same looking histogram. This is because of random sampling error. All of the samples are coming from the same normal distributions, but random chance makes each sample a little bit different (e.g., you don't always get 5 heads and 5 tails when you flip a coin right)

4.2.2.2 Getting the means of the samples

Now, let's look at the means of the samples, we will use `dplyr` to get the means for each sample, and put them in a table:

```
library(dplyr)

sample_means <- my_df %>%
  group_by(samples) %>%
  summarise(means=mean(scores))

knitr::kable(sample_means)
```

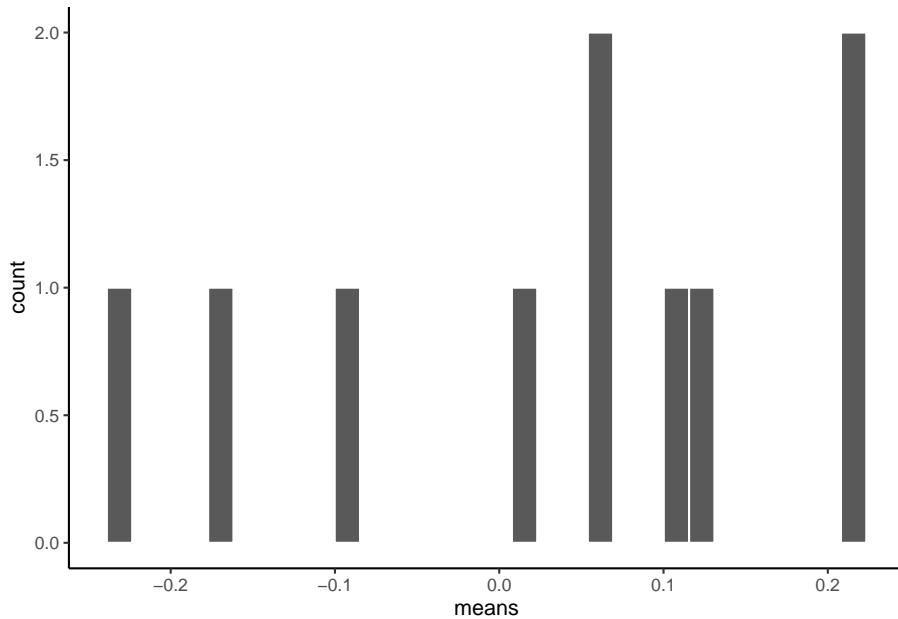
samples	means
1	0.0634425
2	0.1141997
3	-0.0892645
4	0.1192194
5	0.0097653
6	0.2140962
7	0.2181609
8	-0.2286859
9	0.0578422
10	-0.1619086

So, those are the means of our samples. What should the means be? Well, we would hope they are estimating the mean of the distribution they came from, which was 0. Notice, the numbers are all not 0, but they are kind of close to 0.

4.2.2.3 histogram for the means of the samples

What if we now plot these 10 means (of each of the samples) in their own distribution?

```
ggplot(sample_means, aes(x=means))+
  geom_histogram(color="white")+
  theme_classic()
```



That is the distribution of the sample means. It doesn't look like much eh?

That's because we only took 10 samples right.

Notice one more thing...What is the mean of our 10 sample means? This is a mean of means. Remember that.

```
mean(sample_means$means)
```

```
## [1] 0.03168671
```

Well, that's pretty close to zero. Which is good. When we average over our samples, they better estimate the mean of the distribution they came from.

4.2.2.4 simulating the distribution of sample means

Our histogram with 10 sample means looked kind of sad. Let's give it some more friends. How about we repeat our little sampling experiment 1000 times.

Explain...We take 1000 samples. Each sample takes 20 scores from a normal distribution (mean=0, SD=1). Then we find the means of each sample (giving us 1000 sample means). Then, we plot that distribution.

```
# get 1000 samples with 20 scores each

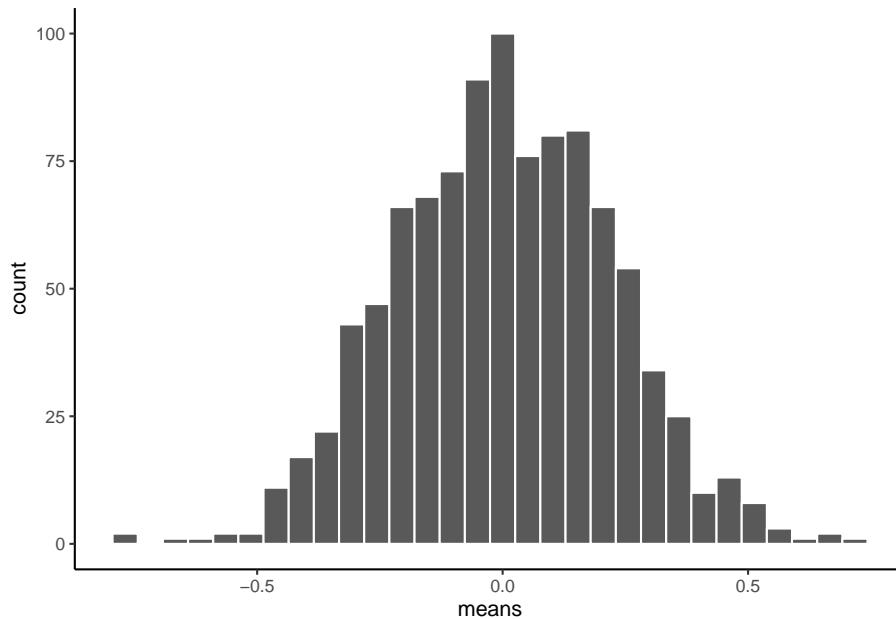
scores <- rnorm(1000*20,0,1)
samples <- rep(1:1000,each=20)
my_df <- data.frame(samples,scores)

# get the means of the samples

sample_means <- my_df %>%
  group_by(samples) %>%
  summarise(means=mean(scores))

# make a histogram

ggplot(sample_means, aes(x=means))+
  geom_histogram(color="white")+
  theme_classic()
```



There, that looks more like a sampling distribution of the sample means. Notice its properties. It is centered on 0, which tells us that sample means are mostly around zero. It is also bell-shaped, like the normal distribution it came from. It is also quite narrow. The numbers on the x-axis don't go much past -.5 to +.5.

We will use things like the sampling distribution of the mean to make inferences about what chance can do in your data later on in this course.

4.2.3 Sampling distributions for any statistic

Just for fun here are some different sampling distributions for different statistics. We will take a normal distribution with mean = 100, and standard deviation =20. Then, we'll take lots of samples with n = 50 (50 observations per sample). We'll save all of the sample statistics, then plot their histograms. We do the sample means, standard deviations, maximum values, and medians. Let's do it.

```
all_df<-data.frame()
for(i in 1:1000){
  sample<-rnorm(50,100,20)
  sample_mean<-mean(sample)
  sample_sd<-sd(sample)
  sample_max<-max(sample)
  sample_median<-median(sample)
  t_df<-data.frame(i,sample_mean,sample_sd,sample_max,sample_median)
  all_df<-rbind(all_df,t_df)
}
```

```

library(ggpubr)
a<-ggplot(all_df,aes(x=sample_mean))+  

  geom_histogram(color="white")+
  theme_classic()
b<-ggplot(all_df,aes(x=sample_sd))+  

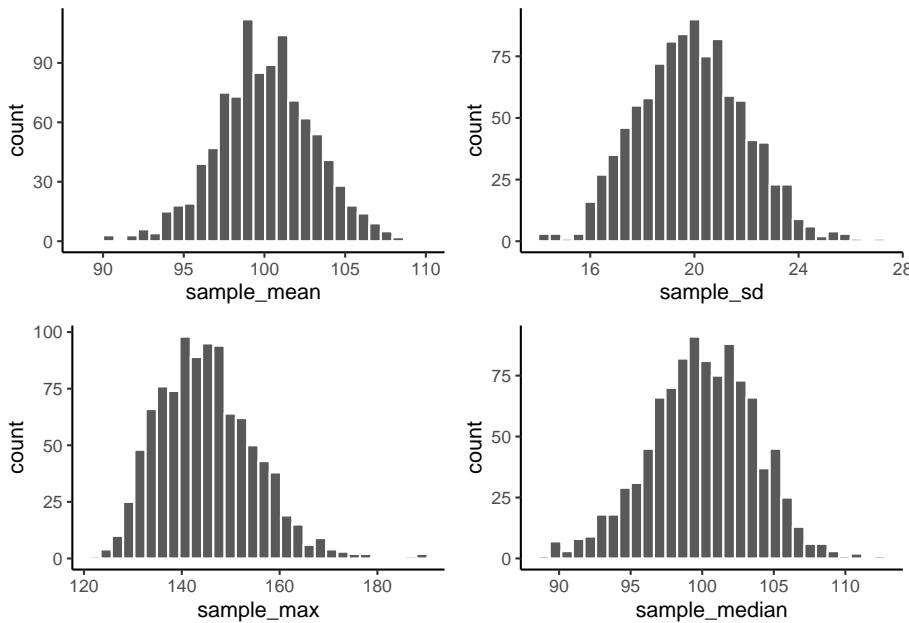
  geom_histogram(color="white")+
  theme_classic()
c<-ggplot(all_df,aes(x=sample_max))+  

  geom_histogram(color="white")+
  theme_classic()
d<-ggplot(all_df,aes(x=sample_median))+  

  geom_histogram(color="white")+
  theme_classic()

ggarrange(a,b,c,d,
          ncol = 2, nrow = 2)

```



From reading the textbook and attending lecture, you should be able to start thinking about why these sampling statistic distributions might be useful...For now, just know that you can make a sampling statistic for pretty much anything in R, just by simulating the process of sampling, measuring the statistic, doing it over a bunch, and then plotting the histogram. This gives you a pretty good estimate of the distribution for that sampling statistic.

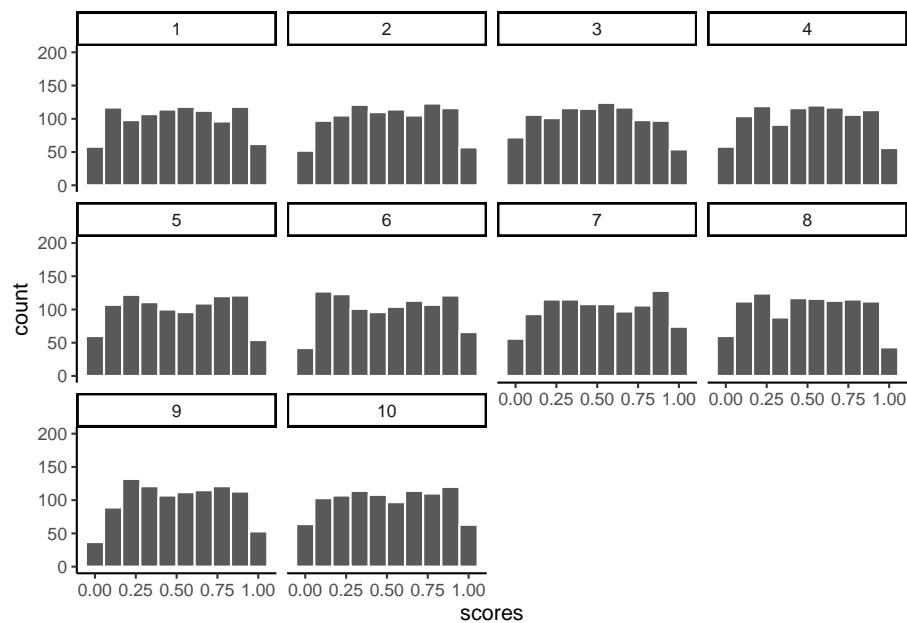
4.2.4 Central limit theorem

We have been building you up for the central limit theorem, described in the textbook and in class. The central limit theorem is basically that the distribution of sample means will be a normal curve. We already saw that before. But, the interesting thing about it, is that the distribution of your sample means will be normal, even if the distribution the samples came from is not normal. Huh what?

To demonstrate this the next bit of code is modified from what we did earlier. We create 100 samples. Each sample has 1000 observations. All of them come from a uniform distribution between 0 to 1. This means all of the numbers between 0 and 1 should occur equally frequently. Below I plot histograms for the first 10 samples (out of the 100 total, 100 is too many to look at). Notice the histograms are not normal, they are roughly flat.

```
scores <- runif(100*1000,0,1)
samples <- rep(1:100,each=1000)
my_df <- data.frame(samples,scores)

ggplot(my_df[1:(10*1000),], aes(x=scores))+
  geom_histogram(color="white", bins=10)+
  facet_wrap(~samples)+
  theme_classic()+
  ylim(0,200)
```



We took samples from a flat uniform distribution, and the samples themselves

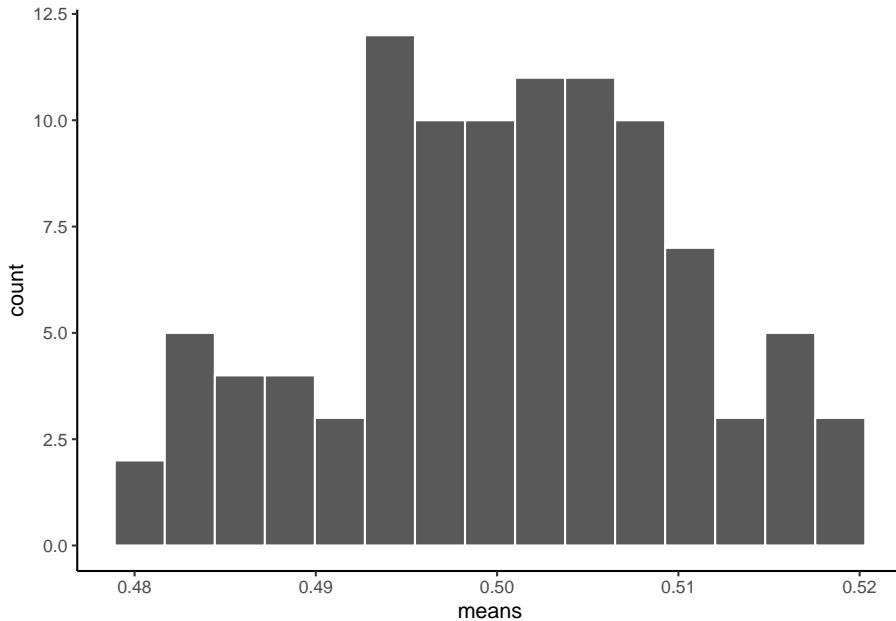
look like that same flat distribution.

HOWEVER, if we now do the next step, and compute the means of each of our 100 samples, we could then look at the sampling distribution of the sample means. Let's do that:

```
sample_means <- my_df %>%
  group_by(samples) %>%
  summarise(means=mean(scores))

# make a histogram

ggplot(sample_means, aes(x=means))+
  geom_histogram(color="white", bins=15)+
  theme_classic()
```



As you can see, the sampling distribution of the sample means is not flat. It's shaped kind of normal-ish. If we had taken many more samples, found their means, and then looked at a histogram, it would become even more normal looking. Because that's what happens according to the central limit theorem.

4.2.5 The normal distribution

“Why does any of this matter, why are we doing this, can we stop now!!!!!! PLEEEEAASSEE, somebody HELP”.

We are basically just repeating what was said in the textbook, and the lecture,

so that you get the concept explained in a bunch of different ways. It will sink in.

The reason the central limit theorem is important, is because researchers often take many samples, then analyse the means of their samples. That's what they do.

An experiment might have 20 people. You might take 20 measurements from each person. That's taking 20 samples. Then, because we know that samples are noisy. We take the means of the samples.

So, what researchers are often looking at, (and you too, very soon) are means of samples. Not just the samples. And, now we know that means of samples (if we had a lot of samples), look like they are distributed normally (the central limit theorem says they should be).

We can use this knowledge. If we learn a little bit more about normal distributions, and how they behave and work, we can take that and use it to understand our sample means better. This will become more clear as head into the topic of statistical inference next week. This is all a build up for that.

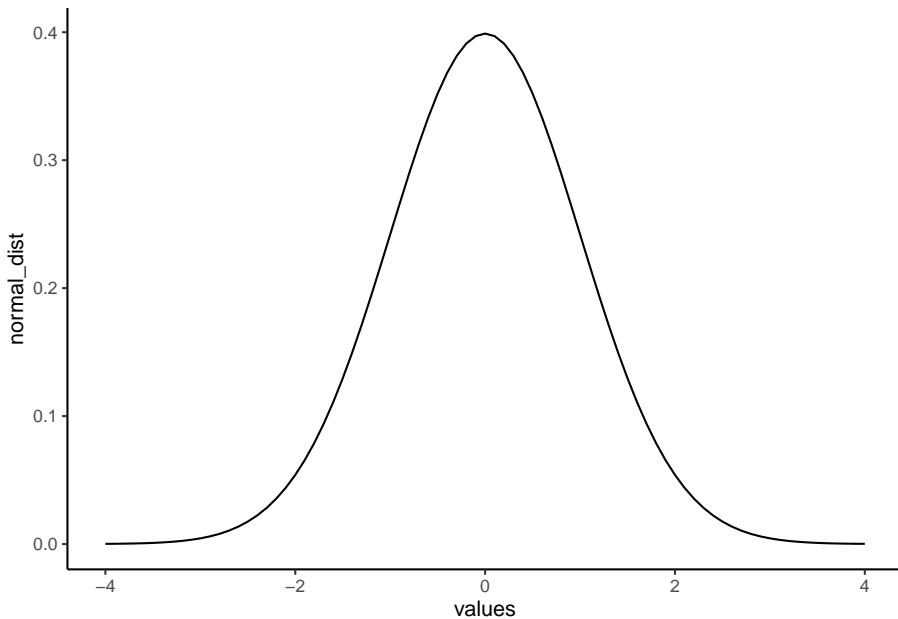
To continue the build-up we now look at some more properties of the normal distribution.

4.2.5.1 Graphing the normal distribution

“Wait, I thought we already did that”. We sort of did. We sampled numbers and made histograms that looked like normal distributions. But, a “normal distribution” is more of an abstract idea. It looks like this in the abstract:

```
normal_dist <- dnorm(seq(-4,4,.1), 0, 1)
values <- seq(-4,4,.1)
normal_df <- data.frame(values,normal_dist)

ggplot(normal_df, aes(x=values,y=normal_dist))+
  geom_line()+
  theme_classic()
```



A really nice shaped bell-like thing. This normal distribution has a mean of 0, and standard deviation of 1. The heights of the lines tell you roughly how likely each value is. Notice, it is centered on 0 (most likely that numbers from this distribution will be near 0), and it goes down as numbers get bigger or smaller (so bigger or smaller numbers get less likely). There is a range to it. Notice the values don't go much beyond -4 and +4. This because those values don't happen very often. Theoretically any value could happen, but really big or small values have really low probabilities.

4.2.5.2 calculating the probability of specific ranges.

We can use R to tell us about the probability of getting numbers in a certain range. For example, when you think about. It should be obvious that you have a 50% probability of getting the number 0 or greater. Half of the distribution is 0 or greater, so you have a 50% probability.

We can use the `pnorm` function to confirm this:

```
pnorm(0, mean = 0, sd= 1, lower.tail=FALSE)
```

```
## [1] 0.5
```

Agreed, `pnorm` tells us the probability of getting 0 or greater is .5.

Well, what is the probability of getting a 2 or greater? That's a bit harder to judge, obviously less than 50%. Use R like this to find out:

```
pnorm(2, mean = 0, sd= 1, lower.tail=FALSE)
```

```
## [1] 0.02275013
```

The probability of getting a 2 or greater is .0227 (not very probable)

What is the probability of getting a score between -1 and 1?

```
ps<-pnorm(c(-1,1), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

```
## [1] 0.6826895
```

About 68%. About 68% of all the numbers would be between -1 and 1. So naturally, about 34% of the numbers would be between 0 and 1. Notice, we are just getting a feeling for this, you'll see why in a bit when we do z-scores (some of you may realize we are already doing that...)

What about the numbers between 1 and 2?

```
ps<-pnorm(c(1,2), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

```
## [1] 0.1359051
```

About 13.5% of numbers fall in that range, not much.

How about between 2 and 3?

```
ps<-pnorm(c(2,3), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

```
## [1] 0.02140023
```

Again a very small amount, only 2.1 % of the numbers, not a lot.

4.2.5.3 summary pnorm

You can always use `pnorm` to figure how the probabilities of getting certain values from any normal distribution. That's great.

4.2.6 z-scores

We just spent a bunch of time looking at a very special normal distribution, the one where the mean = 0, and the standard deviation = 1. Then we got a little bit comfortable with what those numbers mean. 0 happens a lot. Numbers between -1 and 1 happen a lot. Numbers bigger or smaller than 1 also happen fairly often, but less often. Number bigger than 2 don't happen a lot, numbers bigger than 3 don't happen hardly at all.

We can use this knowledge for our convenience. Often, we are not dealing with numbers exactly like these. For example, someone might say, I got a number,

it's 550. It came from a distribution with mean = 600, and standard deviation = 25. So, does 545 happen a lot or not? The numbers don't tell you right away.

If we were talking about our handy distribution with mean = 0 and standard deviation = 1, and I told I got a number 4.5 from that distribution. You would automatically know that 4.5 doesn't happen a lot. Right? Right!

z-scores are a way of transforming one set of numbers into our neato normal distribution, with mean = 0 and standard deviation = 1.

Here's a simple example, like what we said in the textbook. If you have a normal distribution with mean = 550, and standard deviation 25, then how far from the mean is the number 575? It's a whole 25 away ($550+25 = 575$). How many standard deviations is that? It's 1 whole standard deviation. So does a number like 575 happen a lot? Well, based on what you know about normal distributions, 1 standard deviation of the mean isn't that far, and it does happen fairly often. This is what we are doing here.

4.2.6.1 Calculating z-scores

1. get some numbers

```
some_numbers <- rnorm(20, 50, 25)
```

2. Calculate the mean and standard deviation

```
my_mean <- mean(some_numbers)
```

```
my_sd <- sd(some_numbers)
```

```
print(my_mean)
```

```
## [1] 46.49733
```

```
print(my_sd)
```

```
## [1] 25.58757
```

3. subtract the mean from your numbers

```
differences <- some_numbers - my_mean
```

```
print(differences)
```

```
## [1] -14.3883321 -10.8532004 -0.3117502 -9.8412179 51.2391974 17.9532975
## [7] -22.8777204 13.5897467 -31.9097215 7.5526904 -32.7538454 8.8513848
## [13] 23.4865478 -4.2672600 -31.4155690 -16.8536402 -0.9780879 -2.2303676
## [19] 64.8856899 -8.8778419
```

4. divide by the standard deviation

```
z_scores <- differences / my_sd
```

```
print(z_scores)
```

138CHAPTER 4. LAB 4: NORMAL DISTRIBUTION & CENTRAL LIMIT THEOREM

```
## [1] -0.56231714 -0.42415901 -0.01218366 -0.38460924  2.00250307  0.70164123
## [7] -0.89409491  0.53110726 -1.24707877  0.29517023 -1.28006837  0.34592512
## [13]  0.91788877 -0.16677079 -1.22776657 -0.65866501 -0.03822511 -0.08716604
## [19]  2.53582803 -0.34695910
```

Done. Now you have converted your original numbers into what we call standardized scores. They are standardized to have the same properties (assumed properties) as a normal distribution with mean = 0, and SD = 1.

You could look at each of your original scores, and try to figure out if they are likely or unlikely numbers. But, if you make them into z-scores, then you can tell right away. Numbers close to 0 happen a lot, bigger numbers closer to 1 happen less often, but still fairly often, and numbers bigger than 2 or 3 hardly happen at all.

4.2.7 Generalization Exercise

(1 point - Pass/Fail)

Complete the generalization exercise described in your R Markdown document for this lab.

1. Simulate the sampling distribution of the mean for sample-size =10, from a normal distribution with mean =100, and standard deviation = 25. Run the simulation 1000 times, taking 1000 samples, and computing the sample mean each time.
 - a. Plot the sampling distribution of the mean in a histogram
 - b. Report the mean of the sampling distribution of the mean
 - c. Report the standard deviation of the sampling distribution of the mean
2. Repeat all of the above, except change the sample-size to 100 for all simulations
 - a. Plot the sampling distribution of the mean in a histogram
 - b. Report the mean of the sampling distribution of the mean
 - c. Report the standard deviation of the sampling distribution of the mean

4.2.8 Writing assignment

(2 points - Graded)

Complete the writing assignment described in your R Markdown document for this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

1. Explain the concept of sampling error (1 point)
2. Explain why the standard deviation of the sampling distribution of mean gets smaller as sample-size increases (1 point)

General grading.

- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

4.3 Excel

How to do it in Excel

4.4 SPSS

In this lab, we will use SPSS to transform a set of raw data into z-scores using two methods:

1. Saving data as standardized values
2. Computing a new variable manually

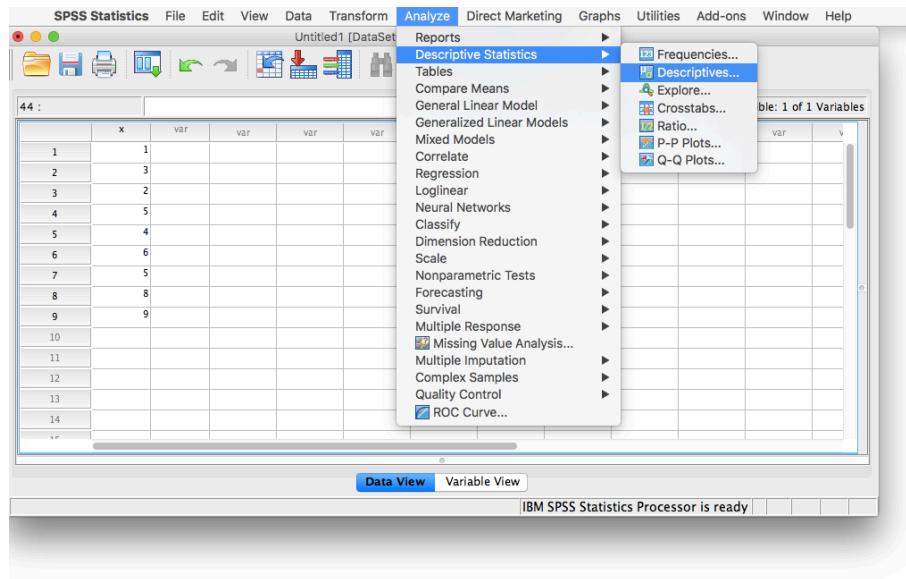
Let's first begin with a short data set we will enter into a new SPSS data spreadsheet. Let's use the following data set: $\{x = 1, 3, 2, 5, 4, 6, 5, 8, 9\}$. Enter these into SPSS and name them appropriately.

	x	var	v								
1	1										
2		3									
3			2								
4				5							
5					4						
6						6					
7							5				
8								8			
9									9		
10											
11											
12											
13											
14											
v											

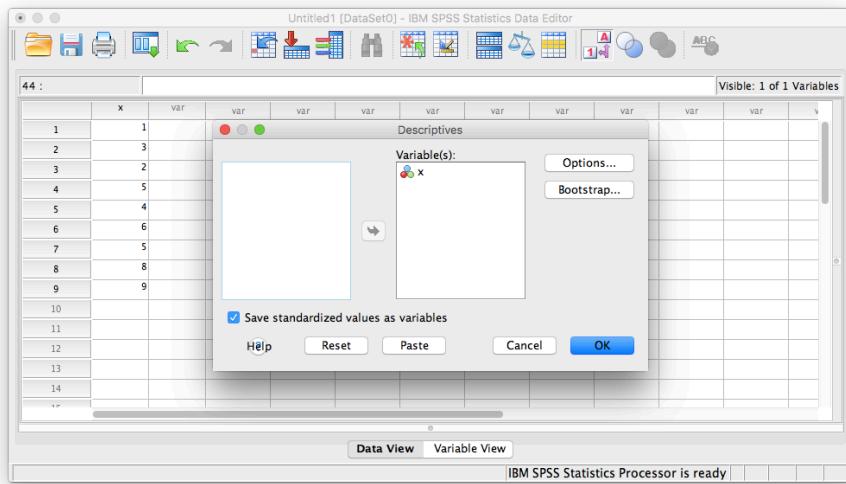
4.4.1 Saving data as standardized values.

140CHAPTER 4. LAB 4: NORMAL DISTRIBUTION & CENTRAL LIMIT THEOREM

This method is a quick and simple way to turn a set of scores in raw format into z-scores. First, go to Analyze, then Descriptive Statistics, then Descriptives:



A window will appear, asking for you to specify for which variable you would like descriptive statistics. Move the x variable into the right field using the arrow. Then, make sure to check off Save standardized values as variables:



Now, click Ok.

SPSS will produce an output table containing some basic descriptive statistics

for the **x** variable:

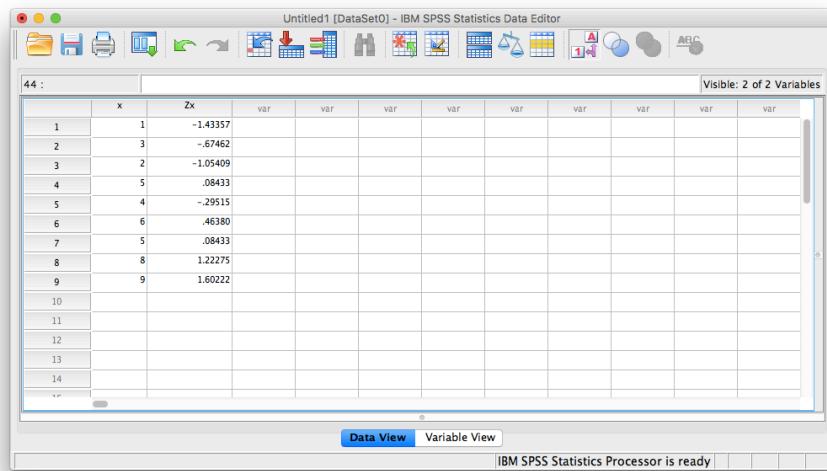
Descriptives

[DataSet0]

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
x	9	1	9	4.78	2.635
Valid N (listwise)	9				

However, you can ignore this window for now. Go back to data view in your spreadsheet and you will notice a new column of data (a new variable) as been created called **Zx**. This variable contains the z-scores that correspond to the values in the **x** column:

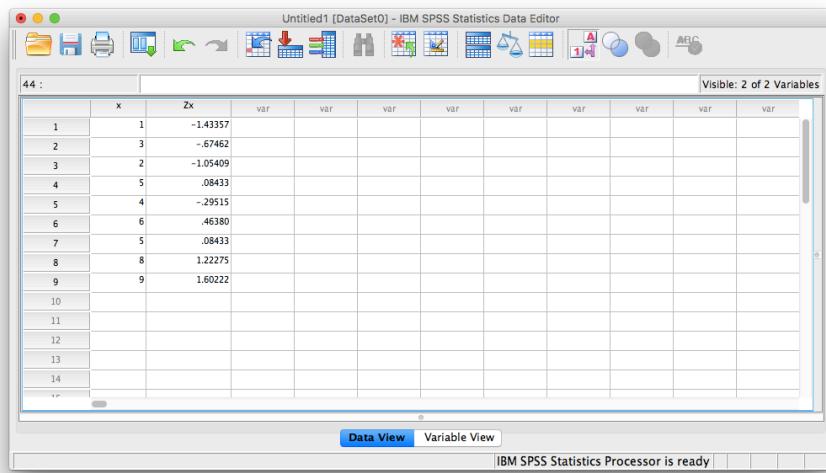


4.4.2 Computing variables

If you use the method shown above to convert a set of measurements to z-scores, you don't need to use this method to do the same thing. However, computing new variables may be useful for more than just converting scores to z-scores, so it's worth illustrating as well. The take-away message here is that you can use SPSS to create new variables based on existing ones.

142CHAPTER 4. LAB 4: NORMAL DISTRIBUTION & CENTRAL LIMIT THEOREM

To illustrate how this is done, let's work with the data sheet we used in the previous example. Currently, it includes a column for our x variable, and the new Zx variable we created using the Analyze > Descriptive Statistics > Descriptives menu.



This time, we are going to turn our x variable measurements into z-scores manually. In order to do this, we're going to need the necessary components to calculate z-scores: We will need the mean and standard deviation of the x variable. Remember, in the previous section a table of descriptive statistics was generated:

Descriptives

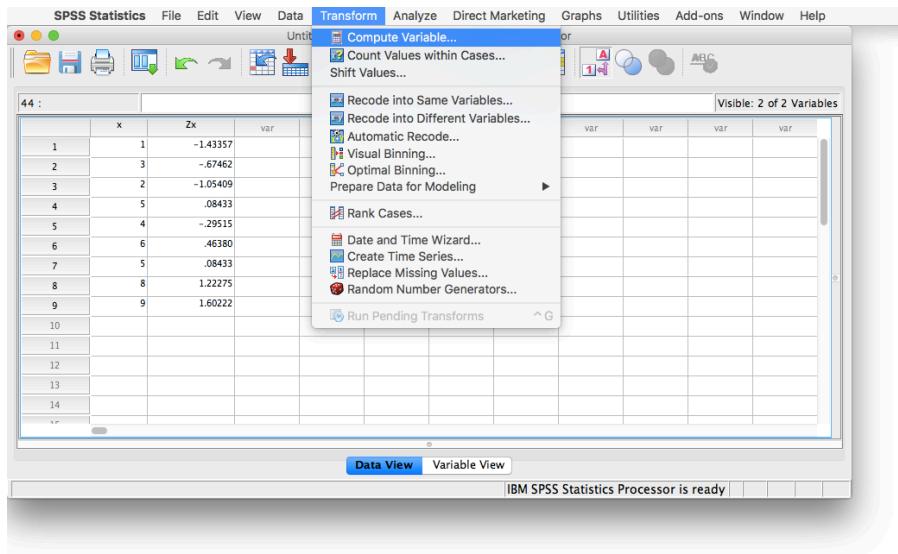
[DataSet0]

Descriptive Statistics

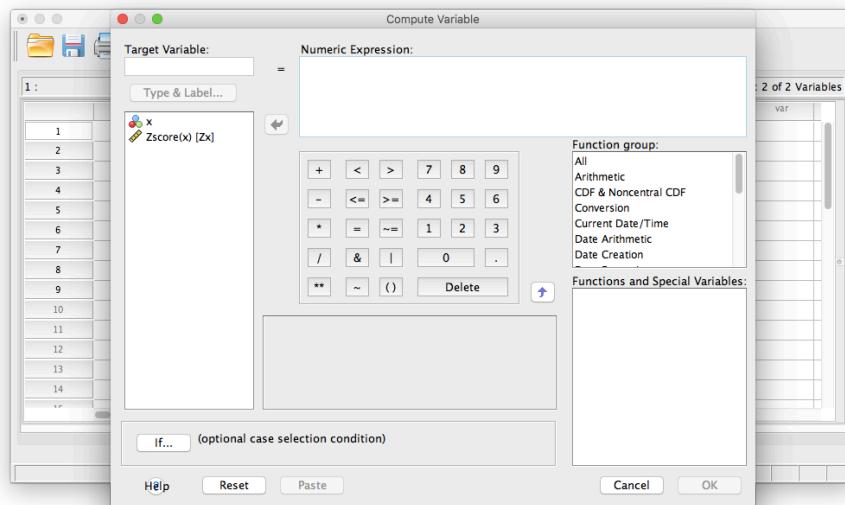
	N	Minimum	Maximum	Mean	Std. Deviation
x	9	1	9	4.78	2.635
Valid N (listwise)	9				

We know that the mean of variable x is 4.78 and the standard deviation is 2.635. In order to turn every measurement in variable x into a z-score, then, we must subtract 4.78 from each x , then divide that difference by 2.635. We are going to ask SPSS to do this by going to the Transform menu, and then selecting

Compute Variable.



This will open the Compute window:



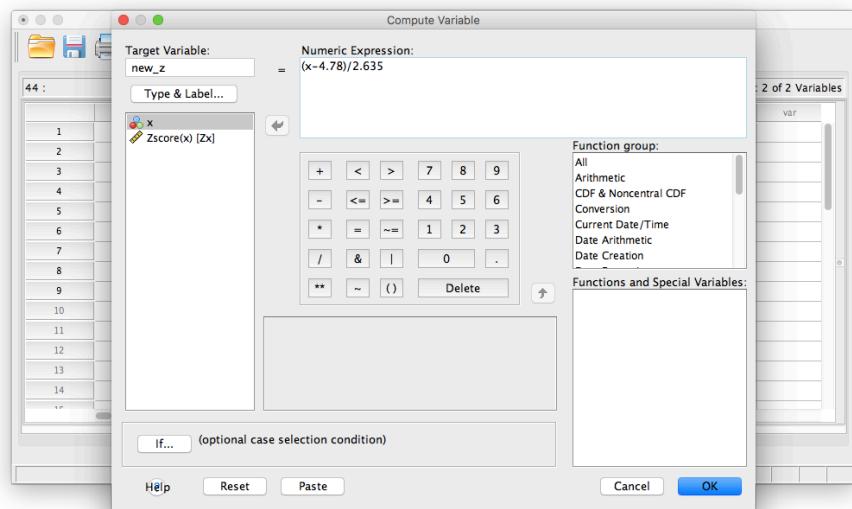
Here, let's first enter a name for the new variable we're about to create. In the field labeled "Target Variable", enter a name for your new variable. Let's try new_z as the name.

Next, in the field labeled "Numeric Expression", we will specify the calculation

144CHAPTER 4. LAB 4: NORMAL DISTRIBUTION & CENTRAL LIMIT THEOREM

to be made. Because I want SPSS to take the variable x , subtract 4.78 from it, and then divide this difference by 2.635, I am going to type in this formula: 1. First, type in an open parenthesis 2. Second, select x from the field on the left and move it to the “Numeric Expression” field using the arrow. 3. Type a minus sign, followed by 4.78 4. Type in a closed parenthesis. 5. Type in a backslash “/” (this specifies division) 6. Type in the standard deviation, 2.635

Your Compute window should now look like this:



Now, click OK. SPSS will generate a blank output window that confirms you have computed a variable. You may minimize or close this window and go back to your SPSS spreadsheet. It should now have a new variable listed, `new_z`, and the computed z-scores should look very similar to the previously calculated z-scores (any differences are due to using rounded figures):

The screenshot shows the IBM SPSS Statistics Data Editor window titled "Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor". The window displays a data table with three columns: "x", "Zx", and "new_z". The "x" column contains raw scores, the "Zx" column contains z-scores, and the "new_z" column contains another set of z-scores. The table has 14 rows, numbered 1 through 14. The "Data View" tab is selected at the bottom of the window. The status bar at the bottom right indicates "IBM SPSS Statistics Processor is ready".

	x	Zx	new_z	var							
1	1	-1.43357	-1.435								
2	3	-.67462	-.676								
3	2	-1.05409	-1.055								
4	5	.08443	.083								
5	4	-.29515	-.296								
6	6	.46380	.463								
7	5	.08443	.083								
8	8	1.22275	1.222								
9	9	1.60222	1.602								
10											
11											
12											
13											
14											

You can use this Compute Variable window to calculate z-scores or make any calculations using existing variables.

4.4.3 Practice Problems

"Professor, do you grade on a curve?"

This is probably the most commonly-asked question in Statistics class. Everyone assumes that grading on a curve will benefit them. But does it?

Here is a link to an SPSS file containing 50 students' exam grades (let's say it's the final exam for a Statistics class).

1. Create a table containing the mean and standard deviation for this sample of scores. Now, produce a frequency histogram of the score data. Describe the distribution.
2. Transform each student's score into a Z-score (you can use either method shown in this tutorial). Now, plot the frequency histogram of this Z-score distribution. Compare it to the raw score distribution. How are they the same? How are they different?
3. Imagine you are a student in this class who received a 90 on this exam. However, the Professor has decided to GRADE ON A CURVE (shock! awe!), such that only the top 10% of the class receives an A (this professor only gives whole grades, no minuses or pluses). Calculate the z-score that corresponds to a raw score of 90 on this exam. Will you get an A with this grade? Why or why not?

4.5 JAMOVI

How to do it in JAMOVI

Chapter 5

Lab 5: Fundamentals of Hypothesis Testing

The null hypothesis is never proved or established, but is possibly disproved, in the course of experimentation. Every experiment may be said to exist only to give the facts a chance of disproving the null hypothesis. —R. A. Fisher

5.1 R

From here on, we will be focusing on making sense of data from experiments. In all of this, we use experiments to ask a question about whether one thing causes change (influences) another thing. Then, we look at the data to help us answer that question. In general, we expect to find a difference in our measurement between the conditions of the experimental manipulation. We expect to find a difference when the manipulation works, and causes change in our measure. We expect not to find a difference when the manipulation does not work, and does not cause change.

However, as you well know from reading the textbook, and attending the lectures. Experimental manipulations are not the only thing that can cause change in our measure. Chance alone can cause change. Our measures are usually variable themselves, so they come along with some change in them due to sampling error.

At a minimum, when we conduct an experiment, we want to know **if the change we observed is bigger than the change that can be produced by chance**. Theoretically, random chance could produce most any change we might measure in our experiment. So, there will always be uncertainty about whether our manipulation caused the change, or chance caused the change. But, we can reduce and evaluate that uncertainty. When we do this, we make **inferences** about

what caused change in our experiments. This process is called **statistical inference**. We use **inferential statistics** as tools to help us make these inferences.

In this lab we introduce you to foundational concepts in **statistical inference**. This is also commonly termed **hypothesis testing**. But, for various reasons using that language to describe the process is tied to particular philosophies about doing statistical inference. We use some of that language here, so that you know what it means. But, we also use our own plain language, so you know what the point is, without the statistical jargon.

The textbook describes a few different statistical tests for building your conceptual understanding for statistical inference. In this lab, we work through some of them. In particular, we work through the Crump test, and the Randomization test. We show you how to conduct these tests in R on fake data, and real data.

5.1.1 The Crump Test

The Crump test is described more fully in the textbook here, but you already read that in preparation for this lab, right! I hope you did.

The big idea behind the Crump test is this. You find out what kind of differences between two conditions can be found by chance alone. This shows you what chance can do. Then, you compare what you actually found in one experiment, with the chance distribution, and make an inference about whether or not chance could have produced the difference.

5.1.1.1 Make assumptions about the distribution for your measurement

The first step in conducting the Crump test is to make a guess at the distribution behind your measurement. We will see in the next part how to do this from real data. For now, we just pick a distribution. For example, let's say we are measuring something that comes from a normal distribution with mean = 75 and standard deviation = 5. Perhaps, this is a distribution for how people perform on a particular test. The mean on the test is 75%, with a standard deviation of 5%. We know from last lab that 3 standard deviations away from the mean is pretty unlikely with this distribution. So, for example, most people never score above 90% ($5*3=15$, $75+15 = 90$) on this test.

In this example situation, we might imagine an experiment that was conducted to determine whether manipulation A improves test performance, compared to a control condition where no manipulation took place. Using the Crump test, we can simulate differences that can occur by chance. We are formally simulating the differences that could be obtained between two control conditions, where no manipulation took place.

To, restate our assumptions, we assume a single score for each subject is sampled from:

```
rnorm(n, mean=75, sd=5)
```

5.1.1.2 Make assumptions about N

In the real world, experiments have some number of subjects in each condition, this number is called N. For our simulation we, need to choose the number of subjects that we have. For this demonstration, we choose N = 20 in each condition.

5.1.1.3 Choose the number of simulations to run

We are going to run a fake experiment with no manipulation, and do this many times over (doing it many times over is called **monte carlo simulation**). Each time we will do this:

1. Sample 20 numbers for control group A using `rnorm(20, mean=75, sd=5)`
2. Sample 20 numbers for control group B using `rnorm(20, mean=75, sd=5)`
3. Compute the means for control group A and B
4. Compute the difference between the mean for group A and B
5. Save the differences score in a variable
6. Repeat as many times as we want

If we repeat the simulation 100 times, we will see the differences that can be produced by chance, when given the opportunity 100 times. For example, in a simulation like this, the biggest difference (the maximum value) only happens once. We can find that difference, and then roughly conclude that a difference of that big happens 1 out of 100 times just by chance. That's not a lot.

If we want to be more restrictive, we can make the simulation go to 1,000, or 10,000, or greater. Each time the maximum value will tell us what is the biggest thing chance did 1 out of 1000 times, or 1 out of 10,000 times.

The textbook uses 10,000 times. Let's use 100 times here to keep things simple.

5.1.1.4 Run the simulation

```
library(ggplot2)

# set parameters of simulation

sims_to_run <- 100
sample_n   <- 20
dist_mean  <- 75
dist_sd    <- 5

# run simulation
```

```

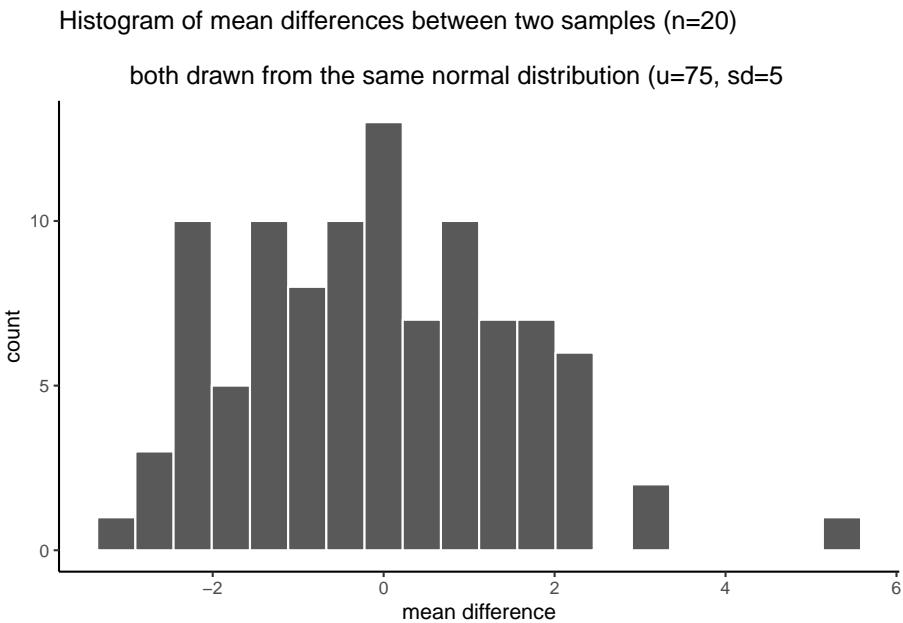
mean_differences <- length(sims_to_run)
for(i in 1:sims_to_run){
  mean_control_A      <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_control_B      <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_differences[i] <- mean_control_A - mean_control_B
}

# plot the distribution of mean difference scores

plot_df <- data.frame(sim=1:sims_to_run,mean_differences)

ggplot(plot_df,aes(x=mean_differences))+
  geom_histogram(bins=20, color="white")+
  theme_classic()+
  ggtitle("Histogram of mean differences between two samples (n=20) \n
           both drawn from the same normal distribution ( $\mu=75$ ,  $sd=5$ )"+
  xlab("mean difference")

```



5.1.1.5 find the range

We can see that chance produces some differences that are non-zero. The histogram shows all the mean differences that were produced by chance. Most of the differences are between -2 and +2, but some of them are bit more negative, or a bit more positive. If we want to know what chance **did** do in this one simulation with 100 runs, then we need to find the range, the minimum

and maximum value. This will tell us the most negative mean difference that chance did produce, and the most positive mean difference that chance did produce. Then, we will also know that chance **did not** produce any larger negative, or larger positive differences, in this simulation.

We use the `min()` and `max()` functions to get the minimum and maximum value.

```
min(mean_differences)

## [1] -3.070447

max(mean_differences)

## [1] 5.419415
```

We now know, that biggest negative difference was -3.07, and the biggest positive difference was 5.419. We also know that any mean difference inside the range **was produced by chance** in our simulation, and any mean difference outside the range **was not produced by chance** in our simulation

5.1.1.6 Make inferences

This part requires you to think about the answers. Let's go through some scenario's.

1. You sample 20 numbers from a normal distribution with $\text{mean} = 75$, and $\text{standard deviation} = 5$. The mean of your sample is 76. Then, you take another sample of the same size, from the same distribution, and the mean of your second sample is 78. The mean difference is +1 (or -1, depending on how you take the difference)
 - a. **Question:** According to the histogram did a mean difference of 1 or -1 occur by chance?
 - b. **Answer:** Yes, it is inside the range
2. Same as above, but the mean of your first sample is 74, and the mean of your second sample is 80, showing a mean difference of 6, or -6.
 - a. **Question:** According to the histogram did a mean difference of 6 or -6 occur by chance?
 - b. **Answer:** No, it is outside the range
3. You run an experiment. Group A receives additional instruction that should make them do better on a test. Group B takes the test, but without the instruction. There are 20 people in each group. You have a pretty good idea that group B's test scores will be coming from a normal distribution with $\text{mean} = 75$, and $\text{standard deviation} = 5$. You know this because you have given the test many times, and this is what the distribution usually looks like. You are making an educated guess. You find that the mean test performance for Group A (with additional instruction) was 76%, and

152 CHAPTER 5. LAB 5: FUNDAMENTALS OF HYPOTHESIS TESTING

the mean test performance for Group B (no additional instruction) was 75%. The mean difference has an absolute value of +1.

- a. **Question #1:** According to the histogram, could chance alone have produced a mean absolute difference of +1?
 - b. **Answer:** Yes, it is inside the range
 - c. **Question #2:** It looks like Group A did better on the test (on average), by 1%, compared to the control group B. Are you willing to believe that your additional instruction **caused the increase in test performance?**
 - d. **Answer:** The answer is up to you. There is no correct answer. It could easily be the case that your additional instruction did not do anything at all, and that the difference in mean test performance was produced by chance. My inference is that I do not know if my instruction did anything, I can't tell its potential influence from chance.
4. Same as 3, except the group mean for A (receiving instruction) is 90%. The group mean for B (no instruction control) is 75%. The absolute mean difference is 15%.
- a. **Question #1:** According to the histogram, could chance alone have produced a mean absolute difference of +15?
 - b. **Answer:** No, it is well outside the range
 - c. **Question #2:** It looks like Group A did better on the test (on average), by 15%, compared to the control group B. Are you willing to believe that your additional instruction **caused the increase in test performance?**
 - d. **Answer:** The answer is up to you. There is no correct answer. You know from the simulation that chance never produced a difference this big, and that producing a difference this big by chance would be like winning the lottery (almost never happens to you). My inference is that I believe chance did not produce the difference, I'm willing to believe that my instructional did cause the difference.

5.1.1.7 Planning your experiment

We've been talking about a hypothetical experiment where an instructor tests whether group A does better (when receiving additional instruction) on a test, compared to a group that does receives no additional instruction and just takes the test.

If this hypothetical instructor wanted to make an experiment, they would get to choose things like how many subjects they will put in each condition. How many subjects should they plan to get?

The number of subjects they plan to get will change what chance can do, and will change the sensitivity of their experiment to detect

differences of various sizes, that are not due to chance.

We can use the simulation process to make informed decisions about how many subjects to recruit for an experiment. This is called **sample-size planning**. There are two goals here. The instructor might have a first goal in mind. They may be only interested in adopting a new method for instruction, if it actually improves test performance beyond more than 1% (compared to control). Differences of less than 1% are just not worth it for the instructor. They want bigger differences, they want to help their students improve more than 1%.

One problem for the instructor, is that they just don't know in advance how good their new teaching materials will be. Some of them will be good and produce bigger differences, and some of them won't. The size of the difference from the manipulation can be unknown. However, this doesn't really matter for planning the experiment. The instructor wants to know that they can **find** or detect any real difference (not due to chance) that is say 2% or bigger. We can use the simulation to figure out roughly (or more exactly, depending on how much we work at it) how many subjects are needed to detect difference of at least 2%.

Notice, from our prior simulation, chance does produce differences of 2% some of the time (given 100 runs). The task now is to re-run the simulation, but use different numbers of subjects to figure out how many subjects are needed to always detect differences of 2%. To be simple about this, we are interested in producing a distribution of mean differences that never produces a mean difference of -2% to + 2% (not once out of 100 times). You can re-run this code, and change N until the min and max are always smaller than -2 to +2.

The code starts out exactly as it was before. You should change the number for **sample_n**. As you make the number bigger, the range (min and max) of the mean differences by chance will get smaller and smaller. Eventually it will be smaller than -2 to +2. When you get it this small, then the N that you used is your answer. Use that many subjects. If you run your experiment with that many subjects, **AND** you find a difference of 2 or greater, then you know that chance does not do this even 1 times out of 100.

```
library(ggplot2)

# set parameters of simulation

sims_to_run <- 100
sample_n   <- 20
dist_mean  <- 75
dist_sd    <- 5

# run simulation

mean_differences <- length(sims_to_run)
```

154 CHAPTER 5. LAB 5: FUNDAMENTALS OF HYPOTHESIS TESTING

```

for(i in 1:sims_to_run){
  mean_control_A <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_control_B <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_differences[i] <- mean_control_A - mean_control_B
}

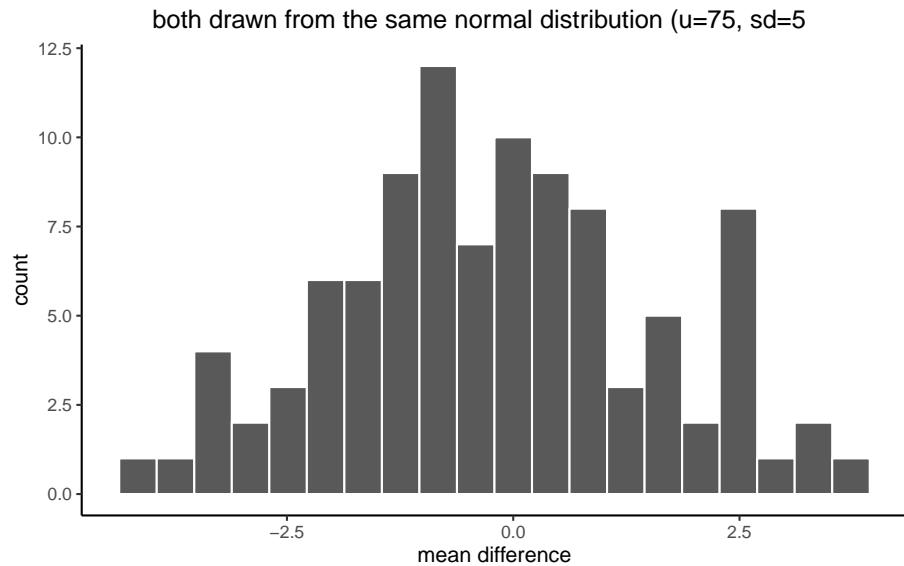
# plot the distribution of mean difference scores

plot_df <- data.frame(sim=1:sims_to_run,mean_differences)

ggplot(plot_df,aes(x=mean_differences))+
  geom_histogram(bins=20, color="white")+
  theme_classic()+
  ggtitle("Histogram of mean differences between two samples (n=20) \n
    both drawn from the same normal distribution (u=75, sd=5")+
  xlab("mean difference")

```

Histogram of mean differences between two samples (n=20)



```
min(mean_differences)
```

```
## [1] -4.316494
max(mean_differences)
```

```
## [1] 3.560732
```

5.1.2 Crumping real data

In this example we look at how you can run a Crump test to evaluate the results in a published paper. The goal of this is to build up your intuitions about whether or not an observed difference could have been caused by chance. We take many liberties, and this is not an exact test of anything. However, we will see how a series of rough, and reasonable assumptions can be made to simulate the results of published experiments, even when exact information is not provided in the paper.

5.1.2.1 Test-enhanced learning

We have already been using an educational example. We've been talking about a manipulation that might be employed to help students learn something better than they otherwise would without the manipulation.

Research in Cognitive Psychology has discovered clear evidence of some teaching practices that really work to enhance memory and comprehension. One of these practices is called test-enhanced learning. Students who study material, and take a quick test (quiz) while they study, do better at remembering the material compared to students who just studied the whole time and did not take a quiz (why do you think we have so many quizzes, in the textbook and for this class? This is why. Prior research shows this will improve your memory for the content, so we are asking you to take the quizzes so that it helps you learn!).

Here is a link to a paper demonstrating the test-enhanced learning effect.

The citation is: Roediger III, H. L., & Karpicke, J. D. (2006). Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological science*, 17(3), 249-255.

5.1.2.2 Brief summary

The subjects learned about some things in two conditions. In one condition (study-study) they studied some things, then studied them again. In the other condition (study-test) they studied some things, then took a quiz about the things rather than studying them one more time.

Everyone received follow up tests to see what they learned and remembered. They came back one week later and took the test. The researchers measured the mean proportion of things remembered in both conditions. They found the study-test condition had a higher mean proportion of remembered idea units than the study-study condition. So, the difference between the mean proportions suggest that taking a quick test after studying was beneficial for remembering the content. The researchers also conducted statistical tests, and they concluded the difference they found was not likely due to chance. Let's apply the Crump test to their findings, and see if we come to the same conclusion about the role of chance.

5.1.2.3 Estimate the parameters of the distribution

To do the Crump test, we need to make assumptions about where the sample data is coming from. Download the paper from the link, then look at Figure 1. We will estimate our distribution by looking at this figure. We will be doing **informed guesstimation** (a real word I just made up).

Look only at the two bars for the 1 week condition.

The mean for the study-study group is about .4. The mean for the study-test group is about .55. The results section reports that the actual mean for the study-study group was .42 (42%) and the mean for the study-test group was .56 (56%). Pretty close to our visual guesstimate.

The data show that the study-test group remembered .14 (14%, or $.56 - .42 = .14$) more idea units than the study-study group.

Estimating the mean

We can imagine for the moment that this difference could have been caused by chance. For example, in this case, both samples would have been drawn from the same distribution. For example, we might say that on average people remember about .49 idea units after a one week delay. I got .49 by averaging the .42 and the .56 together. We can use this as the mean for our distribution in the Crump test.

N

The paper says there were 120 subjects in total, and that different groups of subjects were measured in the three different delay conditions (5 minutes, 2 Days and 1 week). We will assume there were an equal number of subjects in each group. There were 3 groups, $120/3 = 40$, so we assume there were 40 subjects in the 1 week delay group

Estimating Standard Deviation

The paper doesn't directly report the standard deviation for the measurement of proportion idea units. But, we can guesstimate it visually. Look at the little bars with a line on them coming out of each bar. These are called error bars, and they represent the standard error of the mean.

These look like they are about .033 in length. We know the standard error of the mean (SEM) is the standard deviation divided by the square root of N. So, we can infer that the standard deviation is $.033 * \sqrt{40} = .21$.

The paper also reports Cohen's D, which is a measure of effect size using the mean difference divided by the standard deviation. Cohen's D was .83, so the standard deviation must have been $.14/.83 = .168$, which is pretty close to our guesstimate of .21.

5.1.2.4 Findings from the original study

One week after the initial learning conditions, subjects came back and took a retention test to see how much they learned. The mean proportion “idea units” recalled was:

1. study-study : 42% or .42
2. study-test : 56% or .56

The mean difference was $.56 - .42 = .14$ or a whopping 14% improvements. That’s actually pretty big.

What we want to know is whether chance could produce a difference of 14%, or .14, just by itself.

5.1.2.5 Run the simulation

Now we can plug our numbers in to the Crump test simulation.

1. We run the simulation 100 times
2. Our sample N is 40
3. The mean of our distribution is .49
4. The standard deviation of the distribution is .168

```
# set parameters of simulation

sims_to_run <- 100
sample_n   <- 40
dist_mean  <- .49
dist_sd    <- .168

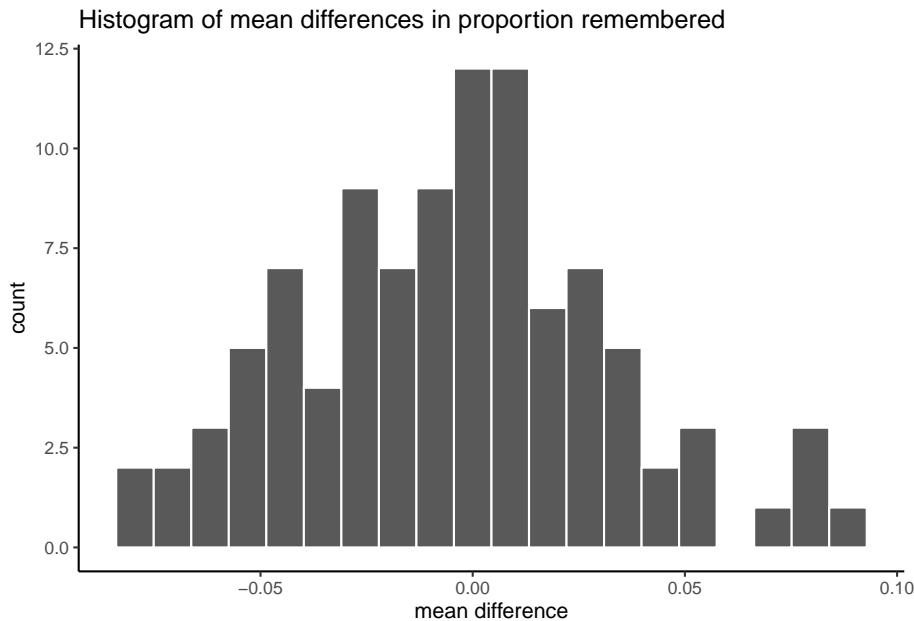
# run simulation

mean_differences <- length(sims_to_run)
for(i in 1:sims_to_run){
  mean_control_A     <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_control_B     <- mean(rnorm(sample_n, dist_mean, dist_sd))
  mean_differences[i] <- mean_control_A - mean_control_B
}

# plot the distribution of mean difference scores

plot_df <- data.frame(sim=1:sims_to_run,mean_differences)

ggplot(plot_df,aes(x=mean_differences))+
  geom_histogram(bins=20, color="white")+
  theme_classic()+
  ggtitle("Histogram of mean differences in proportion remembered")+
  xlab("mean difference")
```



```
min(mean_differences)
## [1] -0.08256789
max(mean_differences)
## [1] 0.08543755
```

According to the simulation, the biggest negative difference was -0.083, and the biggest positive difference was 0.085. We also know that any mean difference inside the range **was produced by chance** in our simulation, and any mean difference outside the range **was not produced by chance** in our simulation.

A difference of .14 or 14% was never produced by chance, it was completely outside the range. Based on this analysis we can be fairly confident that the test-enhanced learning effect was not a fluke, it was not produced by chance. We have evidence to support the claim that testing enhances learning, and because of this we test you while you are learning to enhance your learning while you learn.

5.1.3 The Randomization Test

Unlike the Crump test, which was made to help you understand some basic ideas about how chance can do things, the Randomization test is a well-known, very real, statistical test for making inferences about what chance can do. You will see that it is very similar to the Crump test in many respects. In fact, we might say that the Crump test is really just a randomization test.

You read about the randomization test in the textbook. We won't repeat much about that here, but we will show you how to do one in R.

To briefly remind you, in a randomization test we first obtain some data from an experiment. So we have a sample of numbers for group A, and Group B. We calculate the means for both groups (or other statistic we want to know about), and then see if they are different. We want to know if the difference we found could be produced by chance, so we conduct the randomization test on using the sample data that we have.

The major difference between the Crump test and the Randomization test, is that we make no assumption about the distribution that the sample data came from. We just randomize the sample data. We do:

1. Take all the numbers from group A and B, put them in the same pot. Then randomly take the numbers out and assign them back into A and B. Then, compute the means, and the difference, and save the difference
2. Do this over and over
3. Plot the histogram of the mean differences obtained by shuffling the numbers. This shows you what chance can do.

5.1.3.1 Run the randomization test

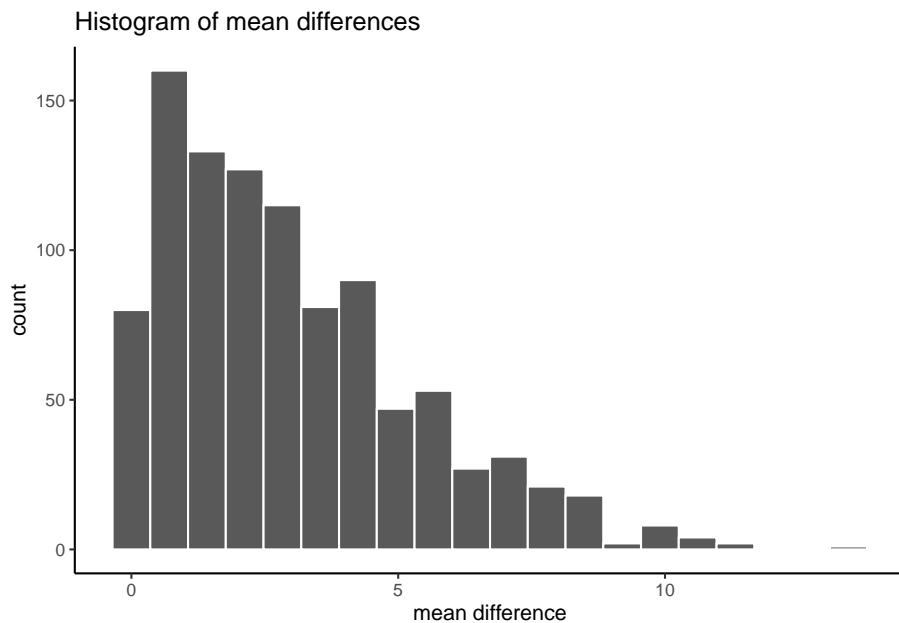
Note, this time when we calculate the mean differences for each new group, we will take the absolute value of the mean difference.

```
# get sample numbers from one experiment
Group_A <- rnorm(20,50,10)
Group_B <- rnorm(20,50,10)

# randomize the numbers, compute mean difference, save the mean difference
mean_differences <- length(1000)
for(i in 1:1000){
  shuffle_numbers <- sample(c(Group_A, Group_B), replace=FALSE)
  new_group_A <- shuffle_numbers[1:20]
  new_group_B <- shuffle_numbers[21:40]
  mean_differences[i] <- abs(mean(new_group_A)-mean(new_group_B))
}

# plot the histogram
plot_df <- data.frame(sim=1:1000,mean_differences)

ggplot(plot_df,aes(x=mean_differences))+
  geom_histogram(bins=20, color="white")+
  theme_classic()+
  ggtitle("Histogram of mean differences")+
  xlab("mean difference")
```



The histogram shows us the kinds of absolute mean difference that happen by chance alone. So, in this data, a mean difference of 10 hardly wouldn't happen very often. A difference between 0 and 2.5 happens fairly often by chance.

5.1.3.2 Decision criteria

When you are determining whether or not chance could have produced a difference in your data, you might want to consider how stringent you want to be about accepting the possibility that chance did something. For example chance could produce a difference of 0 or greater 100% of the time. Or, it produces a difference of 10 or greater, a very small (less than .00001% of the time). How big does the difference need to be, before you will consider the possibility that chance probably didn't cause the difference.

Alpha. Researchers often set what is called an **alpha** criteria. This draws a line in the histogram, and says I will be comfortable assuming that chance did not produce my differences, when chance produces difference less than X percent of the time. This X percent, is the alpha value. For example, it is often set to 5%, or $p \leq .05$.

Where is 5% of the time on our histogram? What mean difference or greater happens less than or equal to 5% of the time.

5.1.3.3 Finding the critical region

1. Take the simulated mean difference scores and order them from smallest to largest

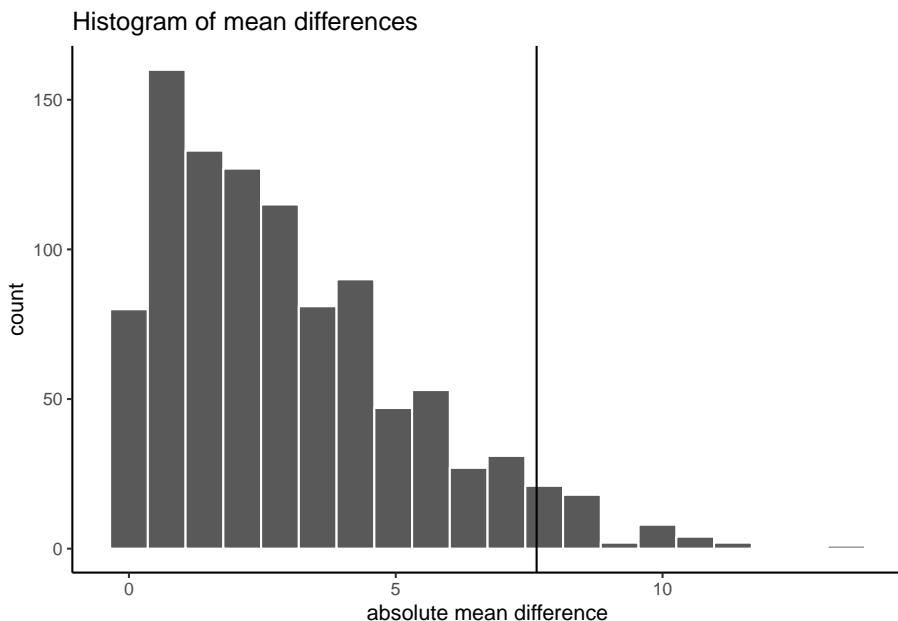
2. We have 1000 numbers, ordered from smallest to largest.
3. The number in position 950 is the alpha location. All of the numbers from position 0 to 950, reflect 95% of the numbers. What is left over is 5% of the numbers.
4. Find the alpha cut-off, and plot it on the histogram

```
ordered_differences <- sort(mean_differences) # sort
alpha_cutoff <- ordered_differences[950] # pick 950th number
alpha_cutoff

## [1] 7.639369

# add to histogram using vline

ggplot(plot_df,aes(x=mean_differences))+ 
  geom_histogram(bins=20, color="white")+
  geom_vline(xintercept=alpha_cutoff)+
  theme_classic()+
  ggtitle("Histogram of mean differences")+
  xlab("absolute mean difference")
```



OK, so our alpha criterion or cutoff is located at 7.64 on the x-axis. This shows us that mean differences of 7.64 or larger happen only 5% of the time by chance.

You could use this information to make an inference about whether or not chance produced the difference in your experiments

1. When the mean difference is 7.64 or larger, you might conclude that chance

did not produce the difference

2. When the mean difference is less than 7.64, you might conclude that chance could have produced the mean difference.

It's up to you to set your alpha criterion. In practice it is often set at $p=.05$. But, you could be more conservative and set it to $p=.01$. Or more liberal and set it to $p=.1$. It's up to you.

5.1.4 Generalization Exercise

Complete the generalization exercise described in your R Markdown document for this lab.

1. Consider taking measurements from a normal distribution with mean = 100 and standard deviation = 25. You will have 10 subjects in two conditions (20 subject total). You will take 1 measurement (sample 1 score) of each subject.

Use the process for simulating the Crump test. Pretend that there are now differences between the conditions. Run a Crump test simulation 100 times.

- a. Report the maximum mean difference in the simulation
- b. Report the minimum mean difference in the simulation
- c. Report a mean difference that, according to the simulation, was not observed by chance
- d. Report a mean difference that, according to the simulation was observed by chance
- e. What is the smallest mean difference that, if you found this difference in an experiment, you would be willing to entertain the possibility that the difference was unlikely to be due to chance (note this is a subjective question, give your subjective answer)

5.1.5 Writing assignment

Complete the writing assignment described in your R Markdown document for this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

Imagine you conduct an experiment. There is one independent variable with two levels, representing the manipulation. There is one dependent variable representing the measurement. Critically, the measurement has variability, so all of the samples will not be the same because of variability. The researcher is interested in whether the manipulation causes a change in the dependent measure. Explain how random chance, or sampling error, could produce a difference in the dependent measure even if the manipulation had no causal role.

- a. Explain how the sampling distribution of the mean differences from the Crump or Randomization test relates to the concept that chance can produce differences (0.5 points)
- b. Imagining a sampling distribution of the mean differences from a Crump test simulation, which values in the distribution are most likely to be produced by chance (0.5 points)
- c. From above, which values are least likely to be produced by chance (0.5 points)
- d. If you obtained a mean difference that was very large and it fell outside the range of the sampling distribution of the mean differences from a Crump test, what would you conclude about the possibility that chance produced your difference. Explain your decision. (0.5 points)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

5.2 Excel

How to do it in Excel

5.3 SPSS

In this lab, we will use SPSS to:

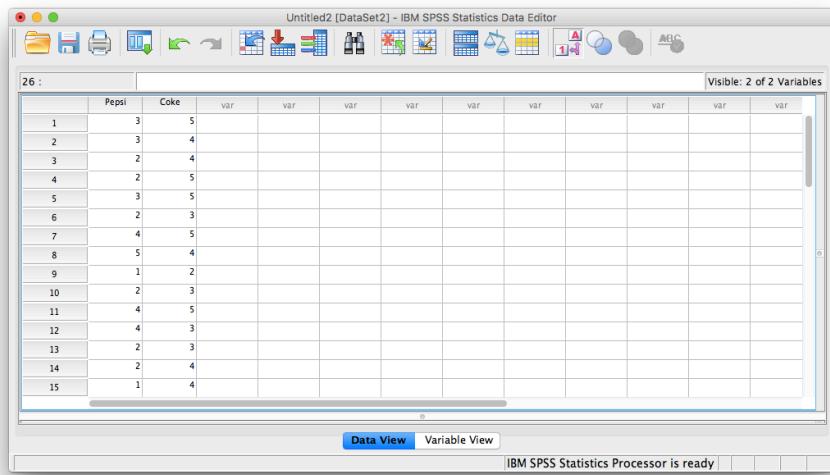
1. Calculate difference scores between pairs of measures
2. Conduct a sign test
3. Entering data for sign test problems

5.3.1 Experiment Background

This is a fictitious experiment based on the infamous Coke vs. Pepsi Taste Test Challenge. 20 blindfolded participants were presented with two cups of soda (Coke and Pepsi). Presentation sequence was randomly determined, and participants were required to rate the taste of the soda on a scale from 1-5.

5.3.2 Calculate difference scores between pairs of measures

First, let's open the relevant data file; Here is the link; it is called PepsiCoke.sav. Open this file in SPSS. Your data should look like this:

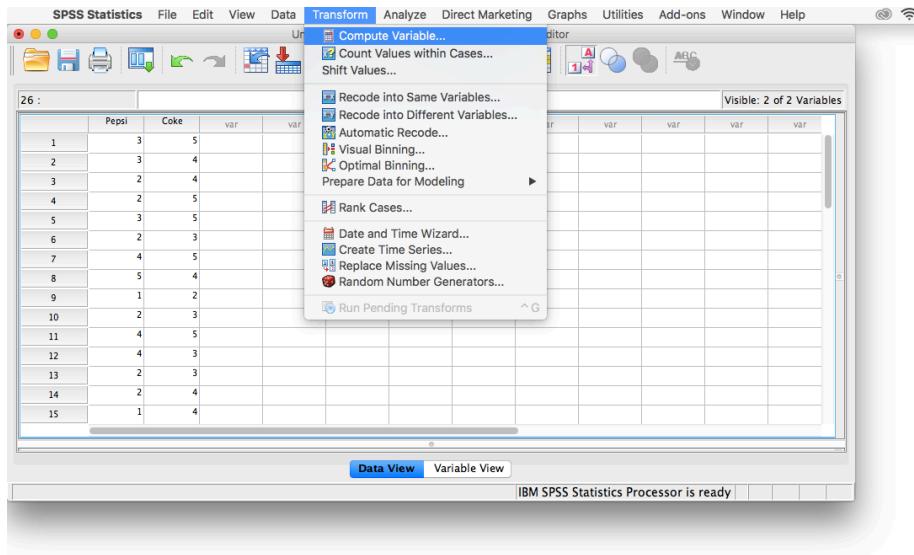


The screenshot shows the IBM SPSS Statistics Data Editor window titled "Untitled2 (DataSet2) - IBM SPSS Statistics Data Editor". The window displays a data table with 15 rows and 12 columns. The first two columns are labeled "Pepsi" and "Coke", while the remaining ten columns are labeled "var". The data in the "Pepsi" and "Coke" columns are as follows:

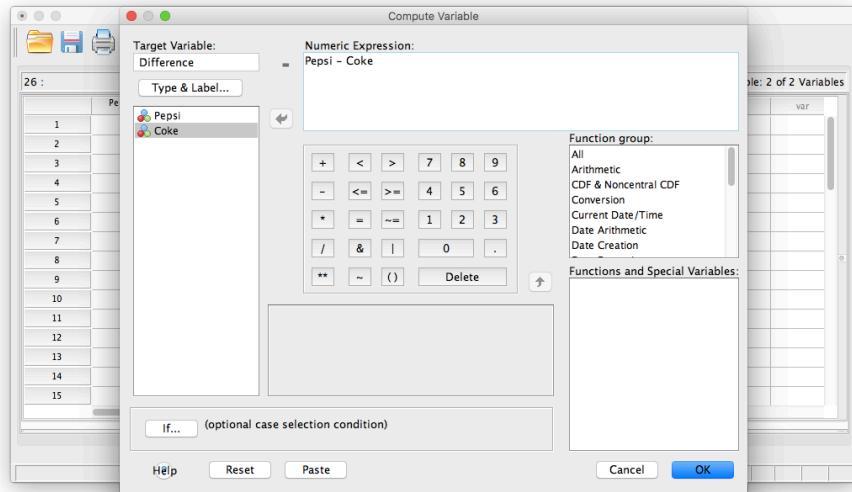
	Pepsi	Coke	var								
1	3	5									
2	3	4									
3	2	4									
4	2	5									
5	3	5									
6	2	3									
7	4	5									
8	5	4									
9	1	2									
10	2	3									
11	4	5									
12	4	3									
13	2	3									
14	2	4									
15	1	4									

First, let's calculate, for each participant, the difference between the rating they gave for Coke vs. Pepsi. We will use SPSS to create a new variable (in a new column) containing only difference scores.

To begin, go to Transform, then Compute Variable...



A calculation window will open. First, give your new variable a name in the field “Target Variable”. I am going to call it **Difference**. Then, in the field labeled “Numeric Expression”, we must convey that we want SPSS to take the **Pepsi** score minus the **Coke** score (the order in which you do this really doesn’t matter, as long as you know the order you used). Take your **Pepsi** variable from the list on the left and move it into the “Numeric Expression” field using the arrow. Then, type (or you can use the keypad in the window to insert) a minus sign, followed by moving the **Coke** variable into the field. The result in this field should look like “**Pepsi - Coke**”.



Click OK. An SPSS output window will pop up to confirm that you have created a new variable, but you can ignore or x it out. Go back to the data spreadsheet and you will find there is a new variable, labeled **Difference** that contains the difference scores for each person. Pay attention to the sign on each difference score. Since we used the calculation “Pepsi - Coke”, anyone who has a positive difference score preferred the Pepsi better, while anyone with a negative difference score preferred the Coke.

Untitled2 [DataSet2] - IBM SPSS Statistics Data Editor

Visible: 3 of 3 Variables

	Pepsi	Coke	Difference	var							
1	3	5	-2.00								
2	3	4	-1.00								
3	2	4	-2.00								
4	2	5	-3.00								
5	3	5	-2.00								
6	2	3	-1.00								
7	4	5	-1.00								
8	5	4	1.00								
9	1	2	-1.00								
10	2	3	-1.00								
11	4	5	-1.00								
12	4	3	1.00								
13	2	3	-1.00								
14	2	4	-2.00								
15	1	4	-3.00								
	-	-	---								

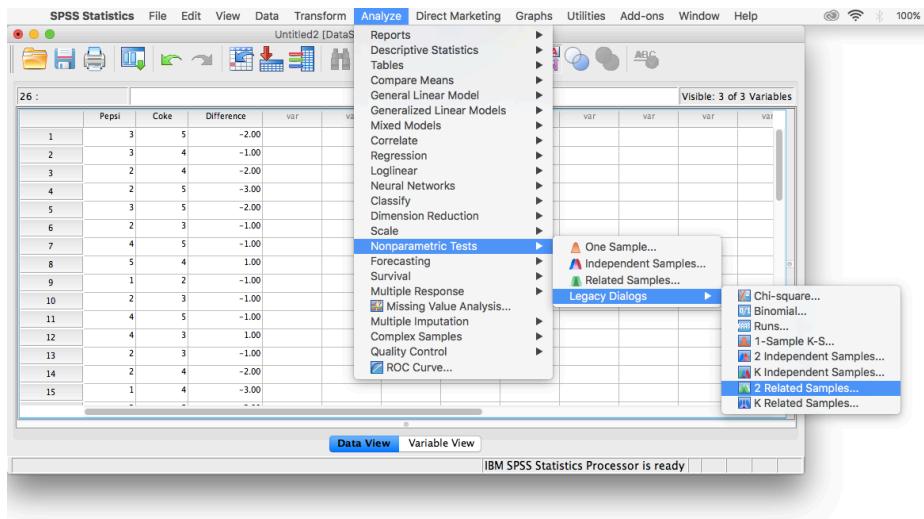
Data View Variable View

IBM SPSS Statistics Processor is ready

5.3.3 Conduct a sign test

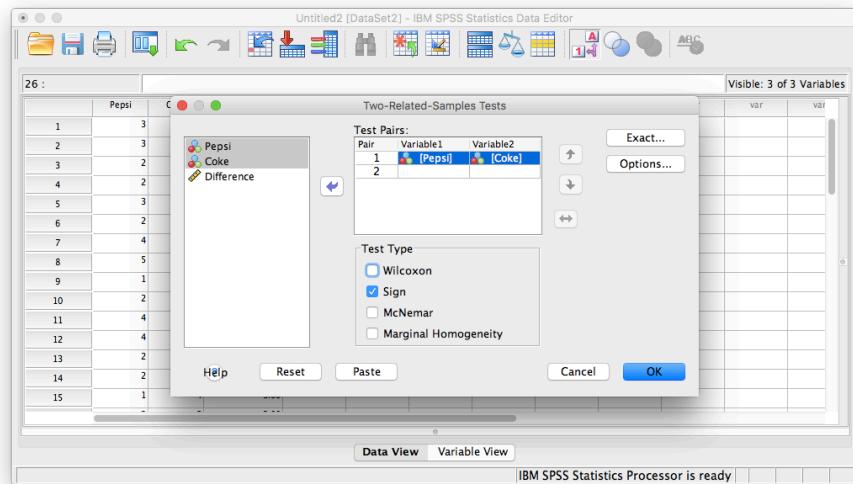
Our goal here is not simply to calculate difference scores. We really want to know if there were more instances of preference for one soda over the other. Since the sign tells us who liked Pepsi better or who liked Coke better, comparing the number of pluses to minuses will tell us which soda wins out (and is also the reason this test is called a Sign test).

To conduct the sign test, go to Analyze, then Nonparametric tests, then Legacy Dialogs, then 2 Related Samples...



In the window that appears, move **Pepsi** and **Coke** from the list on the left to the field labeled “Test Pairs”. Make sure that you uncheck “Wilcoxon” (which is set as the default Test Type), and instead choose Sign Test.

168 CHAPTER 5. LAB 5: FUNDAMENTALS OF HYPOTHESIS TESTING



Then click OK. SPSS will produce two output tables:

NPar Tests

[DataSet2]

Sign Test

Frequencies

		N
Coke - Pepsi	Negative Differences ^a	4
	Positive Differences ^b	16
	Ties ^c	0
	Total	20

- a. Coke < Pepsi
- b. Coke > Pepsi
- c. Coke = Pepsi

Test Statistics^a

	Coke - Pepsi
Exact Sig. (2-tailed)	.012 ^b

- a. Sign Test
- b. Binomial distribution used.

The first table tells you how many difference scores were negative (look at the notes below the table):

- a. Negative differences refers to when Coke < Pepsi, indicating a preference for Pepsi
- b. Positive differences refers to when Pepsi < Coke, indicating a preference for Coke

170 CHAPTER 5. LAB 5: FUNDAMENTALS OF HYPOTHESIS TESTING

- c. Ties refers to an instance when **Pepsi** and **Coke** are given the same rating.
We did not have any of these in our data.

Nota bene: Do not get confused by this table. The reason it says Coke - Pepsi on the left-hand side is because SPSS automatically arranges the variable names alphabetically to label the table. This does not matter for or affect our calculations.

So, looking at this table, it's clear that more people preferred Coke (there are 16 positive differences), and much fewer people preferred Pepsi (there are only 4 negative differences). The question is, is this difference (16 vs. 4) large enough to be statistically significant. The table labeled "Test Statistics" answers this question.

The binomial distribution is used to obtain a p-value (referred to in SPSS output as Sig.) for this exact outcome. The p-value listed in the table is .012. Since this number is less than the commonly used alpha level (.05), we can say that this difference is indeed significant. People significantly prefer Coke over Pepsi in the Taste Test Challenge.

(Neither this lab manual nor any of its authors are affiliated with, funded, or in any way associated with the Coca Cola Company)

5.3.4 Entering data for sign test problems

What if you aren't given a set of paired scores? What if your professor asks you to perform a sign test on a single outcome statement? For example: Test the hypothesis that a coin is weighted if, out of 15 flips of the coin, it lands on tails 13 times.

Notice that in this problem, there are no pairs of scores. We're essentially only given the number of signs: out of 15 flips, the coin landed on tails 13 times (you can think of this as 13 pluses) and on heads obviously twice (you can think of this as 2 minuses).

Since SPSS needs paired input to conduct the sign test, you can simply create two columns: one for **Tails**, and the other for **Heads**. Since you know that 13 of the flips were tails, treat each row as a flip. For 13 of the rows, place a 0 in the **Heads** column and a 1 in the **Tails** column. For the remaining two rows, place a 1 in the **Heads** column and a 0 in the **Tails** column. Your data should look like this:

	Tails	Heads	var											
1	0	1												
2	0	1												
3	1	0												
4	1	0												
5	1	0												
6	1	0												
7	1	0												
8	1	0												
9	1	0												
10	1	0												
11	1	0												
12	1	0												
13	1	0												
14	1	0												
15	1	0												

From here, you can conduct a sign test as described above.

5.3.5 Practice Problems

-
1. In this lab, you will be conducting a sign test to determine whether a coin is weighted. Individually or in small groups, take a coin out of your pocket. If working in groups, use only one coin for the whole group. Now, flip the coin 25 times. Write down how many heads and tails you observed.
 2. Enter this into your SPSS spreadsheet and run a sign test using an alpha level of .05. What is your result?
 3. Have students in the class (or groups) announce their results as well. Did anyone have a trick coin? Do you think some of the coins used were actually weighted? Why or why not?

5.4 JAMOVI

How to do it in JAMOVI

Chapter 6

Lab 6: t-Test (one-sample, paired sample)

Any experiment may be regarded as forming an individual of a ‘population’ of experiments which might be performed under the same conditions. A series of experiments is a sample drawn from this population. —William Sealy Gossett

This lab is modified and extended from Open Stats Labs. Thanks to Open Stats Labs (Dr. Kevin P. McIntyre) for their fantastic work.

6.1 Does Music Convey Social Information to Infants?

This lab activity uses the open data from Experiment 1 of Mehr, Song, and Spelke (2016) to teach one-sample and paired samples t-tests. Results of the activity provided below should exactly reproduce the results described in the paper.

6.1.1 STUDY DESCRIPTION

Parents often sing to their children and, even as infants, children listen to and look at their parents while they are singing. Research by Mehr, Song, and Spelke (2016) sought to explore the psychological function that music has for parents and infants, by examining the hypothesis that particular melodies convey important social information to infants. Specifically, melodies convey information about social affiliation.

The authors argue that melodies are shared within social groups. Whereas children growing up in one culture may be exposed to certain songs as infants (e.g., “Rock-a-bye Baby”), children growing up in other cultures (or even other

groups within a culture) may be exposed to different songs. Thus, when a novel person (someone who the infant has never seen before) sings a familiar song, it may signal to the infant that this new person is a member of their social group.

To test this hypothesis, the researchers recruited 32 infants and their parents to complete an experiment. During their first visit to the lab, the parents were taught a new lullaby (one that neither they nor their infants had heard before). The experimenters asked the parents to sing the new lullaby to their child every day for the next 1-2 weeks.

Following this 1-2 week exposure period, the parents and their infant returned to the lab to complete the experimental portion of the study. Infants were first shown a screen with side-by-side videos of two unfamiliar people, each of whom were silently smiling and looking at the infant. The researchers recorded the looking behavior (or gaze) of the infants during this ‘baseline’ phase. Next, one by one, the two unfamiliar people on the screen sang either the lullaby that the parents learned or a different lullaby (that had the same lyrics and rhythm, but a different melody). Finally, the infants saw the same silent video used at baseline, and the researchers again recorded the looking behavior of the infants during this ‘test’ phase. For more details on the experiment’s methods, please refer to Mehr et al. (2016) Experiment 1.

6.2 Lab skills learned

1. Conducting a one-sample t-test
2. Conducting a two-sample t-test
3. Plotting the data
4. Discussing inferences and limitations

6.3 Important Stuff

- citation: Mehr, S. A., Song, L. A., & Spelke, E. S. (2016). For 5-month-old infants, melodies are social. *Psychological Science*, 27, 486-501.
- Link to .pdf of article
- Data in .csv format
- Data in SPSS format

6.4 R

6.4.1 Loading the data

The first thing to do is download the .csv formatted data file, using the link above, or just click here. It turns out there are lots of ways to load .csv files into R.

1. Load the `data.table` library. Then use the `fread` function and supply the web address to the file. Just like this. No downloading required.

```
library(data.table)
all_data <- fread("https://raw.githubusercontent.com/CrumpLab/statisticsLab/master/data/MehrSongSpelke2016.csv")
```

2. Or, if you downloaded the .csv file. Then you can use `fread`, but you need to point it to the correct file location. The file location in this next example will not work for you, because the file is on my computer.

```
library(data.table)
all_data <- fread("data/MehrSongSpelke2016.csv")
```

6.4.2 Inspect the data frame

When you have loaded data it's always a good idea to check out what it looks like. You can look at all of the data in the environment tab on the top right hand corner. The data should be in a variable called `all_data`. Clicking on `all_data` will load it into a viewer, and you can scroll around. This can be helpful to see things. But, there is so much data, can be hard to know what you are looking for.

6.4.2.1 `summarytools`

The `summarytools` packages give a quick way to summarize all of the data in a data frame. Here's how. When you run this code you will see the summary in the viewer on the bottom right hand side. There's a little browser button (arrow on top of little window) that you can click to expand and see the whole thing in a browser.

```
library(summarytools)
view(dfSummary(all_data))
```

6.4.3 Get the data for Experiment one

The data contains all of the measurements from all five experiments in the paper. By searching through the `all_data` data frame, you should look for the variables that code for each experiment. For example, the third column is called `exp1`, which stands for experiment 1. Notice that it contains a series of 1s. If you keep scrolling down, the 1s stop. These 1s identify the rows associated with the data for Experiment 1. We only want to analyse that data. So, we need to filter our data, and put only those rows into a new variable. We do this with the `dplyr` library, using the `filter` function.

```
library(dplyr)
experiment_one <- all_data %>% filter(exp1==1)
```

Now if you look at the new variable `experiment_one`, there are only 32 rows of data. Much less than before. Now we have the data from experiment 1.

6.4.4 Baseline phase: Conduct a one sample t-test

You first want to show that infants' looking behavior did not differ from chance during the baseline trial. The baseline trial was 16 seconds long. During the baseline, infants watched a video of two unfamiliar people, one of the left and one on the right. There was no sound during the baseline. Both of the actors in the video smiled directly at the infant.

The important question was to determine whether the infant looked more or less to either person. If they showed no preference, the infant should look at both people about 50% of the time. How could we determine whether the infant looked at both people about 50% of the time?

The `experiment_one` data frame has a column called `Baseline_Proportion_Gaze_to_Singer`. All of these values show how the proportion of time that the infant looked to the person who would later sing the familiar song to them. If the average of these proportion is .5 across the infants, then we would have some evidence that the infants were not biased at the beginning of the experiment. However, if the infants on average had a bias toward the singer, then the average proportion of the looking time should be different than .5.

Using a one-sample t-test, we can test the hypothesis that our sample mean for the `Baseline_Proportion_Gaze_to_Singer` was not different from .5.

To do this in R, we just need to isolate the column of data called `Baseline_Proportion_Gaze_to_Singer`. We will do this using the `$` operator. The `$` operator is placed after any data frame variable, and allows you to select a column of the data. The next bit of code will select the column of data we want, and put it into a new variable called `Baseline`. Note, if you type `exp1$` then R-Studio should automatically bring up all the columns you can choose from.

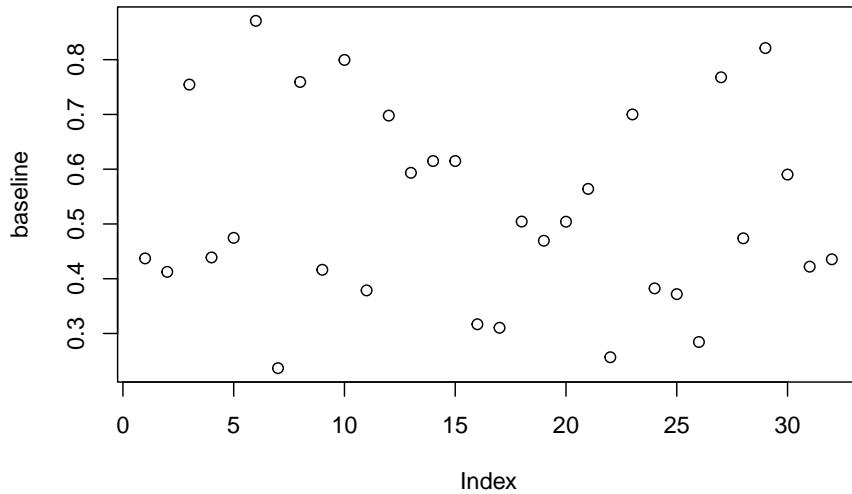
```
baseline <- experiment_one$Baseline_Proportion_Gaze_to_Singer
```

6.4.4.1 Look at the numbers

Question: Why is it important to look at your numbers? What could happen if you didn't?

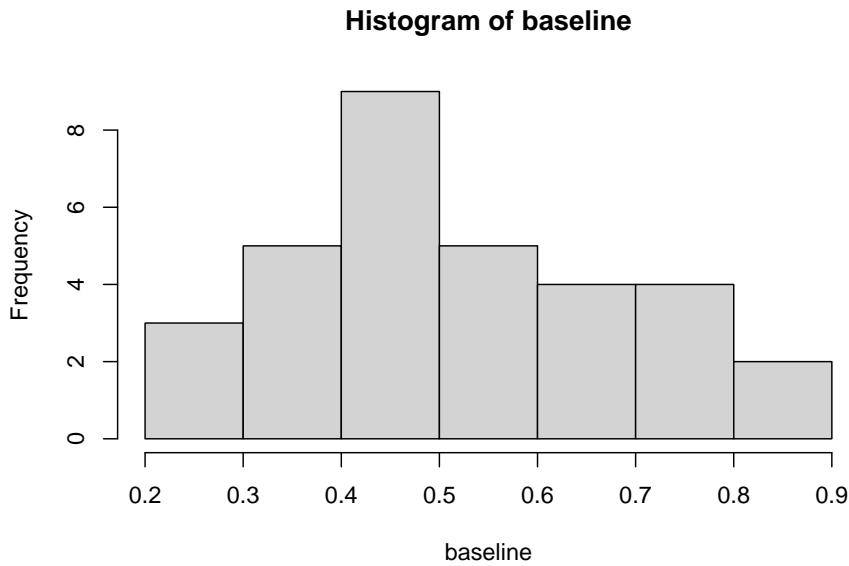
OK, we could just do the t-test right away, it's really easy, only one line of code. But, we haven't even looked at the numbers yet. Let's at least do that. First, we'll just use `plot`. It will show every data point for each infant as a dot.

```
plot(baseline)
```



That's helpful, we see that the dots are all over the place. Let's do a histogram, so we can get a better sense of the frequency of different proportions.

```
hist(baseline)
```



6.4.4.2 Look at the descriptives

Let's get the mean and standard deviation of the sample

```
mean(baseline)
## [1] 0.5210967
sd(baseline)
## [1] 0.1769651
```

OK, so just looking at the mean, we see the proportion is close to .5 (it's .521). And, we see there is a healthy amount of variance (the dots were all over the place), as the standard deviation was about .176.

Question: Based on the means and standard deviations can you make an educated guess about what the t and p values might be? Learn how to do this and you will be improving your data-sense.

Now, before we run the t-test, what do you think is going to happen? We are going to get a t-value, and an associated p-value. If you can make a guess at what those numbers would be right now in your head, and those end up being pretty close to the ones we will see in a moment, then you should pat yourself on the back. You have learned how to have intuitions about the data. As I am writing this I will tell you that 1) I can't remember what the t and p was from the paper, and I haven't done the test yet, so I don't know the answer. So, I am allowed to guess what the answer will be. Here are my guesses $t(31) = 0.2$, $p = .95$. The numbers in the brackets are degrees of freedom, which we know are 31 ($df = n-1 = 32-1 = 31$). More important than the specific numbers I am guessing (which will probably be wrong), I am guessing that the p-value will be pretty large, it will not be less than .05, which the author's set as their alpha rate. So, I am guessing we will not reject the hypothesis that .52 is different from .5.

Let's do the t-test and see what happens.

6.4.4.3 Conduct t.test

```
t.test(baseline, mu=.5)
##
##  One Sample t-test
##
## data: baseline
## t = 0.67438, df = 31, p-value = 0.5051
## alternative hypothesis: true mean is not equal to 0.5
## 95 percent confidence interval:
##  0.4572940 0.5848994
## sample estimates:
## mean of x
```

```
## 0.5210967
```

Question: Why was the baseline condition important for the experiment? What does performance in this condition tell us?

So, there we have it. We did a one-sample t-test. Here's how you would report it, $t(31) = .67$, $p = .505$. Or, we might say something like:

During the baseline condition, the mean proportion looking time toward the singer was .52, and was not significantly different from .5, according to a one-sample test, $t(31) = .67$, $p = .505$.

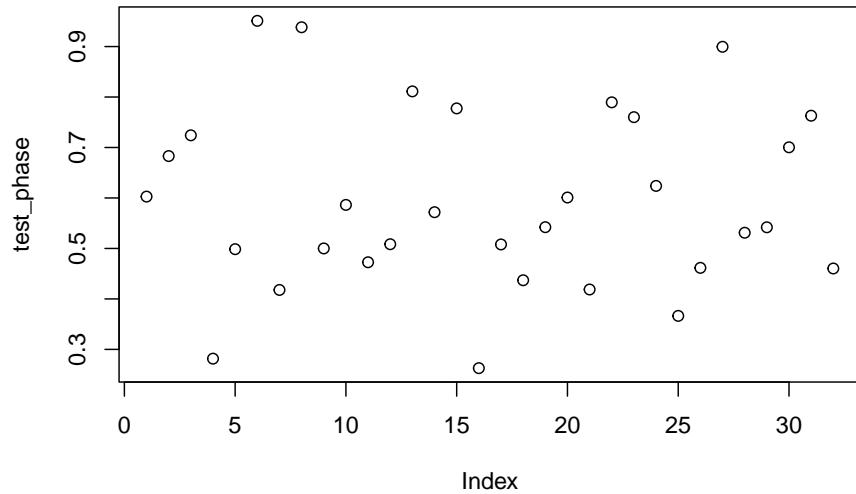
You should take the time to check this result, and see if it is the same one that was reported in the paper.

6.4.5 Test phase

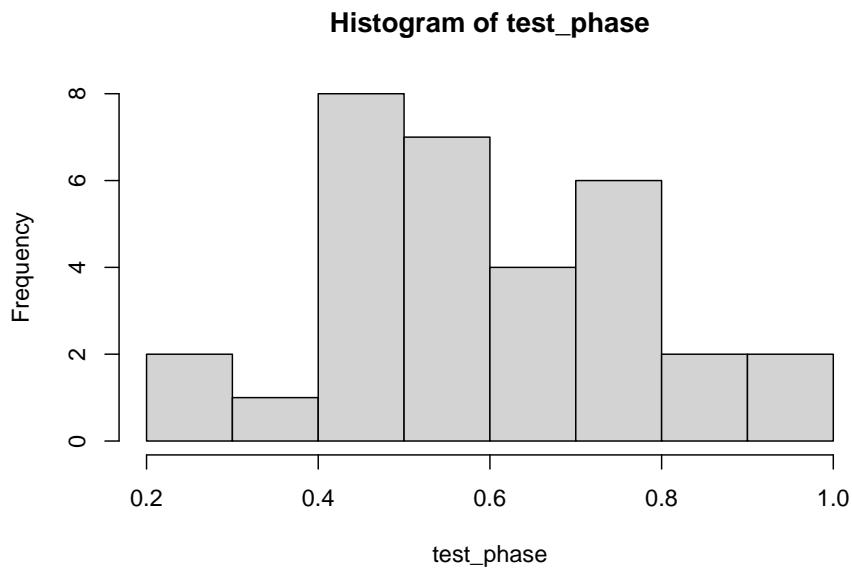
Remember how the experiment went. Infants watched silent video recordings of two women (Baseline). Then each person sang a song, one was familiar to the infant (their parents sung the song to them many times), and one was unfamiliar (singing phase). After the singing phase, the infants watched the silent video of the two singers again (test phase). The critical question was whether the infants would look more to the person who sang the familiar song compared to the person who sang the unfamiliar song. If the infants did this, they should look more than 50% of the time to the singer who sang the familiar song. We have the data, we can do another one sample t-test to find out. We can re-use all the code we already wrote to do this. I'll put it all in one place. If we run all of this code, we will see all of the things we want to see.

We only need to make two changes. We will change `experiment_one$Baseline_Proportion_Gaze_to_Singer` to `experiment_one$Test_Proportion_Gaze_to_Singer`, because that column has the test phase data. And, instead of putting the data into the variable `baseline`. We will make a new variable called `test_phase` to store the data.

```
test_phase <- experiment_one$Test_Proportion_Gaze_to_Singer
plot(test_phase)
```



```
hist(test_phase)
```



```
mean(test_phase)
```

```
## [1] 0.5934913
```

```
sd(test_phase)

## [1] 0.1786884

t.test(test_phase, mu = .5)

##
## One Sample t-test
##
## data: test_phase
## t = 2.9597, df = 31, p-value = 0.005856
## alternative hypothesis: true mean is not equal to 0.5
## 95 percent confidence interval:
## 0.5290672 0.6579153
## sample estimates:
## mean of x
## 0.5934913
```

Question: Why was the test condition important for the experiment? What does performance in this condition tell us?

Alright. What did we find? You should take a stab at writing down what we found. You can use the same kind of language that I used from the first one sample-test. You should state the mean proportion, the t-value, the dfs, and the p-value. You should be able to answer the question, did the infants look longer at the singer who sang the familiar song? And, did they look longer than would be consist with chance at 50%.

6.4.6 Paired-samples t-test

The paired samples t-test is easy to do. We've already made two variables called `baseline`, and `test_phase`. These contain each of the infants looking time proportions to the singer for both parts of the experiment. We can see if the difference between them was likely or unlikely due to chance by running a paired samples t-test. We do it like this in one line:

```
t.test(test_phase, baseline, paired=TRUE, var.equal=TRUE)

##
## Paired t-test
##
## data: test_phase and baseline
## t = 2.4164, df = 31, p-value = 0.02175
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
## 0.01129217 0.13349698
## sample estimates:
## mean difference
```

```
##          0.07239458
```

Question: Why was the paired samples t-test necessary if we already did two one sample t-test? What new question is the paired samples t-test asking?

I'll leave it to you to interpret these values, and to see if they are the same as the ones in the paper. Based on these values what do you conclude? Is there a difference between the mean proportion looking times for the baseline and testing phase?

6.4.6.1 Relationship between one-sample and paired sample t-test

Question: Why is it that a paired samples t-test can be the same as the one sample t-test? What do you have to do the data in the paired samples t-test in order to conduct a one-sample t-test that would give you the same result?

We've discussed in the textbook that the one-sample and paired sample t-test are related, they can be the same test. The one-sample test whether a sample mean is different from some particular mean. The paired sample t-test, is to determine whether one sample mean is different from another sample mean. If you take the scores for each variable in a paired samples t-test, and subtract them from one another, then you have one list of difference scores. Then, you could use a one sample t-test to test whether these difference scores are different from 0. It turns out you get the same answer from a paired sample t-test testing the difference between two sample means, and the one sample t-test testing whether the mean difference of the difference scores between the samples are different from 0. We can show this in r easily like this:

```
t.test(test_phase, baseline, paired=TRUE, var.equal=TRUE)

##
##  Paired t-test
##
## data: test_phase and baseline
## t = 2.4164, df = 31, p-value = 0.02175
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  0.01129217 0.13349698
## sample estimates:
## mean difference
##          0.07239458
difference_scores<-test_phase-baseline
t.test(difference_scores, mu=0)

##
##  One Sample t-test
##
## data: difference_scores
```

```
## t = 2.4164, df = 31, p-value = 0.02175
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.01129217 0.13349698
## sample estimates:
## mean of x
## 0.07239458
```

6.4.6.2 Usefulness of difference scores

OK fine, the paired samples t-test can be a one sample t-test if you use the difference scores. This might just seem like a mildly useful factoid you can use for stats trivia games (which no one plays). The point of drawing your attention to the relationship, is to get you to focus on the difference scores. These are what we are actually interested in.

Let's use the difference scores to one more useful thing. Sometime the results of a t-test aren't intuitively obvious. By the t-test we found out that a small difference between the test phase and baseline was not likely produced by chance. How does this relate to the research question about infants using familiar songs as cues for being social? Let's ask a very simple question. How many infants actually showed the bias? How many infants out of 32 looked longer at the singer who sang the familiar song during test, compared to during baseline.

We can determine this by calculating the difference scores. Then, asking how many of them were greater than zero:

```
difference_scores <- test_phase-baseline
length(difference_scores[difference_scores>0])
```

```
## [1] 22
```

So, 22 out of 32 infants showed the effect. To put that in terms of probability, 68.75% of infants showed the effect. These odds and percentages give us another way to appreciate how strong the effect is. It wasn't strong enough for all infants to show it.

6.4.7 Graphing the findings

It is often useful to graph the results of our analysis. We have already looked at dot plots and histograms of the individual samples. But, we also conducted some t-tests on the means of the baseline and test_phase samples. One of the major questions was whether these means are different. Now, we will make a graph that shows the means for each condition. Actually, we will make a few different graphs, so that you can think about what kinds of graphs are most helpful. There are two major important things to show: 1) the sample means, and 2) a visual aid for statistical inference showing whether the results were likely due to chance.

We will use the `ggplot2` package to make our graphs. Remember, there are two steps to take when using `ggplot2`. 1) put your data in a long form data frame, where each measure has one row, and 2) define the layers of the `ggplot`.

6.4.7.1 Make the dataframe for plotting

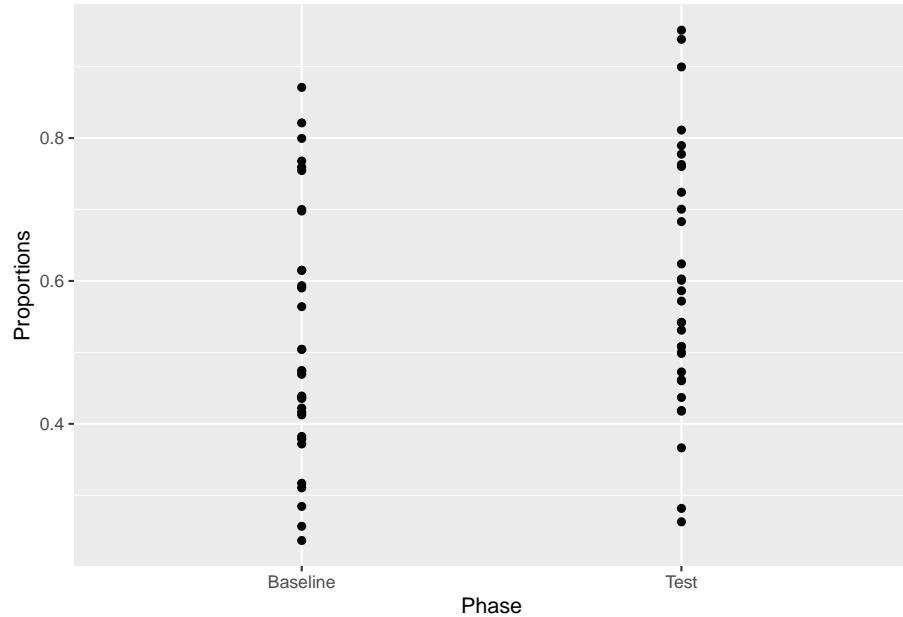
To start we will need 2 columns. One column will code the experimental phase, Baseline or Test. There are 32 observations in each phase, so we want the word `Baseline` to appear 32 times, followed by the word `Test` 32 times. Then we want a single column with each of the proportions for each infant.

```
Phase <- rep(c("Baseline", "Test"), each = 32)
Proportions <- c(baseline,test_phase)
plot_df <- data.frame(Phase,Proportions)
```

6.4.7.2 Dot plot of raw scores

This shows every scores value on the y-axis, split by the baseline and test groups. If you just looked at this, you might not think the test phase was different from the baseline phase. Still very useful to see the spread of individual scores. Supports the intuition that the scores are still kind of in a similar ballpark.

```
library(ggplot2)
ggplot(plot_df, aes(x=Phase, y=Proportions))+
  geom_point()
```



6.4.7.3 Dot plot with means and raw scores

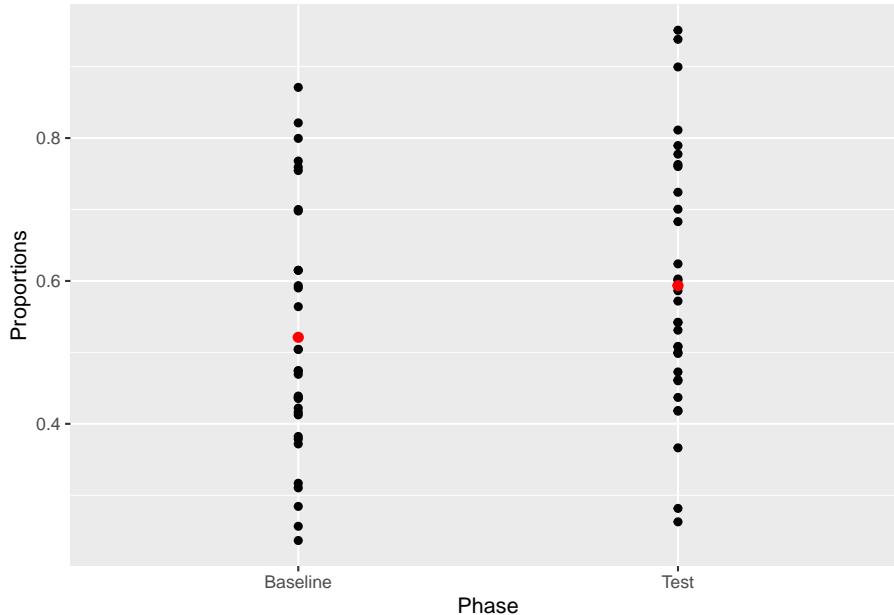
Question: What kinds of inferences about the role of chance in producing the difference between the means can we make from this graph? What is missing?

ggplot2 is great because it let's us add different layers on top of an existing plot. It would be good to see where the mean values for each sample lie on top of the sample scores. We can do this. But, to do it, we need to supply ggplot with another data frame, one that contains the means for each phase in long form. There are two phases, and two means, so we will make a rather small data.frame. It will have two columns, a Phase column, and Mean_value column. There will only be two rows in the data frame, one for the mean for each phase.

To make the smaller data frame for the means we will use the `aggregate` function. This allows us to find the means for each phase from the `plot_df` data frame. It also automatically returns the data frame we are looking for.

```
mean_df <- aggregate(Proportions ~ Phase, plot_df, mean)

ggplot(plot_df, aes(x=Phase, y=Proportions))+
  geom_point()+
  geom_point(data=mean_df, color="Red", size=2)
```



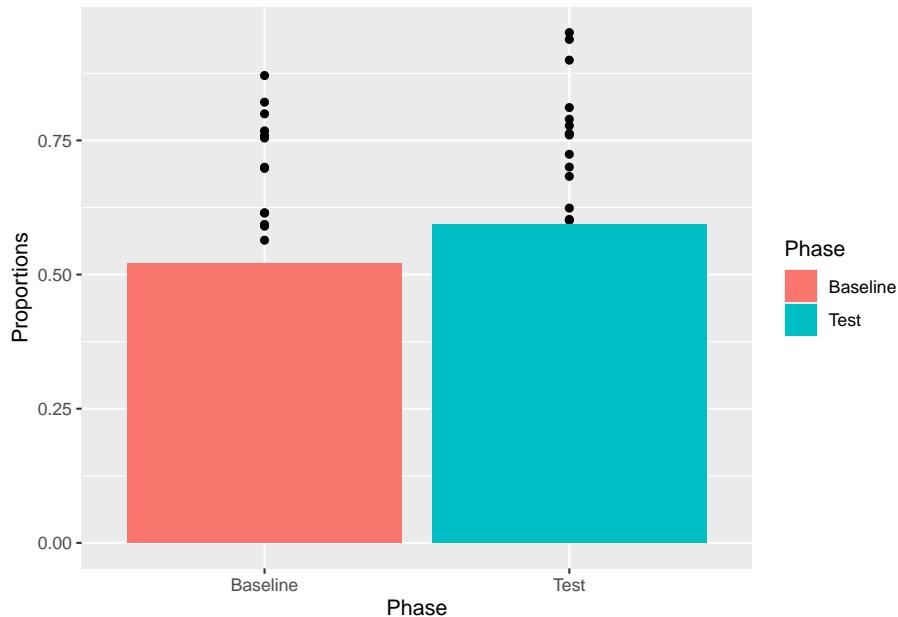
6.4.7.4 Bar plot

It's very common to use bars in graphs. We can easily do this by using `geom_bar`, rather than `geom_point`. Also, we can plot bars for the means, and keep showing the dots like this...(note this will be messed up, but I want to show you why).

Also look for these changes.

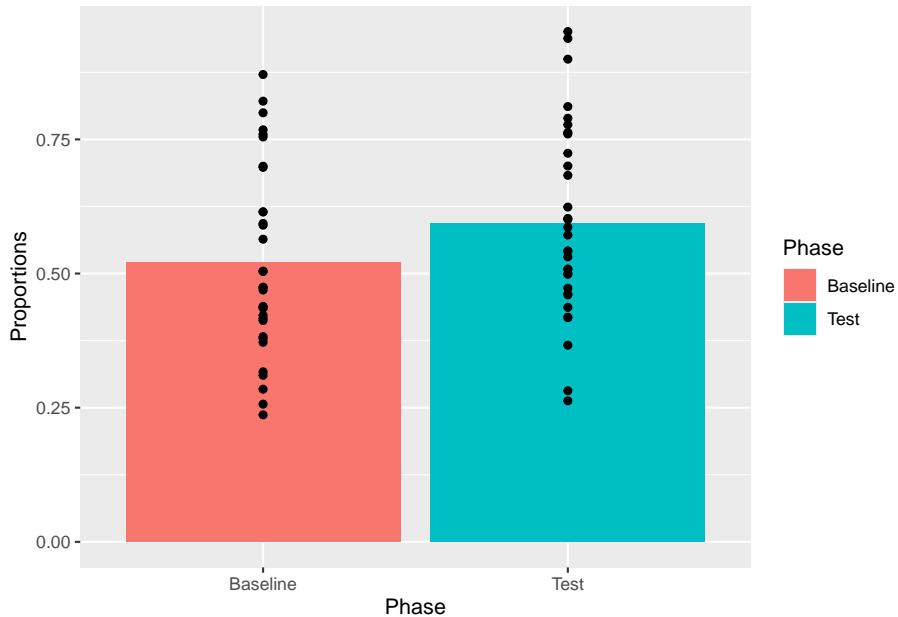
1. added `stat="identity"` Necessary for bar plot to show specific numbers
2. added `aes(fill=Phase)` Makes each bar a different color, depending on which phase it comes from

```
ggplot(plot_df, aes(x=Phase, y=Proportions))+
  geom_point()+
  geom_bar(data=mean_df, stat="identity", aes(fill=Phase))
```



OK, we see the bars and some of the dots, but not all of them. What is going on? Remember, ggplot2 works in layers. Whatever layer you add first will be printed first in the graph, whatever layer you add second will be printed on top of the first. We put the bars on top of the dots. Let's change the order of the layers so the dot's go on top of the bars.

```
ggplot(plot_df, aes(x=Phase, y=Proportions))+
  geom_point()+
  geom_bar(data=mean_df, stat="identity", aes(fill=Phase))
```



6.4.7.5 Bar plot with error bars

So far we have only plotted the means and individual sample scores. These are useful, but neither of them give us clear visual information about our statistical test. Our paired sample t-test suggested that the mean difference between Baseline and Test was not very likely by chance. It could have happened, but wouldn't happen very often.

Question: Why would the standard deviation of each mean, or the standard error of each mean be inappropriate to use in this case?

Question: How would error bars based on the standard error of the mean differences aid in visual inference about the role of chance in producing the difference?

Error bars are commonly used as an aid for visual inference. The use of error bars can be a subtle nuanced issue. This is because there are different kinds of error bars that can be plotted, and each one supports different kinds of inference. In general, the error bar is supposed to represent some aspect of the variability associated with each mean. We could plot little bars that are +1 or -1 standard deviations of each mean, or we could do +1 or -1 standard errors of each mean. In the case of paired samples, neither of these error bars would be appropriate, they wouldn't reflect the variability associated with mean we are interested in. In a paired samples t-test, we are interested in the variability of the mean of the difference scores. Let's calculate the standard error of the mean (SEM) for the difference scores between Baseline and Test, and then add error bars to the

plot.

```
difference_scores <- baseline-test_phase #calculate difference scores
standard_error <- sd(difference_scores)/sqrt(length(difference_scores)) #calculate SEM

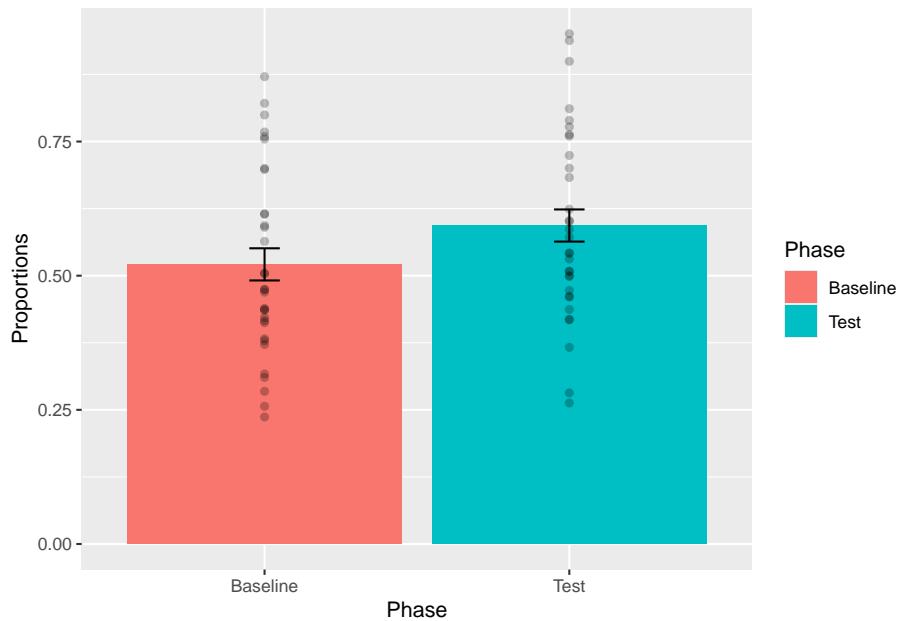
ggplot(plot_df, aes(x=Phase, y=Proportions))+  

  geom_bar(data=mean_df, stat="identity",aes(fill=Phase))+  

  geom_errorbar(data=mean_df, aes(ymax=Proportions+standard_error,  

                                  ymin=Proportions-standard_error), width=.1) +  

  geom_point(alpha=.25)
```



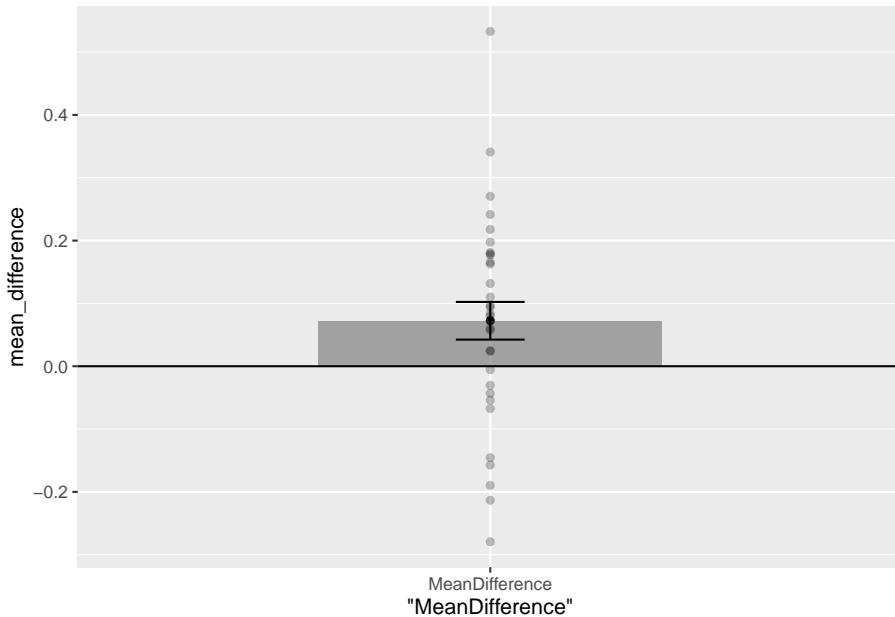
Question: What is one reason why these error bars (standard error of the mean difference between Baseline and Test) are appropriate to use. What is one reason they are not appropriate to use?

We have done something that is useful for visual inference. From the textbook, we learned that differences of about 2 standard errors of the mean are near the point where we would claim that chance is unlikely to have produced the difference. This is a rough estimate. But, we can see that the top of the error bar for Baseline is lower than the bottom of the error bar for Test, resulting in a difference greater than 2 standard error bars. So, based on this graph, we might expect the difference between conditions to be significant. We can also complain about what we have done here, we are placing the same error bars from the mean difference scores onto the means for each condition. In some sense this is misleading. The error bars are not for the depicted sample means,

they are for the hidden single set of difference scores. To make this more clear, we will make a bar plot with a single bar only for the differences scores.

```
difference_scores <- test_phase-baseline #calculate difference scores
standard_error <- sd(difference_scores)/sqrt(length(difference_scores)) #calculate SEM
mean_difference <- mean(difference_scores)

qplot(x="MeanDifference", y=mean_difference)+
  geom_bar(stat="identity", width=.5, alpha=.5)+
  geom_hline(yintercept=0)+
  geom_point(aes(y=difference_scores), alpha=.25)+
  geom_errorbar(aes(ymin=mean_difference-standard_error,
                     ymax=mean_difference+standard_error), width=.1)
```



Question: Why is it more appropriate to put the standard error of the difference on this bar graph? What important aspects of the original results shown in the previous graph are missing from this graph?

This plot is very useful too, it gives us some new information. We can see that the mean difference (test - baseline) was greater than 0. And, we are plotting the standard error of the mean differences, which are the error bars that more formally belong to this graph. Still, we are in a position to make some guesses for visual inference. The lower error bar represents only 1 SEM. It does not cross 0, but the fact that 1 SEM does not cross zero isn't important. For SEMs it's usually closer to 2. We can sort of visually guesstimate that that 2 SEMs would not cross zero, which would suggest we would obtain a small p-value.

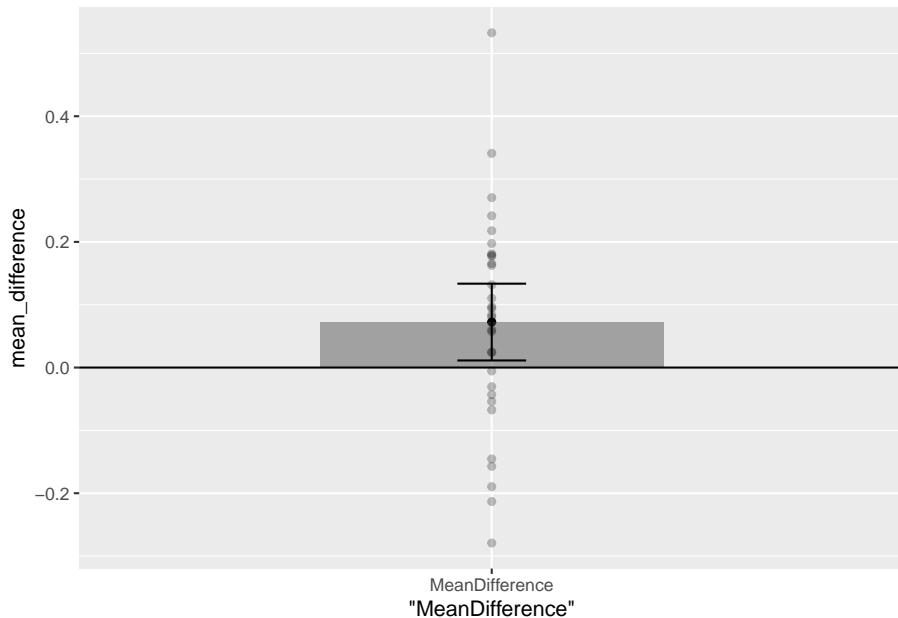
We also see each of the mean difference scores. It's very clear that they are all over the place. This means that not every infant showed a looking time preference toward the singer of the familiar song. Finally, this single bar plot misses something. It doesn't tell us what the values of the original means were.

6.4.7.6 Bar plot with confidence intervals

Confidence intervals are also often used for error bars, rather than the standard error (or other measure of variance). If we use 95% confidence intervals, then our error bars can be even more helpful for visual inference. Running the `t.test` function produces confidence interval estimates, and we can pull them out and use them for error bars.

```
t_test_results <- t.test(difference_scores)
lower_interval<- t_test_results$conf.int[1]
upper_interval<- t_test_results$conf.int[2]

qplot(x="MeanDifference", y=mean_difference)+
  geom_bar(stat="identity", width=.5, alpha=.5)+
  geom_hline(yintercept=0)+
  geom_point(aes(y=difference_scores), alpha=.25)+
  geom_errorbar(aes(ymin=lower_interval,
                     ymax=upper_interval), width=.1)
```



Notice that the 95% confidence intervals around the mean are wider than the SEM error bars from the previous graph. These new confidence intervals tell

us that 95% of the time our sample mean will fall between the two lines. The bottom line is slightly above 0, so we can now visually see support for our statistical inference that chance was unlikely to produce the result. If chance was likely to produce the result, the horizontal line indicating 0, would be well inside the confidence interval. We can also notice that the mean difference was just barely different from chance, that lower bar is almost touching 0.

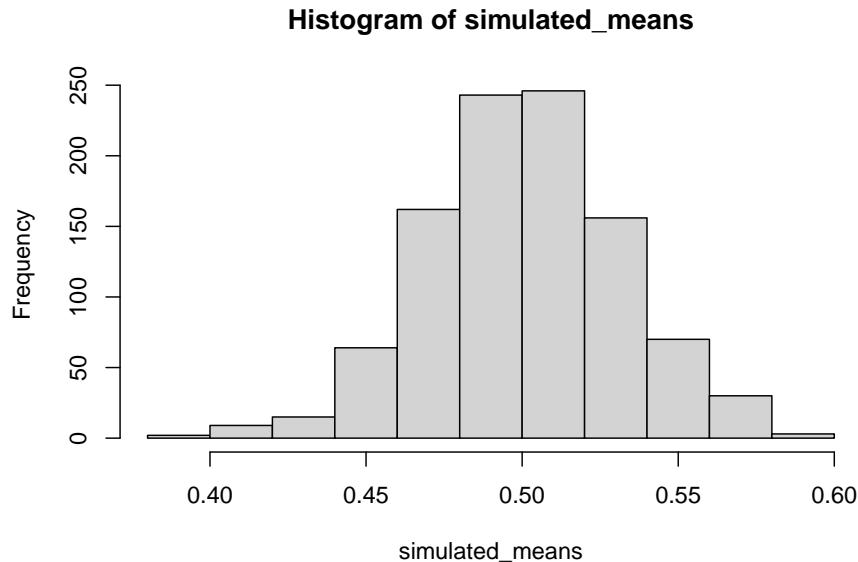
6.4.8 Data-simulation

We can do a little bit of data simulation to get a better feel for this kind of data. For example, we saw that the dots in our plots were quite variable. We might wonder about what chance can do in the current experiment. One way to do this is to estimate the standard deviation of the looking time proportions. Perhaps the best way to do that would be to an average of the standard deviation in the baseline and test_phase conditions. Then, we could simulate 32 scores from a normal distribution with mean = .5, and standard deviation equaling our mean standard deviation. We could calculate the mean of our simulated sample. And, we could do this many times, say 1000 times. Then, we could look at a histogram of our means. This will show the range of sample means we would expect just by chance. This is another way to tell whether the observed difference in this experiment in the testing phase was close or not close from being produced by chance. Take a look at the histogram. What do you think?

```
sample_sd <- (sd(baseline)+sd(test_phase))/2

simulated_means <- length(1000)
for(i in 1:1000){
  simulated_means[i] <- mean(rnorm(32,.5, sample_sd))
}

hist(simulated_means)
```



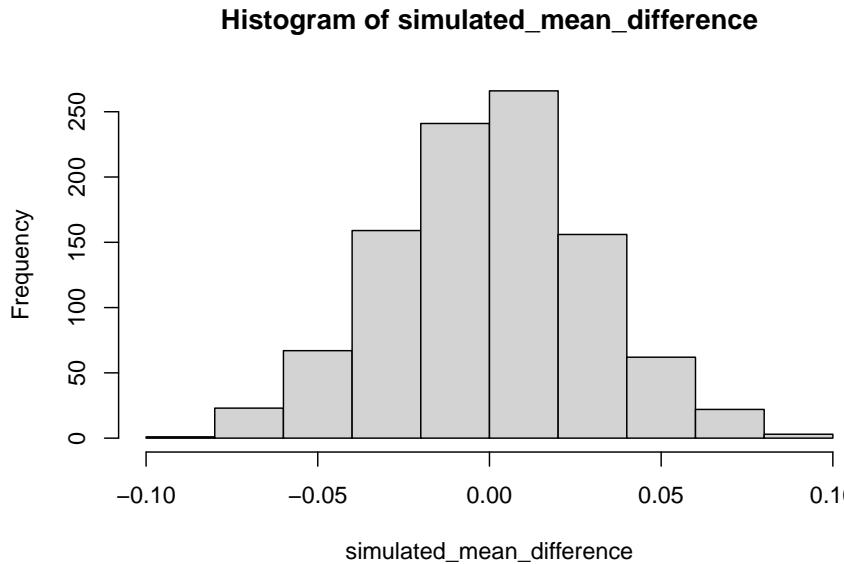
```
##### Simulating the mean differences
```

We can do the simulation slightly differently to give us a different look at chance. Above we simulated the sample means from a normal distribution centered on .5. The experimental question of interest was whether the mean difference between the baseline and test_phase condition was different. So, we should do a simulation of the difference scores. First, we estimate the standard deviation of the difference scores, and then run the simulation with a normal distribution centered on 0 (an expected mean difference of 0). This shows roughly how often we might expect mean differences of various sizes to occur. One major limitation is that we likely had only a rough estimate of the true standard deviation of these mean differences, after all there were only 32 of them, so we should take this with a grain of salt. Nevertheless, the pattern in the simulation fits well with the observations in that we already made from the data.

```
sample_sd <- sd(baseline-test_phase)

simulated_mean_difference <- length(1000)
for(i in 1:1000){
  simulated_mean_difference[i] <- mean(rnorm(32,0, sample_sd))
}

hist(simulated_mean_difference)
```



6.4.9 Generalization Exercise

(1 point - Pass/Fail)

The following code creates two variables with simulated means for each of 20 subjects in two conditions (A and B). Each sample comes from a normal distribution with mean = 100, and standard deviation = 25.

```
Condition_A <- rnorm(20,100,25)
Condition_B <- rnorm(20,100,25)
```

1. Conduct a paired samples t-test on sample data stored in the variables `Condition_A`, and `Condition_B`. Report the results of the t-test, and the mean difference between conditions
2. The above code assumes no difference in the means between Condition A and Condition B. Both samples come from the same normal distribution. Change the mean for one of the distributions. Make the change large enough so that you find a significant p-value ($p < 0.05$) when you conduct the t-test on the new simulated data. Report the t-test and the means for each condition

6.4.10 Writing assignment

(2 points - Graded)

Complete the writing assignment described in your R Markdown document for

this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

1. Answer the following questions
 - a. Explain the general concept of a t value, and how it is different from a mean and the standard error. (.33 points)
 - b. Imagine you obtained a t-value with an associated p-value of .25. Explain what this means. (.33 points)
 - c. Imagine the critical t value for a particular test was 2.6. What does the critical t-value mean (assuming alpha = 0.05 and the test is two-tailed) (.33 points)
 - d. Explain the differences between a one-sample and paired-sample t-test (1 point)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

6.5 Excel

How to do it in Excel

6.6 SPSS

In this lab, we will use SPSS to:

1. Perform a one-sample t-test
2. Perform a paired-samples t-test
3. Graph the data
4. Assess the relationship between a one-sample and a paired-samples t-test

6.6.1 Experiment Background

For this lab, we will be using the data from Mehr, Song, and Spelke's (2016) study of how music may convey important social information to infants.

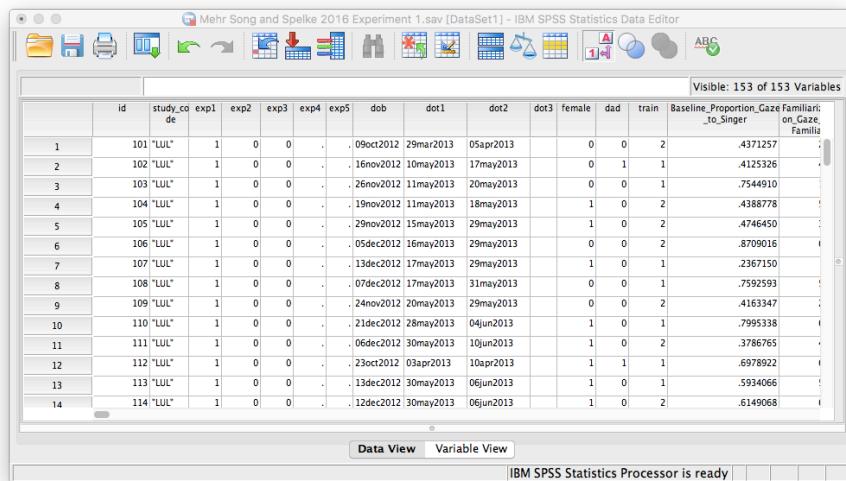
To test their hypothesis, the researchers recruited 32 infants and their parents to complete an experiment. During their first visit to the lab, the parents were taught a new lullaby (one that neither they nor their infants had heard

before). The experimenters asked the parents to sing the new lullaby to their child every day for the next 1-2 weeks.

Following this exposure period, the parents and their infant returned to the lab to complete the experimental portion of the study. Infants were first shown a screen with side-by-side videos of two unfamiliar people, each of whom were silently smiling and looking at the infant. The researchers recorded the looking behavior (or gaze) of the infants during this ‘baseline’ phase. Next, one by one, the two unfamiliar people on the screen sang either the lullaby that the parents learned or a different lullaby (that had the same lyrics and rhythm, but a different melody). Finally, the infants saw the same silent video used at baseline, and the researchers again recorded the looking behavior of the infants during this ‘test’ phase.

6.6.2 Performing a one-sample t-test

First, let’s open the relevant data file; Here is the link. It’s called “MehrSongSpelke2016_Experiment1.sav” Open this file in SPSS. Your data should look like this:



The screenshot shows the IBM SPSS Statistics Data Editor window. The title bar reads "Mehr Song and Spelke 2016 Experiment 1.sav [DataSet1] - IBM SPSS Statistics Data Editor". The main area displays a data table with 153 columns and 153 rows. The columns are labeled with variable names such as "Id", "study_code", "exp1", "exp2", "exp3", "exp4", "exp5", "dob", "dot1", "dot2", "dot3", "female", "dad", "train", and "Baseline_Proportion_Gaze_Familiar_to_Singer". The data consists of numerical values and dates. The status bar at the bottom indicates "Visible: 153 of 153 Variables" and "IBM SPSS Statistics Processor is ready".

We will use SPSS to determine whether the infants looked equally at each of the people in the side-by-side videos at baseline. Let’s think about why this is important. The infants in this experiment watched silent video recordings of two women (Baseline). Later, each person sings a song, one familiar (their parents sung the song to them many times), and one unfamiliar. Afterwards, the infants watched the silent video of the two singers again (test phase). The critical question was whether the infants would look more to the person who

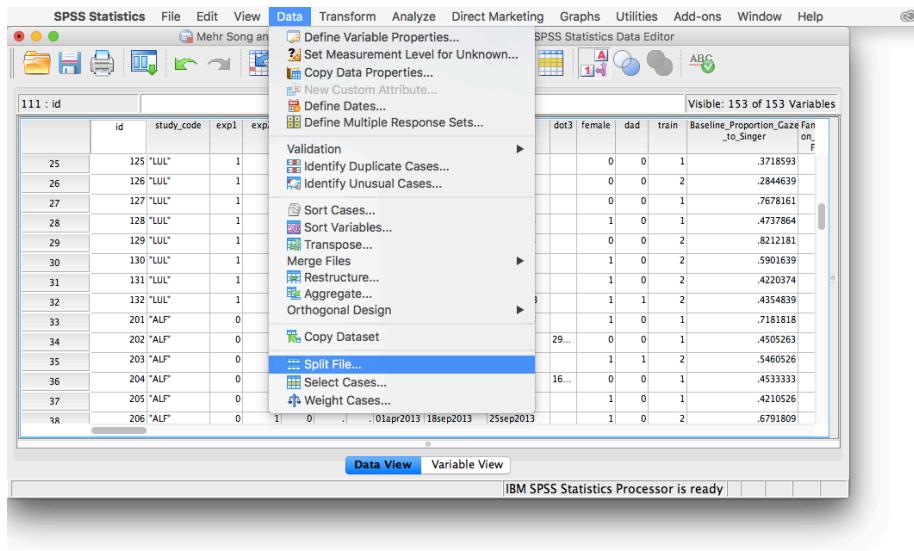
sung the familiar song compared to the person who sung the unfamiliar song.

If the measure of what infants “like” or “prefer” is based on how long they look at a stimulus (in this case, a person in a video), we need to know that at baseline (before either of the people were associated with a song), infants looked equally at each person. If one of the people in the video was more memorable or attractive to the infants, then they would have a preferential bias towards looking at one person over the other. This could mean that the results of the study may have nothing to do with the songs, but rather an inherent bias towards the appearance of one of the people. This is why it is important to ensure that the infants looked at the people equally to start with (at baseline).

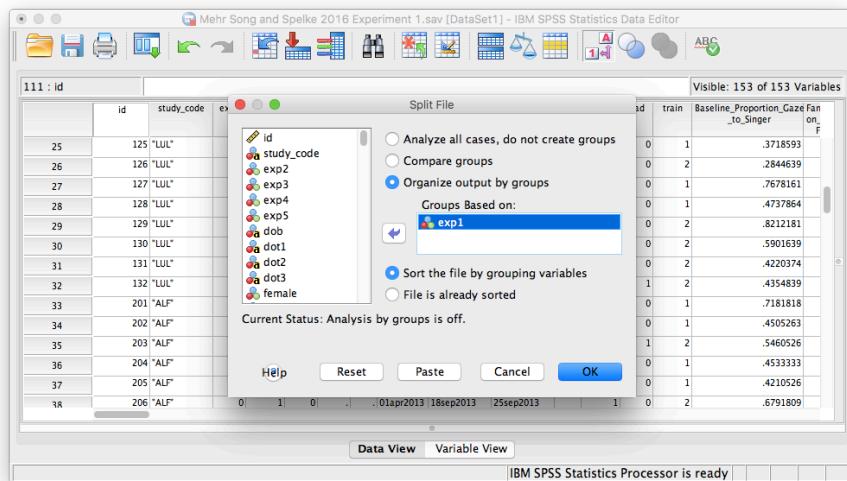
In our data file, the variable we will use is called “Baseline_Proportion_Gaze_to_Singer”; this variable represents the proportion of time (out of 100%) that the infant spent looking at the person who would later be the singer of the familiar song. If the infant looks equally at each person, then the proportion of time spent looking at one out of the two people should be .5, or half of the time. Thus, we want to test whether the proportions measured are different from .5. Whenever we want to compare a sample data set (the proportions measured for all 32 infants) to a known mean (.5, or the proportion we expect if infants looked equally at both people), we use a one-sample t-test.

Before we begin, it’s important to notice that our data file has 96 rows in it, for 96 infants. However, Experiment 1 only used 32 infants. If you find the column labeled `exp1`, it will show you which infants belong to that experiment. You’ll notice there are a series of 1s (32 of them, to be exact) signifying the infants in Experiment 1. We will have to split our data file so we can run subsequent analyses only on this Experiment 1 data.

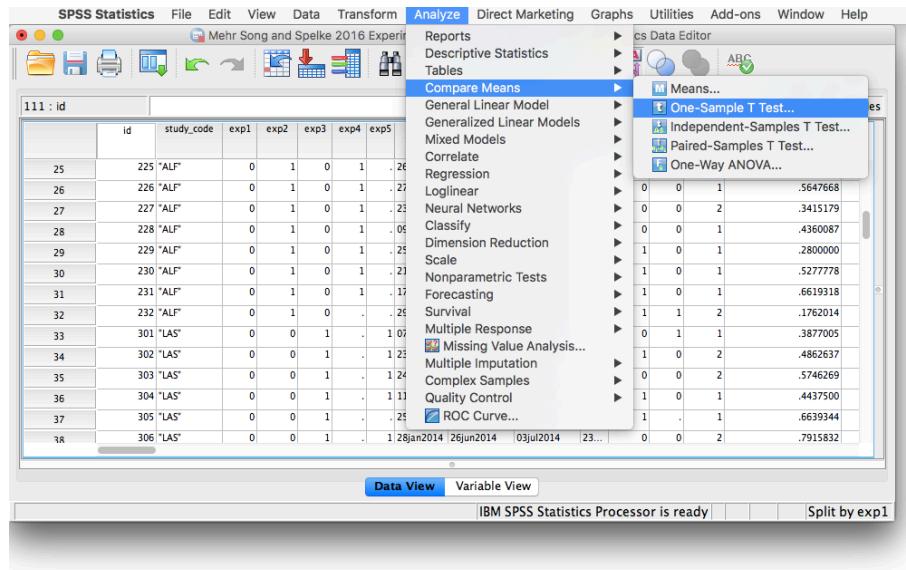
Go to Data, then Split file...



A window will appear. Choose Organize output by groups and enter the `exp1` variable in the “Groups Based on” field:

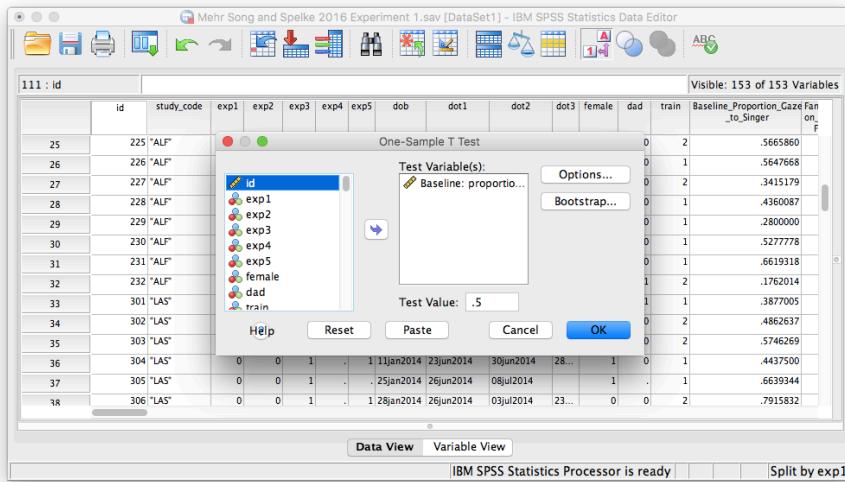


Click OK. SPSS will generate an output window to confirm you have split your file. In order to run the t-test, go to the Analyze menu, then choose Compare Means and then One-Sample T-test:



This will open up a window in which you will specify which variable contains the relevant sample data. Remember, we are interested in the variable labeled **Baseline_Proportion_Gaze_to_Singer**, so let's choose this variable from the list on the left and move it to the right-hand field using the arrow.

Next, make sure to change the value in the field labeled “Test Value” to the mean to which you are comparing the sample. Since we are comparing our sample’s proportions to the proportion that represents equal time spent looking at both people in the video, we are going to use .5 as our test value. Enter this number in the appropriate field. Your window should look like this:



Now, click OK. SPSS will produce several output tables; some for $\text{exp1} = 0$ (these are the infants not included in experiment 1) and for $\text{exp} = 1$ (for the infants included in experiment 1—this is the one we want!), as follows:

exp1 = 1

One-Sample Statistics ^a				
	N	Mean	Std. Deviation	Std. Error Mean
Baseline: proportion gaze to singer	32	.521096675	.176965064	.031283299

a. $\text{exp1} = 1$

One-Sample Test ^a						
	Test Value = .5					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
Baseline: proportion gaze to singer	.674	31	.505	.021096675	-.04270603	.084899384

a. $\text{exp1} = 1$

The first table contains descriptive statistics related to the sample. We can see from this table that the average proportion of time spent looking at the person who would later be the familiar song singer is .521. This is very close to .50.

The second table contains the information resulting from our t-test: The t-statistic calculated is .674, with 31 (N-1) degrees of freedom. The p-value is listed under “Sig.”. The fact that this value is not less than our alpha level (set at .05) means we fail to reject the null hypothesis. In other words, the

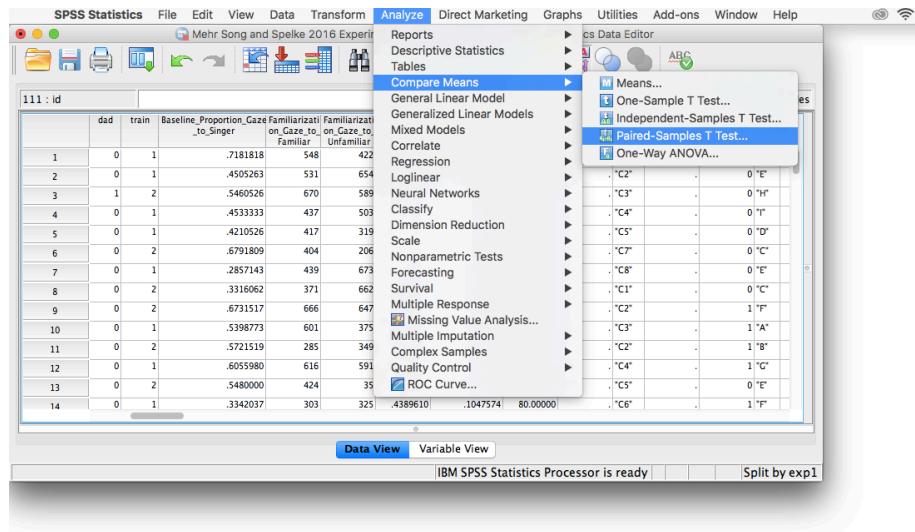
proportion of time infants spent looking at the singer was not different from .50. We might formally words this as follows:

During the baseline condition, the mean proportion looking time toward the singer was .52, and was not significantly different from .5, according to a one-sample test, $t(31) = .67$, $p = .505$.

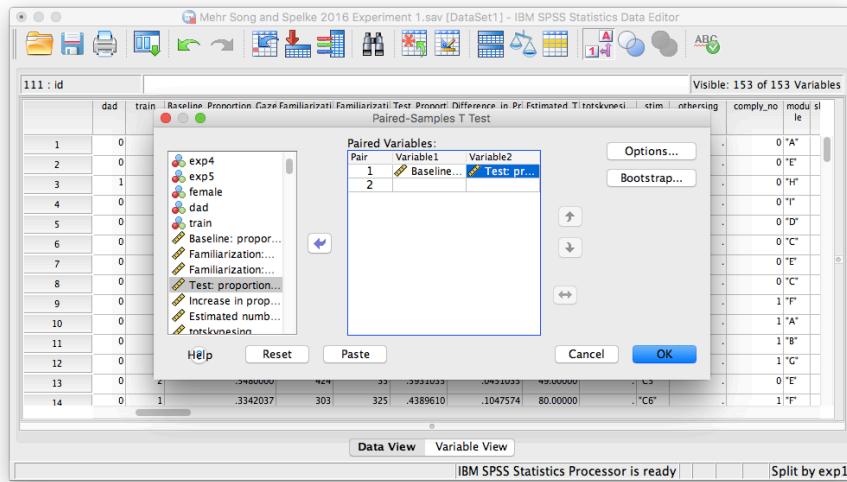
6.6.3 Performing a paired-samples t-test

In this example, we will be comparing data from two samples taken from the same individuals. In the same data file as above, we have two variables named **Baseline_Proportion_Gaze_to_Singer** and **Test_Proportion_Gaze_to_Singer**. These variables represent the proportion of time spent looking at the familiar song singer at baseline, and then again in the test phase (after the 1-2 week period in which infants have been familiarized with the song). Because each infant was measured once (baseline), and then re-measured again (test), we will use a paired-samples t-test.

Let's continue with the split file we used in the previous example. In order to run the paired-samples t-test, go to Analyze, then Compare Means, then Paired-Samples T-test...



A window will appear with a list of all variables in a field on the left. We must choose the two variables that contain our data: **Baseline_Proportion_Gaze_to_Singer** and **Test_Proportion_Gaze_to_Singer**. Move these into the right-hand field individually using the arrow. Make sure that they appear next to each other in the right-hand field, as follows:



Click OK, and SPSS will produce several output tables. Remember, we're only choosing the tables for `exp1 = 1` (this includes only the 32 infants in experiment 1). The tables are as follows:

`exp1 = 1`

Paired Samples Statistics^a

	Mean	N	Std. Deviation	Std. Error Mean
Pair 1 Baseline: proportion gaze to singer	.521096675	32	.176965064	.031283299
Test: proportion gaze to singer of familiar song	.593491250	32	.178688366	.031587939

a. `exp1 = 1`

Paired Samples Correlations^a

	N	Correlation	Sig.
Pair 1 Baseline: proportion gaze to singer & Test: proportion gaze to singer of familiar song	32	.546	.001

a. `exp1 = 1`

Paired Samples Test^a

	Paired Differences				t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference			
				Lower			
Pair 1 Baseline: proportion gaze to singer - Test: proportion gaze to singer of familiar song	-.07239457	.169475427	.029959306	-.13349698	-2.416	31	.022

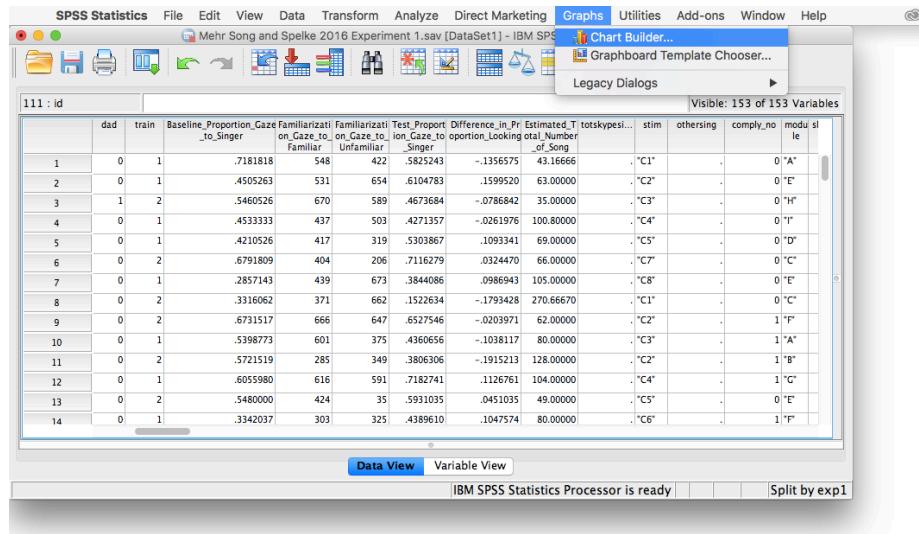
a. `exp1 = 1`

The paired-samples t-test produces a t-value, degrees of freedom ($N-1$), and a p-value. You can interpret these values the same way as in the previous example.

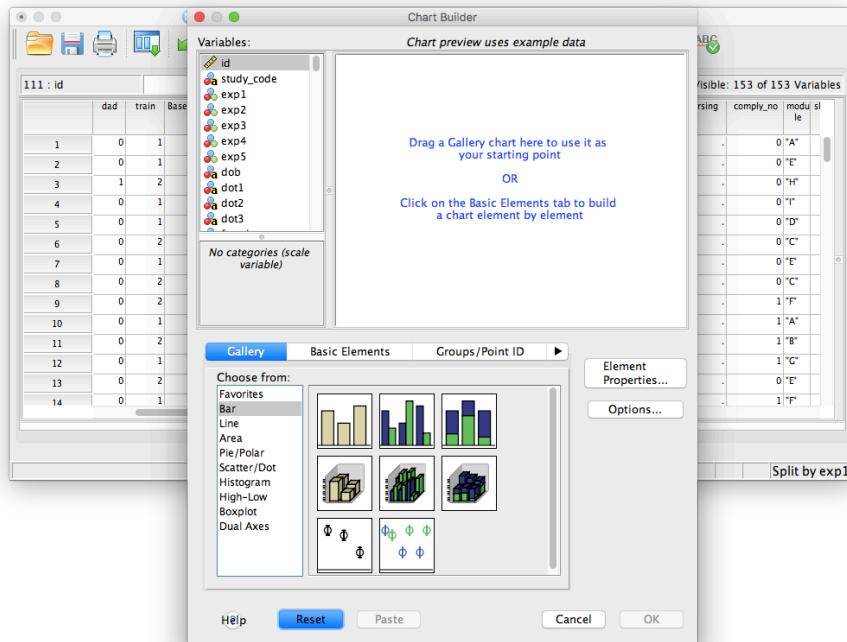
During the baseline condition, The difference between Infants' mean proportion looking time toward the singer at baseline and the test phase was $-.072$, and was significant according to a paired-samples t-test, $t(31) = -2.416$, $p < .05$.

6.6.4 Graphing your results

In order to visualize the difference between means (for a paired-samples t-test, in this case), it is most appropriate to create a bar chart, or bar plot. Because our two conditions (baseline and test) are categorical, discrete (and not continuous) variables, this type of chart is most appropriate. Let's begin by going to **Graphs**, then **Chart Builder...**

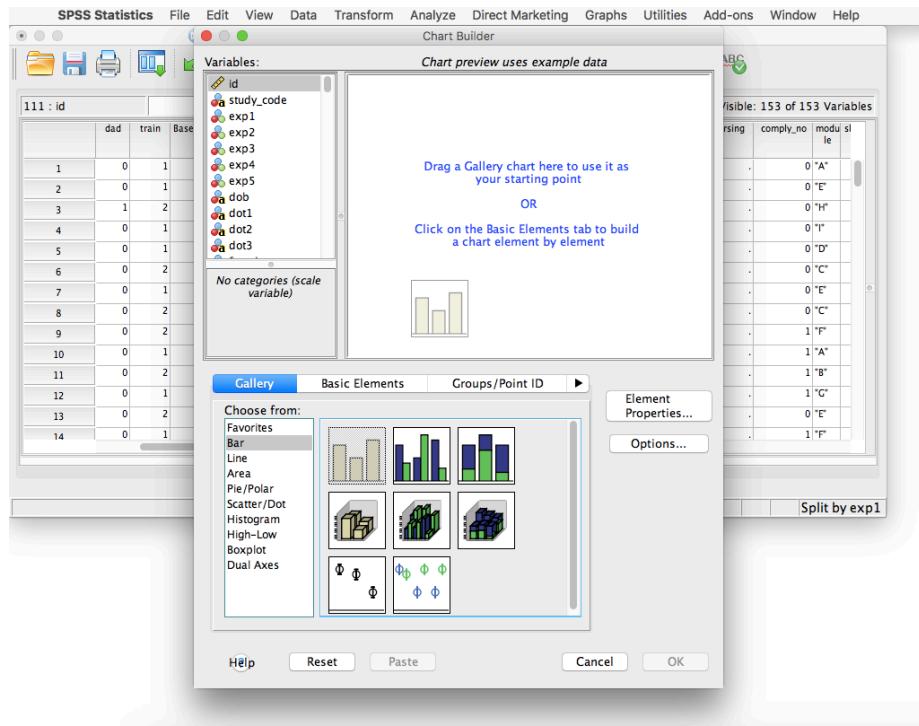


The Chart Builder window will open and look like this:

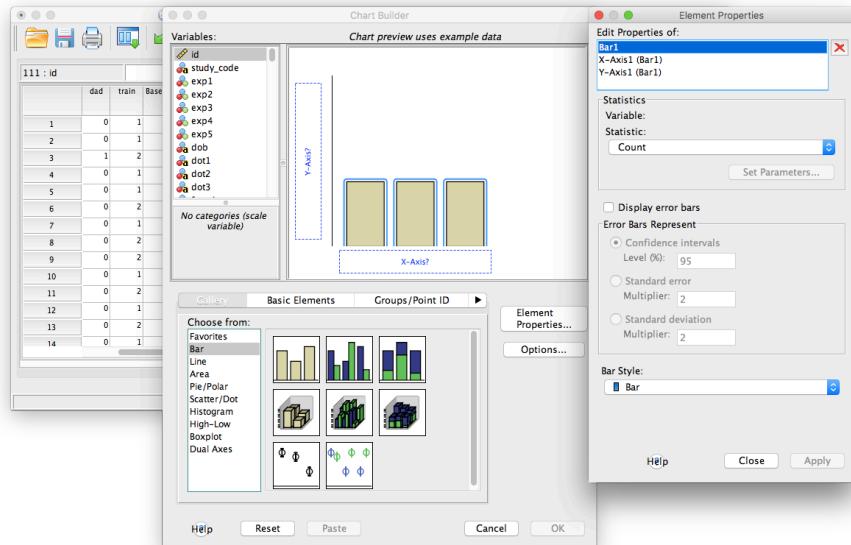


We are going to choose a simple bar chart, which is represented by the bar icon at the top left of the set of chart options. Click on this icon and drag it into the chart window as follows:

204 CHAPTER 6. LAB 6: T-TEST (ONE-SAMPLE, PAIRED SAMPLE)

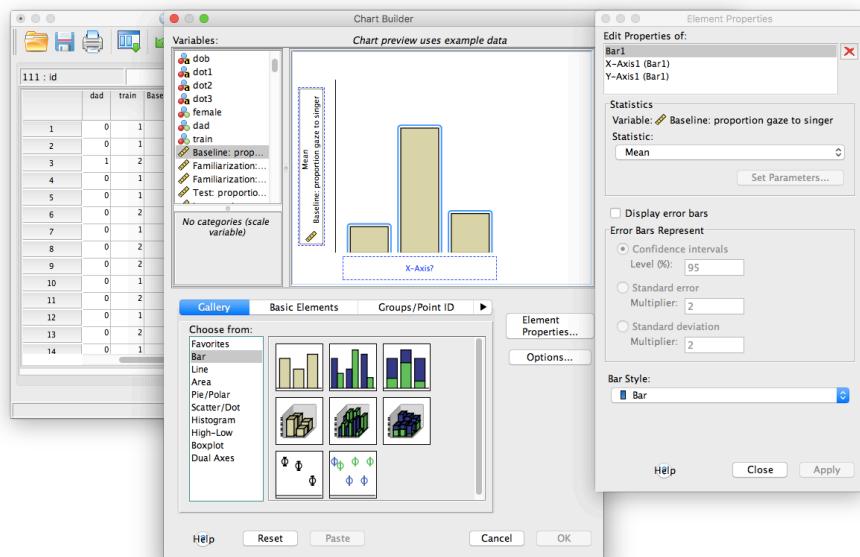


Once you do this, your window should look like this:

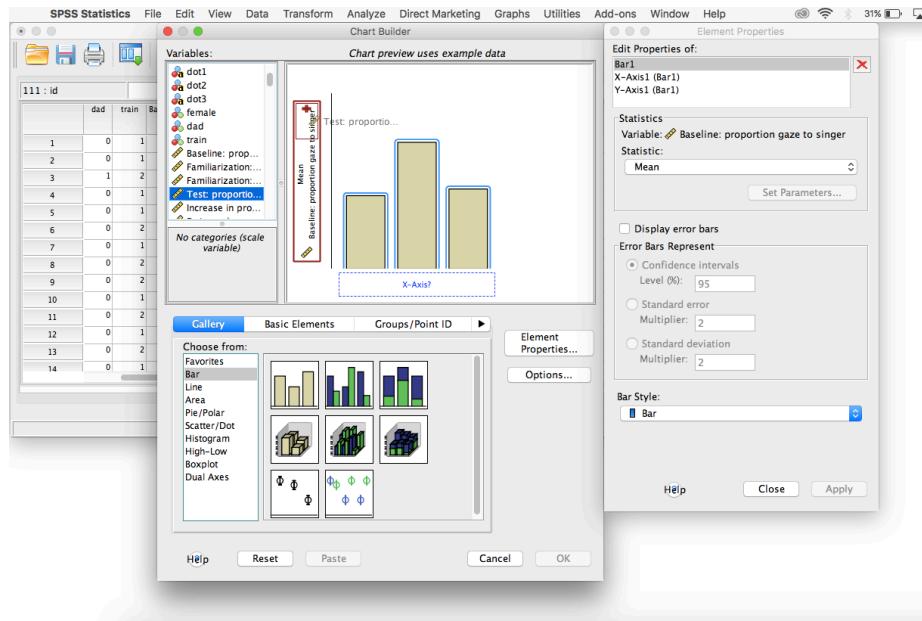


From here, you must indicate which variables should be included in the chart

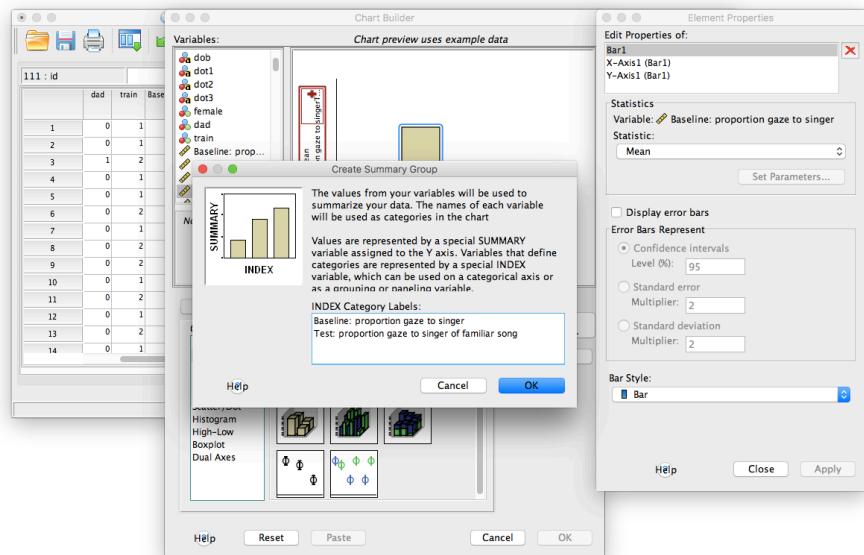
on the y-axis. We will first take the **Baseline_Proportion_Gaze_to_Singer** variable, found in the list on the left of the window, and drag it to the y-axis in the chart window. You will see that the y-axis area will highlight with a red border when you do this.



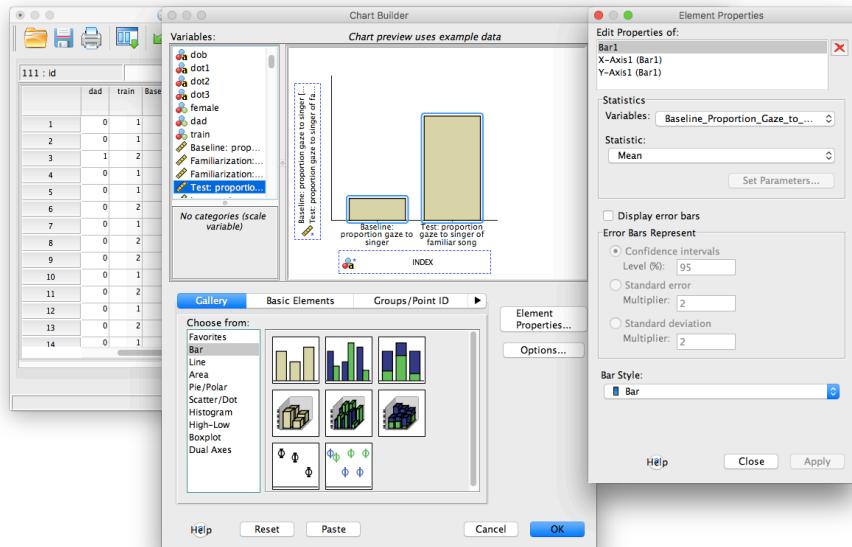
Now, take the next variable, **Test_Proportion_Gaze_to_Singer**, and drag it over the same y-axis title, but this time aim for the very top of the variable field. You will know you're doing it right when a red plus sign appears in this area as you hover:



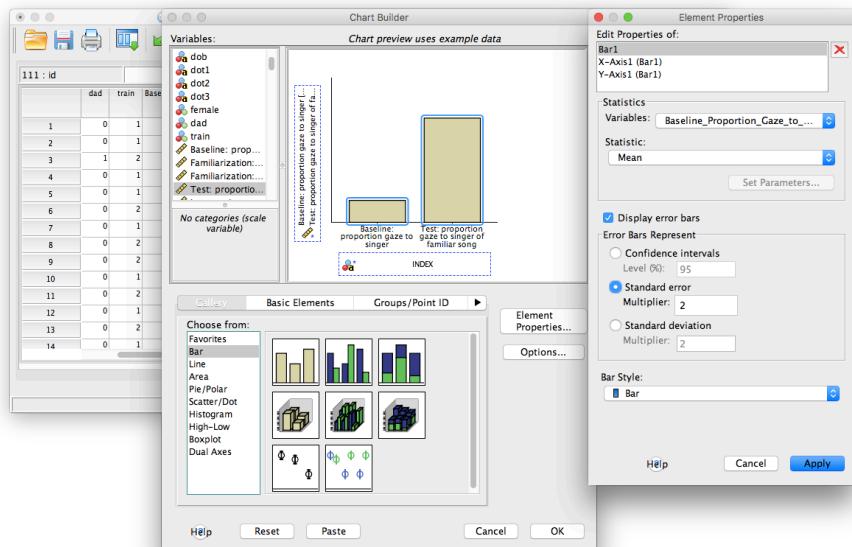
Drop this variable into this field once you see the plus sign. A window will pop up asking if you want both variables on this axis. Click OK.



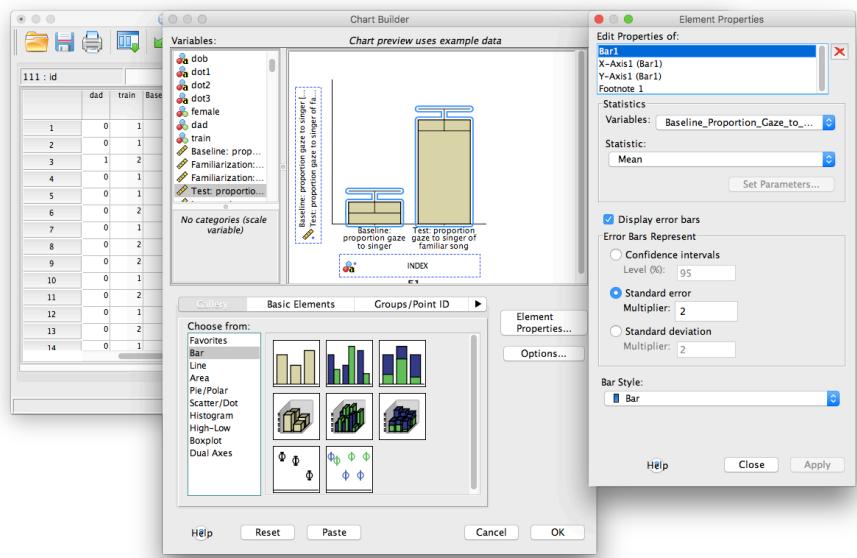
The result will look like this (both variables' means, baseline and test, appear in the chart).



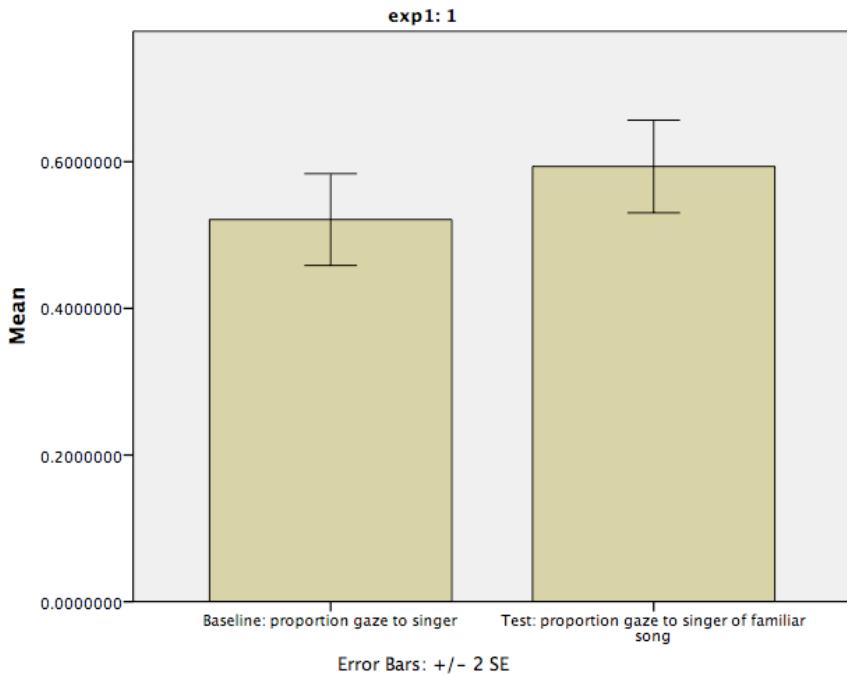
From here, you can customize the appearance of the graph. One thing we will do is add Error Bars. Find the window to the right of the main chart builder window, and select Error Bars. You will have the option to select Confidence Intervals, Standard Error, etc. Choose Standard Error and set the Multiplier to 2. Click Apply.



Now, you should be able to see the bars in the chart builder window.



If you are satisfied with the chart, click OK and it will appear in an SPSS output window:



6.6.5 The relationship between the one-sample and the paired-samples t-test

This section is dedicated to demonstrating that the one-sample and paired-samples t-tests are almost the very same thing. In order to understand this, we must reframe how we think about the paired-samples t-test. We typically consider it a comparison between two samples in which observations have been repeated (like in a before-after scenario, for example). We can also think of the paired-samples t-test as testing the DIFFERENCE scores between two conditions. If the difference between two conditions is 0, then there is no difference, or there is no effect of the independent variable. If the difference is non-zero, then we can say there IS a difference.

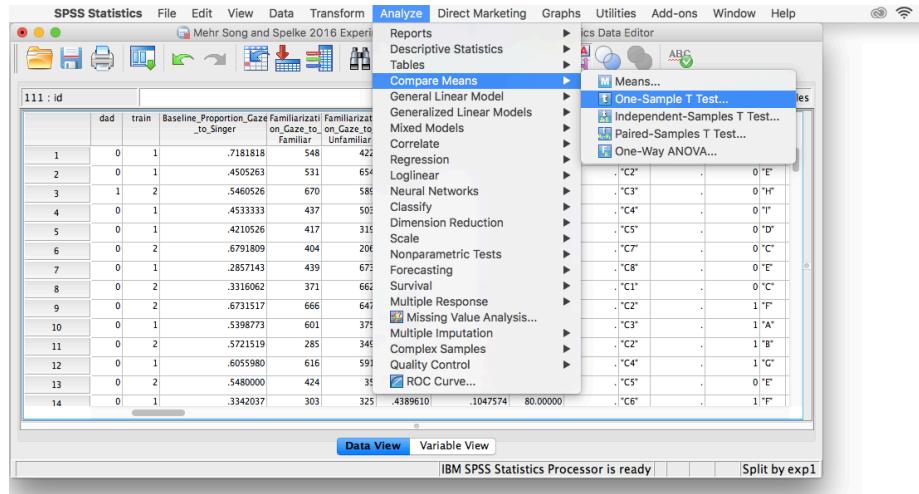
Thus, the paired-samples t-test is exactly the same as the one-sample t-test if we consider it as a comparison of the mean DIFFERENCE score to 0. Let's demonstrate this using SPSS.

We'll continue using the same data as above. Specifically, locate the variable named `Difference_in_Proportion_Looking`. This variable contains the difference score between the baseline and test conditions for each infant. We're going to treat this as our one sample (as in the one-sample t-test), and we'll be

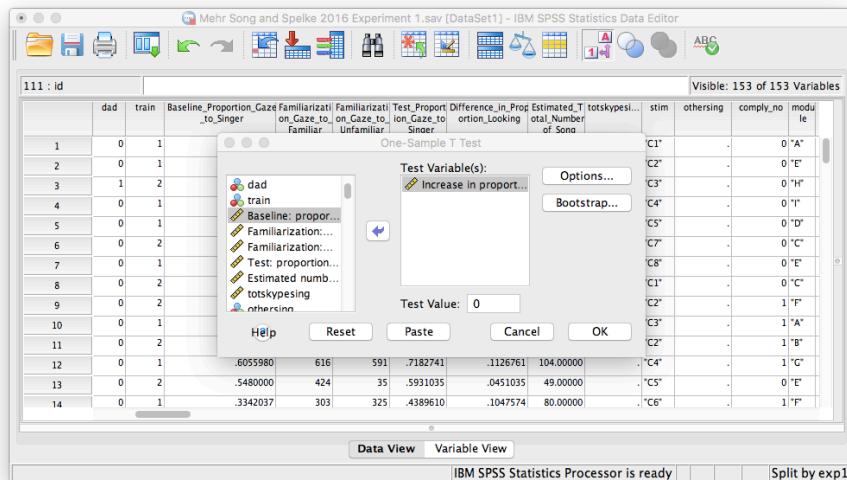
210 CHAPTER 6. LAB 6: T-TEST (ONE-SAMPLE, PAIRED SAMPLE)

comparing it to a mean of 0. Remember also that our file is still split so we can do analyses on only those individuals in experiment 1.

Let's begin. Go to Analyze, then Compare Means, then One-Sample t-test...



A window will appear. Move the variable of interest, `Difference_in_Proportion_Looking`, into the “Test Variable(s)” field. Below that, make sure to enter 0 in the “Test Value” field.



Then click OK. SPSS will generate several tables in the output window. Make sure to find the tables associated with `exp1 = 1`. They should look like this:

exp1 = 1

One-Sample Statistics^a

	N	Mean	Std. Deviation	Std. Error Mean
Increase in proportion of looking from baseline to test	32	.072394569	.169475428	.029959306

a. exp1 = 1

One-Sample Test^a

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Increase in proportion of looking from baseline to test	2.416	31	.022	.072394569	.011292161	.133496976

a. exp1 = 1

You will see that the t-statistic and p-value are identical to the ones obtained when we conducted a paired-samples t-test. Therefore, a paired-samples t-test conducted on two samples is equivalent to a one-sample t-test conducted on the difference scores (calculated on the two samples) compared against 0.

Nota Bene: You will see that in the previous paired-samples t-test we obtained a t-statistic of -2.416, but here the t-statistic is positive. This difference comes from the fact that SPSS automatically subtracts the later-occurring variable in your spreadsheet from the earlier-occurring one. If your variables were ordered differently, you would obtain a positive t-statistic as well. In other words, the sign does not matter; it results from the order of the variables.

6.6.6 Practice Problems

1. Use the data file from this lab tutorial to test whether the number of frames the baby spent gazing at the familiar song is significantly different than the number of frames spent gazing at the unfamiliar song (use alpha = .05). Report your results in proper statistical reporting format.
2. Graph this result (including 1 SEM error bars) as a bar graph.
3. Compute a new variable representing the difference in number of frames the baby spent gazing between the familiar and unfamiliar song conditions. Test this difference score against a mean of 0 (use alpha=.05). Report your results in proper statistical reporting format.

6.7 JAMOVI

How to do it in JAMOVI

Chapter 7

Lab 7: t-test (Independent Sample)

I think he [Gosset] was really the big influence in statistics... he asked the questions and Pearson and Fisher put them into statistical language, and then Neyman came to work with the mathematics. But I think most of it came from Gosset. —F. N. David

This lab is modified and extended from Open Stats Labs. Thanks to Open Stats Labs (Dr. Kevin P. McIntyre) for their fantastic work.

7.1 Do you come across as smarter when people read what you say or hear what you say?

7.1.1 STUDY DESCRIPTION

Imagine you were a job candidate trying to pitch your skills to a potential employer. Would you be more likely to get the job after giving a short speech describing your skills, or after writing a short speech and having a potential employer read those words? That was the question raised by Schroeder and Epley (2015). The authors predicted that a person's speech (i.e., vocal tone, cadence, and pitch) communicates information about their intellect better than their written words (even if they are the same words as in the speech).

To examine this possibility, the authors randomly assigned 39 professional recruiters for Fortune 500 companies to one of two conditions. In the audio condition, participants listened to audio recordings of a job candidate's spoken job pitch. In the transcript condition, participants read a transcription of the job candidate's pitch. After hearing or reading the pitch, the participants rated

the job candidates on three dimensions: intelligence, competence, and thoughtfulness. These ratings were then averaged to create a single measure of the job candidate's intellect, with higher scores indicating the recruiters rated the candidates as higher in intellect. The participants also rated their overall impression of the job candidate (a composite of two items measuring positive and negative impressions). Finally, the participants indicated how likely they would be to recommend hiring the job candidate (0 - not at all likely, 10 - extremely likely).

What happened? Did the recruiters think job applicants were smarter when they read the transcripts, or when they heard the applicants speak? We have the data, we can find out.

7.2 Lab skills learned

1. Conduct independent samples t-tests
2. Generate figures
3. Discuss the results and implications

7.3 Important Stuff

- citation: Schroeder, J., & Epley, N. (2015). The sound of intellect: Speech reveals a thoughtful mind, increasing a job candidate's appeal. *Psychological Science*, 26, 877-891.
- Link to .pdf of article
- Data in .csv format
- Data in SPSS format

7.4 R

7.4.1 Load the data

Remember that any line with a # makes a comment and the code does not run. Below is how to load the .csv data from the online repository, or from a local file (you need to change the file path to where the local file is, if you downloaded it). The data contains all of the measures and conditions from Experiment 4.

```
library(data.table)
# load from github repo
#all_data <- fread("https://raw.githubusercontent.com/CrumpLab/statisticsLab/master/data/SchroederEpley2015data.csv") # load from file on computer
```

7.4.2 Inspect data frame

This will give you a big picture of the data frame. Click the button to view it in your browser, then take a look to see what is in it.

```
library(summarytools)
view(dfSummary(all_data))
```

7.4.3 Find the data you need

This time the data comes pre-filtered for us. The authors ran lots of experiments, but we only have the data from Experiment 4. This is great, we don't need to subset the data frame to find all of the data that we need. But, we do still need to understand what data we want to analyze. Let's start with identify the column that codes the experimental conditions for whether or not the evaluator read a transcript or heard the interview.

7.4.3.1 Condition variable

Lucky for us, the condition variable is called `CONDITION`! Let's take a look. We printed it out just by writing down `all_data$CONDITION`. There 0s and 1s for each condition (audio vs. transcript). But which one is which? This isn't clear from the data, and it isn't clear from the paper, or from the repository. We have to do some guess work. I went ahead and computed the means for the `Intellect_rating` between each condition, and then compared those to the graph in the paper for E4. It looks like 1 = audio condition, and 0 = transcript condition.

```
all_data$CONDITION

## [1] 1 1 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1
## [39] 1

aggregate(Intellect_Rating~CONDITION,all_data,mean)

##   CONDITION Intellect_Rating
## 1           0      3.648148
## 2           1      5.634921
```

Let's use words instead of 0s and 1s to refer to our experimental conditions. To do this, we will change the values of 0 and 1, to the words `transcript` and `audio`. We can do this in two steps. First we convert the `CONDITION` column to a factor. This will automatically turn the 0s and 1s into strings (not numbers, text). Factors have an internal variable for the names of the levels, which will be 0 and 1. We can simply change the level names to `transcript` and `audio`.

```
all_data$CONDITION <- as.factor(all_data$CONDITION)
levels(all_data$CONDITION) <- c("transcript","audio")
```

Now if you look at the `all_data` variable, you will see the words transcript and audio, where 0s and 1s used to be.

7.4.3.2 Dependent Measures

Next it's time to find the dependent measure columns. The graph from the paper shows three different measures in each condition. These included `Intellect`, `General Impression`, and `Hiring Likelihood`. Every evaluator (either given a transcript or audio recording of the interview) gave ratings on a scale of 1 to 10 for each of those concepts. It's not immediately clear which columns in `all_data` correspond to those three measures. There are lots of different measures that could be the ones they reported. It turns out the relevant ones are called

1. `Intellect_Rating`
2. `Impression_Rating`
3. `Hire_Rating`

In this tutorial we are going to walk through doing an independent samples t-test for the first measure, `Intellect_Rating`. You can follow these same steps to complete the same kind of t-test for the other two variables.

7.4.4 Look at the dependent variable.

Question: Why do we always want to look at the data?

What is the first thing we do before even considering an inferential test? Look at the data. Always look at the data. We could make a dot plot or histogram of the data from the `Intellect_ratings`. But, from our last lab we already learned how to make graphs showing most of the information we would want to look at. For example, we could make a bar graph that has the means for each condition (transcript vs. audio), standard errors of the mean and the actual scores as little dots. This would be great to look at it. Not only will it tell us if there are really weird numbers in the data (who knows maybe the data file is corrupted, you need to look), it will also give us strong intuitions about what to expect for the t-test.

We can plot each score as a dot using the `all_data` data frame. If we want to add on a layer for the sample means, and for the sample standard errors, we have to compute those and put them in a new data frame first. Then we use both data frames with `ggplot` to plot all of the information.

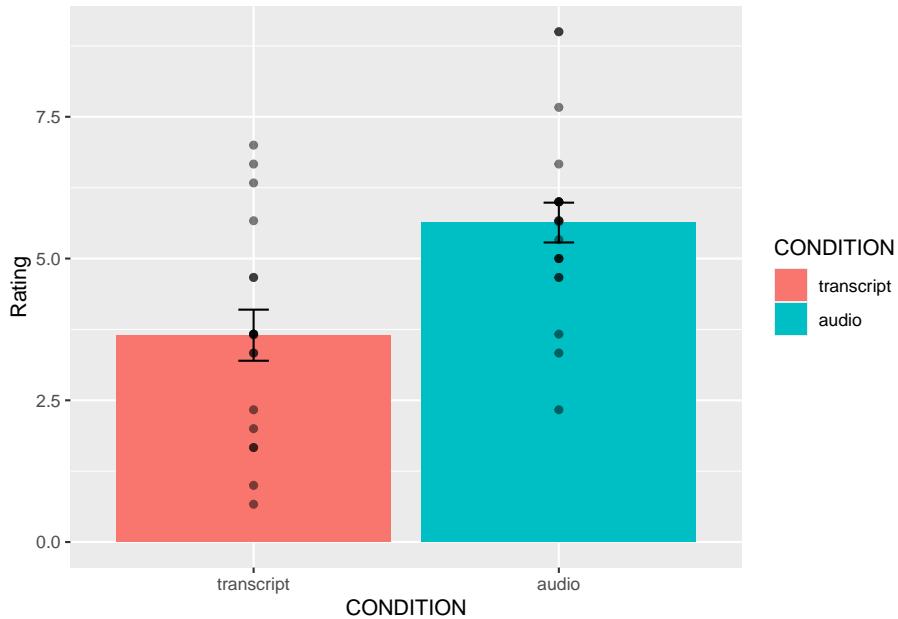
We will use `dplyr` to quickly get the means and the standard errors and put them in a new data frame called `descriptive_df`.

```
library(dplyr)
library(ggplot2)

# get means and SEs
```

```
descriptive_df <- all_data %>%
  group_by(CONDITION) %>%
  summarise(means = mean(Intellect_Rating),
            SEs = sd(Intellect_Rating)/sqrt(length(Intellect_Rating)))

# Make the plot
ggplot(descriptive_df, aes(x=CONDITION, y=means)) +
  geom_bar(stat="identity", aes(fill=CONDITION))+ # add means
  geom_errorbar(aes(ymin=means-SEs, # add error bars
                     ymax=means+SEs), width=.1) +
  geom_point(data=all_data, aes(x=CONDITION, y=Intellect_Rating), alpha=.5) +
  geom_point(alpha=.25) +
  ylab("Rating")
```



This plot is very useful. First, we can see the numbers in our dependent measure are behaving sensibly. We know that the numbers have to be between 1-10, because those were the only options in the scale. If we found numbers bigger or smaller, we would know something was wrong. Checking for things that are obviously wrong in the data is one reason why we always look at first. We are checking for obvious errors. There are other ways to check to, but looking is fast and easy.

Question: Why are the standard errors of each sample an appropriate thing to use for error bars?

Now that you can see the patterns in the data, you should form an intuition

about how the independent samples t-test will turn out. You can see how big the error bars (+1/-1 standard error of each sample man). The t-test will tell us whether the observed difference (or greater) is likely due to chance. Should we find a big t-value or a small t-value? Should we find a big p-value or a small t-value. If you understand how t-values and p-values work, the answer should be very clear from the graph. You should already know how the t-test will turn out before you run it. Running it will confirm what you already suspect to be true.

7.4.5 Conduct Independent samples t-test

Question: Why are we conducting an independent samples t-test, and not a one-sample or paired samples t-test?

We use the very same `t.test` function that we used last time to conduct a t-test. The only difference is that we don't tell the R to use a paired sample t-test. We leave the `paired=TRUE` statement out, and R automatically knows we want to do an independent samples t-test. Remember to set the `var.equal=TRUE`, otherwise R will compute a different version of the t-test.

You can use different syntax to run the t-test. Because our data is already in a data frame we can use this syntax.

```
t.test(Intellect_Rating~CONDITION, data=all_data, var.equal=TRUE)

##
## Two Sample t-test
##
## data: Intellect_Rating by CONDITION
## t = -3.5259, df = 37, p-value = 0.001144
## alternative hypothesis: true difference in means between group transcript and group
## 95 percent confidence interval:
## -3.1284798 -0.8450652
## sample estimates:
## mean in group transcript      mean in group audio
##                 3.648148                  5.634921
```

The `t.test` function also will work on two variables, not in a data frame. For example, the following does the same thing. But, it's harder to read, and the means are described in terms of X and Y, not terms of `transcript` and `audio`, like the report above.

```
t.test(all_data[all_data$CONDITION=='transcript',]$Intellect_Rating,
      all_data[all_data$CONDITION=='audio',]$Intellect_Rating,
      var.equal=T)

##
## Two Sample t-test
##
```

```
## data: all_data[all_data$CONDITION == "transcript", ]$Intellect_Rating and all_data[all_data$C
```

t = -3.5259, df = 37, p-value = 0.001144
 ## alternative hypothesis: true difference in means is not equal to 0
 ## 95 percent confidence interval:
 ## -3.1284798 -0.8450652
 ## sample estimates:
 ## mean of x mean of y
 ## 3.648148 5.634921

Question: What conclusions do we draw from the t-test? Based on these results, if you were being evaluated for a job interview, would you rather have the evaluator read a transcript of your interview or listen to an audio recording?

So, now we have the t-test. It shows the t-value, the p-value, and the means for each group. You can double-check with the paper to see if we found the same results as reported by the authors.

7.4.6 Remaining ratings

Now, you should use what you have learned to analyse the last two ratings for the dependent variables `Impression_Rating`, and `Hire_Rating`. This is your task for the generalization exercise. Remember to plot the data for each, and conduct a t-test for each. Then compare what you found to the original article. What did you find, and what do the results mean?

7.4.7 Reconstructing the graph from the paper

The results from Experiment 4 in the paper plot the means and error bars (+1 / -1 SEM) for all three dependent measures, for both experimental conditions. We can do this in ggplot using the data. We will have to make a couple changes to the data frame. But, it won't be too hard. What we need to do is make a fully long form data frame. Remember a long form data frame has one row per dependent measure.

The `all_data` frame is partly long and partly wide. If we are only interested in one dependent measure, then it is a long data frame for that measure. For example, example if we are only interested in plotting `Intellect_Rating`, then we already have one observation of that dependent measure for each row. But, in the other columns, the dependent measures for `Impression_Rating` and `Hire_Rating` are in the same rows.

Before continuing, it is very much worth mentioning that this part of data analysis happens a lot, and it is kind of annoying. I call it the rubix cube problem, because we need to “rotate” and transform the format of the data to accomplish different kinds of analysis goals. It's good to be able to know how to do this. This problem occurs all of the time, and can occur for any software package. It's a good thing you are learning R, because we can do these things easily in R. They are not often so easy to do without a computer programming

language like R. The worst thing to do is transform the data by hand. That really sucks. Believe me you don't want to do it. Why? Because you will make mistakes, and you will mess up the data, then you will mess up your analysis. And, you won't be able to find your mistakes, and it will take you ages to correct them. That sucks.

There's more than one way to transform data in R. For example the `cast` and `melt` functions do this kind of thing. You can look those up. In this example we will not use those functions. Instead we will show some steps to build the required data frame one step at a time.

```
# repeat CONDITION column three times

condition <- rep(all_data$CONDITION,3)

# make a ratings variable with all three ratings in one variable

ratings <- c(all_data$Intellect_Rating,
              all_data$Impression_Rating,
              all_data$Hire_Rating)

# make a new factor variable with the names of the ratings
# need to repeat each level name the appropriate number of times

num_to_repeat <- length(all_data$CONDITION)

rating_type <- rep(c("Intellect","Impression","Hire"),num_to_repeat)

# put the new variables into a data frame

plot_all <- data.frame(condition,rating_type,ratings)

# Get the means and standard errors for each rating by condition

descriptive_all <- plot_all %>%
  group_by(condition,rating_type) %>%
  summarise(means= mean(ratings),
            SEs = sd(ratings)/sqrt(length(ratings)))

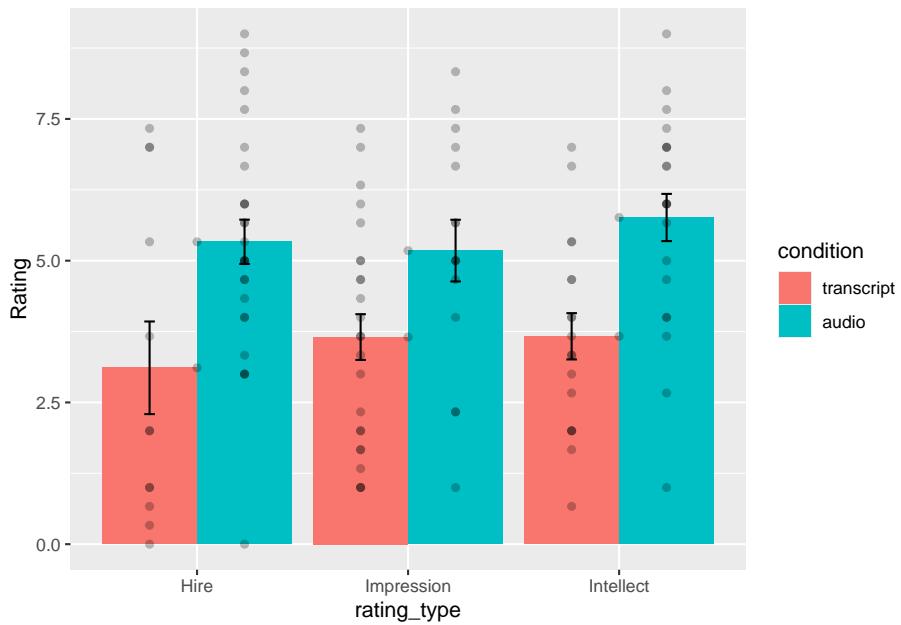
# Make the plot

ggplot(descriptive_all, aes(x=rating_type, y=means, group=condition))+
  geom_bar(stat="identity", aes(fill=condition), position='dodge')+
  geom_errorbar(aes(ymin=means-SEs,
                     ymax=means+SEs),
                width=.1,
```

```

position = position_dodge(width=.9)) +
geom_point(data=plot_all, aes(x=rating_type,
                               y=ratings,
                               group=condition),
           alpha=.25,
           position = position_dodge(width=.9))+ 
geom_point(alpha=.25) +
ylab("Rating")

```



Well, we didn't make the exact graph. We have the bars, the error bars, and we added the individual scores because they are useful to look at. Otherwise, it's the same graph (except the the ordering of bars is determined alphabetically here. We change that in ggplot, but we won't do that today.)

7.4.8 Generalization Exercise

(1 point - Pass/Fail)

Complete the generalization exercise described in your R Markdown document for this lab.

Now, you should use what you have learned to analyse the last two ratings for the dependent variables `Impression_Rating`, and `Hire_Rating`. Report the t-tests and means for each.

7.4.9 Writing assignment

(2 points - Graded)

Complete the writing assignment described in your R Markdown document for this lab. When you have finished everything. Knit the document and hand in your stuff (you can submit your .RMD file to blackboard if it does not knit.)

1. Answer the following questions
 - a. Explain the difference between a paired-samples t-test and an independent samples t-test. (1 point)
 - b. Imagine you were to conduct a between-subjects experiment with two groups, where the manipulation had no effect whatsoever. If you repeated this experiment (that doesn't work) 100 times and computed a t-test each time, approximately how many of the experiments out of 100 would you expect find the associated p-value is .05 or smaller? Explain (1 point)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

7.5 Excel

How to do it in Excel

7.6 SPSS

In this lab, we will use SPSS to:

1. Perform an independent-samples t-test
2. Graph the data

7.6.1 Experiment Background

Schroeder and Epley (2015) conducted an experiment to determine whether a person's speech (i.e., vocal tone, cadence, and pitch) communicates information about their intellect better than their written words (even if they are the same words as in the speech).

To conduct this study, the authors randomly assigned 39 professional recruiters for Fortune 500 companies to one of two conditions. In the audio condition,

participants listened to audio recordings of a job candidate's spoken job pitch. In the transcript condition, participants read a transcription of the job candidate's pitch. After hearing or reading the pitch, the participants rated the job candidates on three dimensions: intelligence, competence, and thoughtfulness. These ratings were then averaged to create a single measure of the job candidate's intellect, with higher scores indicating the recruiters rated the candidates as higher in intellect. The participants also rated their overall impression of the job candidate (a composite of two items measuring positive and negative impressions). Finally, the participants indicated how likely they would be to recommend hiring the job candidate (0 - not at all likely, 10 - extremely likely).

So, what happened? Did the recruiters think job applicants were smarter when they read the transcripts, or when they heard the applicants speak? We have the data, we can find out.

7.6.2 Performing an independent-samples t-test

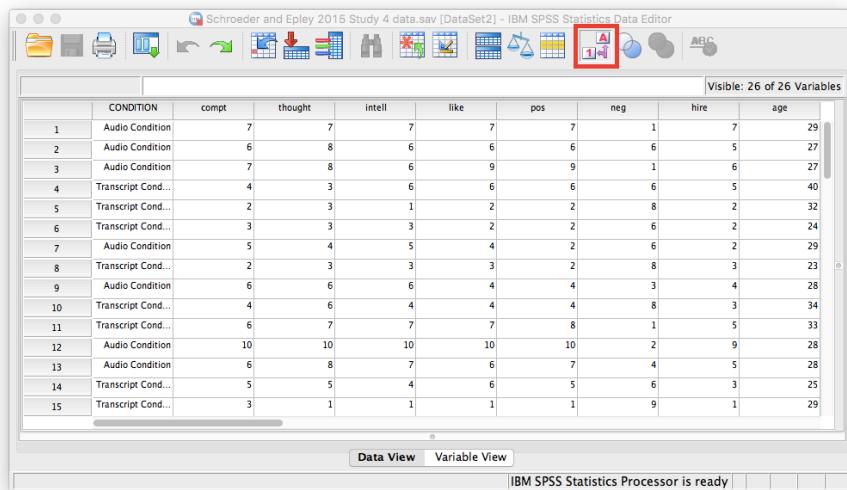
First, let's open the relevant data file; Here is the link. It's called "SchroederEpley2015_Study4_data.sav" Open this file in SPSS. Your data should look like this:

	CONDITION	compt	thought	intell	like	pos	neg	hire	age
1	Audio Condition	7	7	7	7	7	1	7	29
2	Audio Condition	6	8	6	6	6	6	5	27
3	Audio Condition	7	8	6	9	9	1	6	27
4	Transcript Cond...	4	3	6	6	6	6	5	40
5	Transcript Cond...	2	3	1	2	2	8	2	32
6	Transcript Cond...	3	3	3	2	2	6	2	24
7	Audio Condition	5	4	5	4	2	6	2	29
8	Transcript Cond...	2	3	3	3	2	8	3	23
9	Audio Condition	6	6	6	4	4	3	4	28
10	Transcript Cond...	4	6	4	4	4	8	3	34
11	Transcript Cond...	6	7	7	7	8	1	5	33
12	Audio Condition	10	10	10	10	10	2	9	28
13	Audio Condition	6	8	7	6	7	4	5	28
14	Transcript Cond...	5	5	4	6	5	6	3	25
15	Transcript Cond...	3	1	1	1	1	9	1	29

For our analysis, we will focus on only one of the three measures mentioned above: **intellect**. We want to know if perceived intellect is different in the audio condition (where recruiters listened to a job pitch) than in the transcript condition (where recruiters read a transcript of a job pitch).

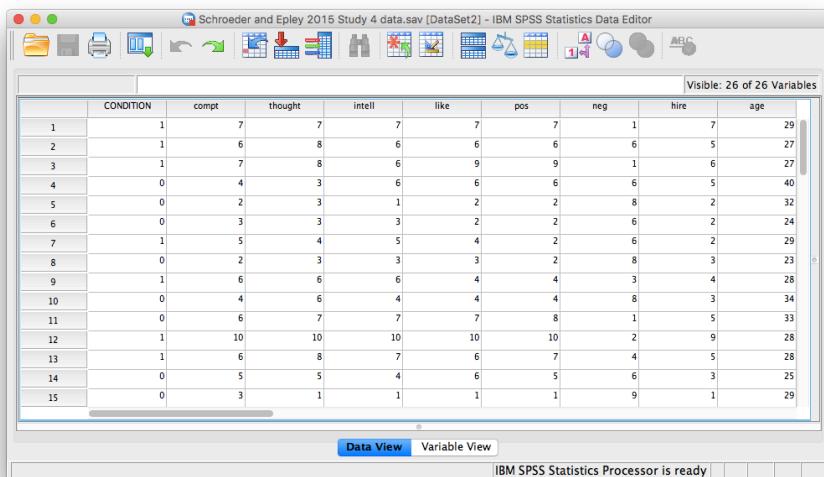
Before we run the t-test, let's introduce a button at the top of our SPSS spread-

sheet window. This will be important for identifying the values assigned to groups later on. The button I refer to here is highlighted below; it has two boxes, one with an A and one with a 1 on it, with arrows in-between.



	CONDITION	compt	thought	intell	like	pos	neg	hire	age
1	Audio Condition	7	7	7	7	7	1	7	29
2	Audio Condition	6	8	6	6	6	6	5	27
3	Audio Condition	7	8	6	9	9	1	6	27
4	Transcript Cond...	4	3	6	6	6	6	5	40
5	Transcript Cond...	2	3	1	2	2	8	2	32
6	Transcript Cond...	3	3	3	2	2	6	2	24
7	Audio Condition	5	4	5	4	2	6	2	29
8	Transcript Cond...	2	3	3	3	2	8	3	23
9	Audio Condition	6	6	6	4	4	3	4	28
10	Transcript Cond...	4	6	4	4	4	8	3	34
11	Transcript Cond...	6	7	7	7	8	1	5	33
12	Audio Condition	10	10	10	10	10	2	9	28
13	Audio Condition	6	8	7	6	7	4	5	28
14	Transcript Cond...	5	5	4	6	5	6	3	25
15	Transcript Cond...	3	1	1	1	1	9	1	29

Press this button and see what happens to your data spreadsheet:

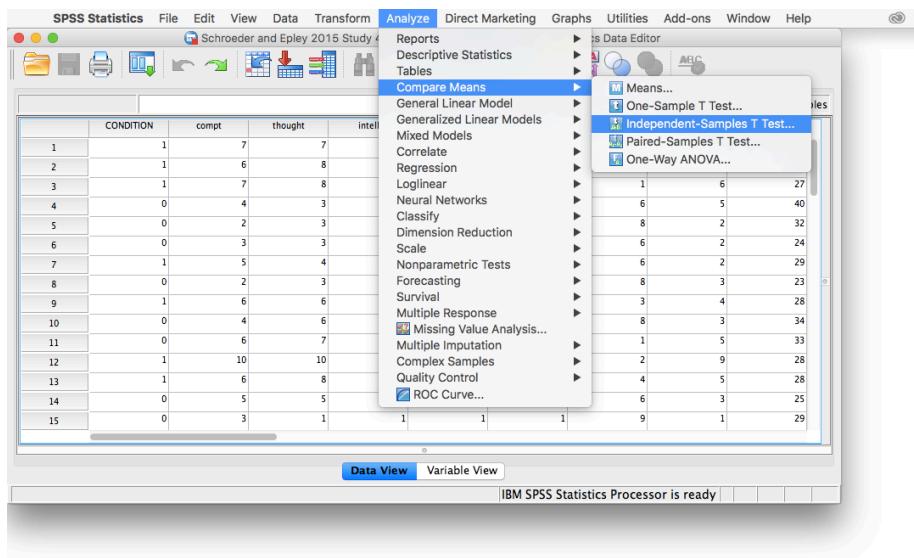


	CONDITION	compt	thought	intell	like	pos	neg	hire	age
1	1	7	7	7	7	7	1	7	29
2	1	6	8	6	6	6	6	5	27
3	1	7	8	6	9	9	1	6	27
4	0	4	3	6	6	6	6	5	40
5	0	2	3	1	2	2	8	2	32
6	0	3	3	3	2	2	6	2	24
7	1	5	4	5	4	2	6	2	29
8	0	2	3	3	3	2	8	3	23
9	1	6	6	6	4	4	3	4	28
10	0	4	6	4	4	4	8	3	34
11	0	6	7	7	7	8	1	5	33
12	1	10	10	10	10	10	2	9	28
13	1	6	8	7	6	7	4	5	28
14	0	5	5	4	6	5	6	3	25
15	0	3	1	1	1	1	9	1	29

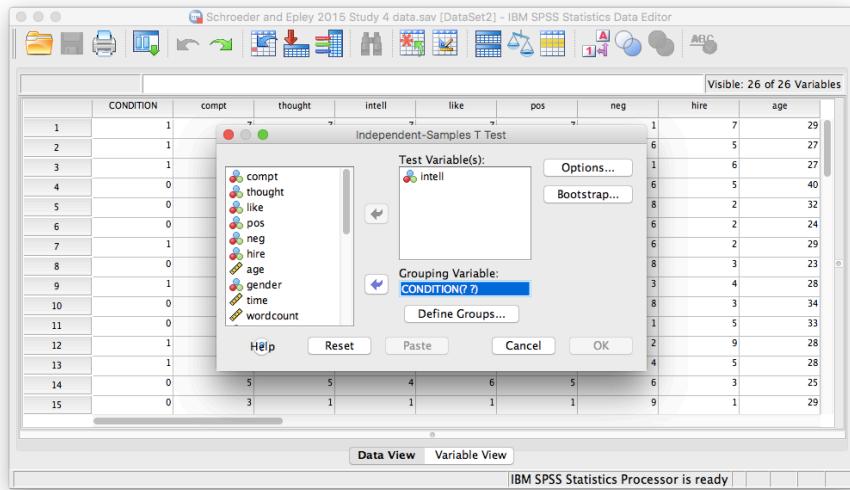
Notice that this button has revealed the values used to signify the Audio and Transcript groups. We will need to know that these groups are signified by 0s and 1s when we come to running our t-test. If you click the button again, the cells will go back to displaying their labels. Keep in mind that revealing the

values or labels does not change anything about the data itself or any subsequent analyses; it is merely a cosmetic change.

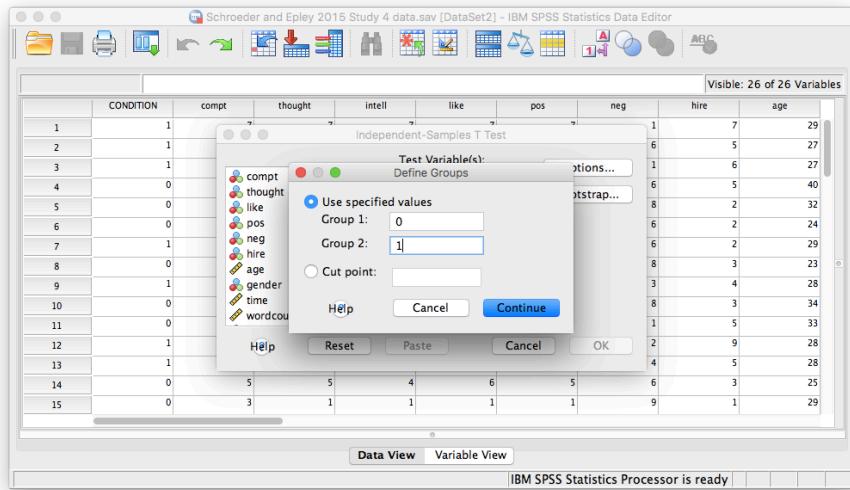
Now, we're ready to run the independent-samples t-test. Go to Analyze, then Compare Means, then Independent Samples T-test...



A window will appear asking you to specify which variable to use in this analysis. Remember, we are only using `intell`, so find this variable in the left-hand list and move it into the “Test Variable(s)” field on the right using the arrow. For the field labeled “Grouping Variable”, we will specify which variable is our independent variable. That variable is `Condition` (we have two conditions: audio and transcript). Move `Condition` into the “Grouping Variable” field. Notice that next to the variable name there are two question marks and the OK button is disabled:



Here, SPSS wants to know the values used to signify these conditions (remember the 0s and 1s?). So, click Define Groups, and place 0 and 1 in the fields that ask for Values for Group 1 and Group 2.



Click Continue, and then OK. SPSS will produce output tables as follows:

T-Test

Group Statistics				
	CONDITION	N	Mean	Std. Deviation
intell	Transcript Condition	18	4.39	2.173
	Audio Condition	21	6.43	1.434

Independent Samples Test										
	Levene's Test for Equality of Variances	t-test for Equality of Means								
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
		4.795	.035	-3.505	37	.001	-2.040	.582	-3.219	-.861
intell	Equal variances assumed			-3.398	28.668	.002	-2.040	.600	-3.268	-.811
	Equal variances not assumed									

The t-test output is very similar to our previous t-test examples and can be interpreted similarly. In the case of an independent-samples t-test, there is one special consideration to be made: the equality of variances. One of the assumptions of a t-test is that the groups being compared have roughly equal variances. Conveniently, the independent-samples t-test performs another test within itself to determine if the variances are equal or not: this test is called the Levene's test.

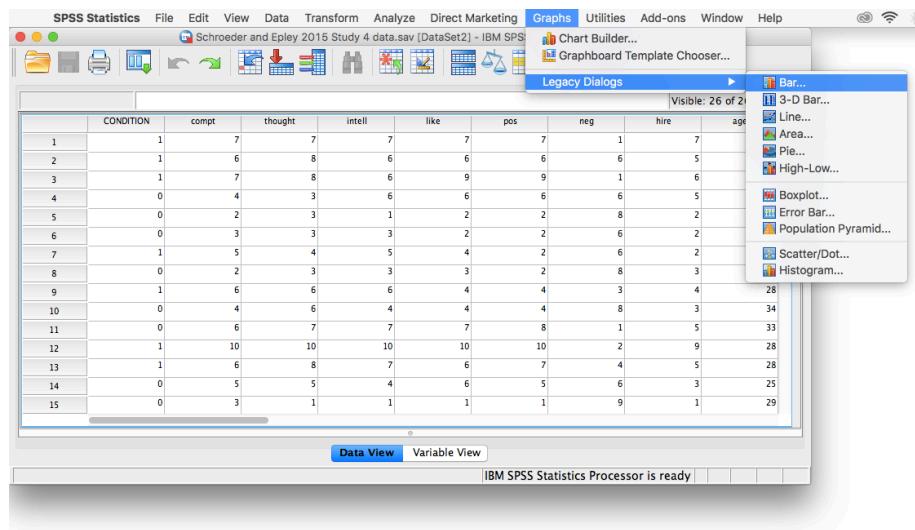
In the second output table, there is a section dedicated to the Levene's test. The rule is as follows: If the Sig. value under the Levene's test heading is smaller than your alpha level, then your variances are NOT equal and you must refer to the values in the second row of the table, labeled "Equal variances not assumed." If the Sig. value for Levele's test is larger than alpha, then you can use the top row labeled "Equal variances assumed."

In this case, the Sig. value associated with the Levene's test is .035, which is less than the conventional alpha level of .05. This means that the t-statistic for our independent-samples t-test is -3.398, the degrees of freedom are adjusted to 28.668, and the p-value for the t-test is .002. What does this mean? We can put our finding into words to clarify:

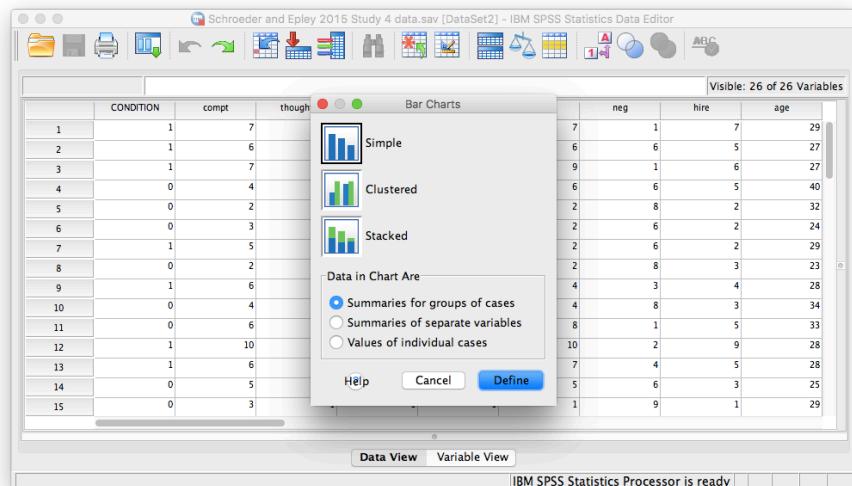
Recruiter ratings for audio job pitches are 2.04 points higher than ratings for transcripts of pitches. This difference is significant, $t(28.668) = -3.398$, $p < .05$.

7.6.3 Graphing your data

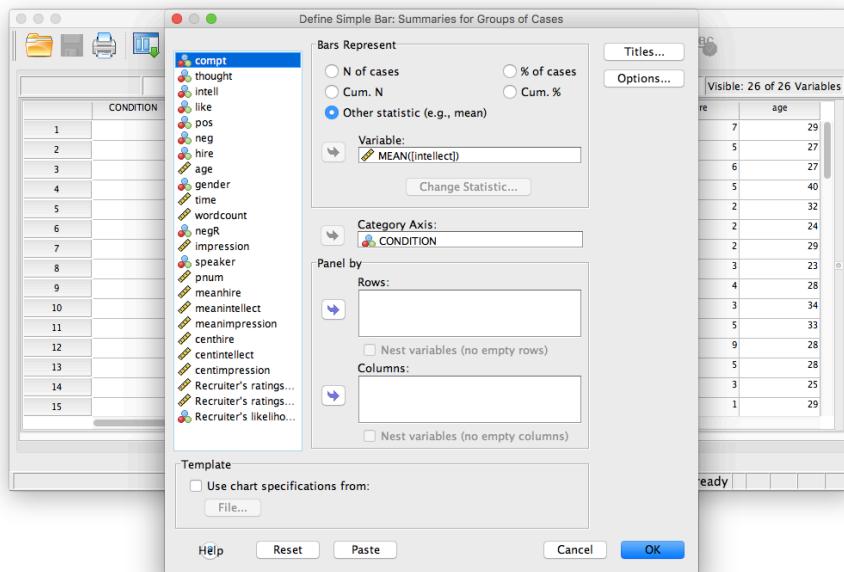
Condition in this experiment represents a discrete, categorical variable. Therefore, we will be using a bar chart to visualize this data. To begin, go to Graphs, then Legacy Dialogs, then Bar...



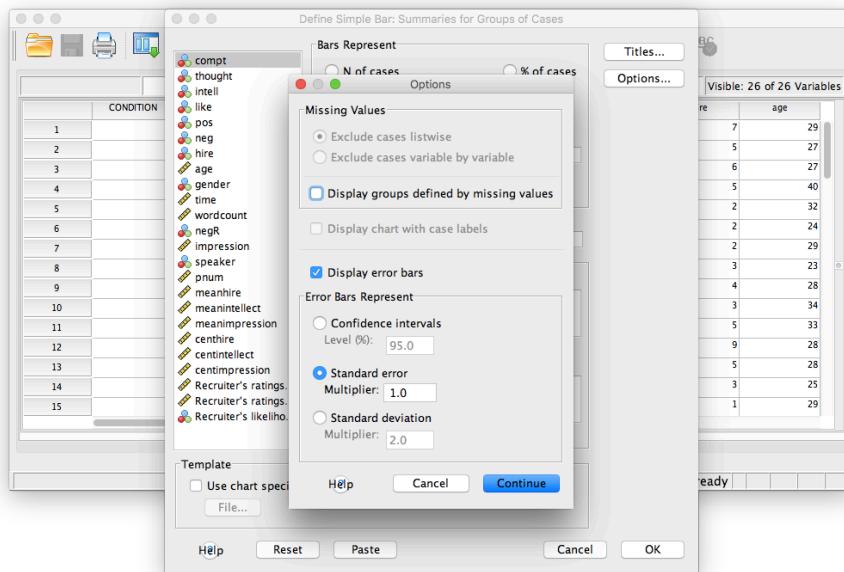
Next, choose Simple in the window that appears, then click Define.



In the next window, make sure that under “Bars Represent”, you have chosen Other statistic. Move your variable of interest (`intellect`) into the field labeled “Variable.” Then, move `CONDITION` to the “category axis” field.



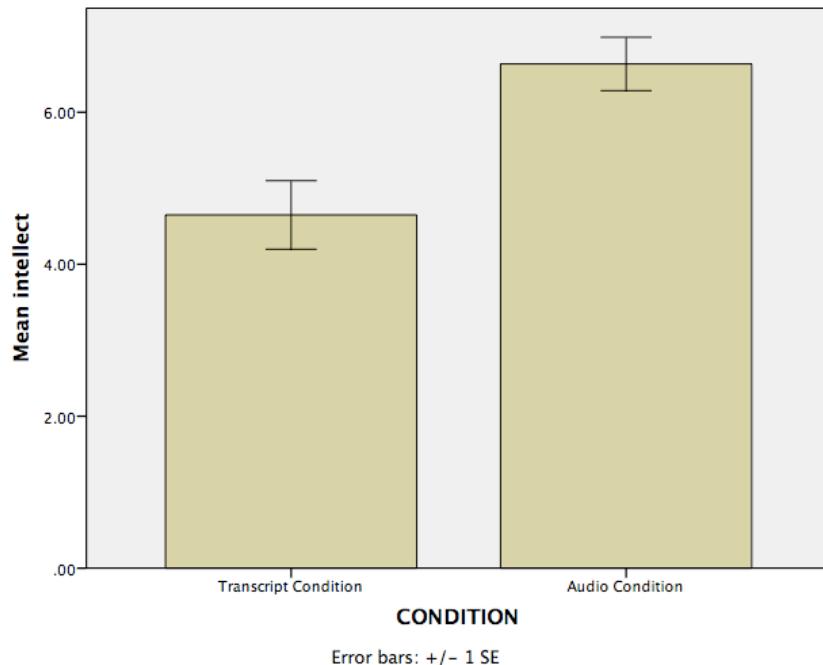
Next, click Options. Then choose Display Error Bars, and have them represent Standard Error with a multiplier of 1.



Click Continue, then OK.

SPSS will produce a bar graph of your mean ratings for both audio and transcript conditions.

Graph



7.6.4 Practice Problems

-
1. Use the same data file from this lab's tutorial to test whether intelligence ratings were different between genders (use alpha = .05). Who was rated as more intelligent, males or females? Report your result in proper statistical reporting format.
 2. Graph this result (including 1 SEM error bars) as a bar graph.

7.7 JAMOVI

How to do it in JAMOVI

Chapter 8

Lab 8 One-way ANOVA

The analysis of variance is not a mathematical theorem, but rather a convenient method of arranging the arithmetic. —R. A. Fisher

This lab is modified and extended from Open Stats Labs. Thanks to Open Stats Labs (Dr. Kevin P. McIntyre) for their fantastic work.

8.1 How to not think about bad memories by playing Tetris

This lab activity uses the open data from Experiment 2 of James et al. (2015) to teach one-way ANOVA with planned comparisons. Results of the activity provided below should exactly reproduce the results described in the paper.

8.1.1 STUDY DESCRIPTION

Following traumatic experiences, some people have flashbacks, which are also called “intrusive memories” and are characterized by involuntary images of aspects of the traumatic event. Although people often try to simply forget traumatic memories, this approach is not very effective. Instead, previous research suggests that a better approach may be to try to change aspects of the memory after it is formed. For example, some research shows that traumatic memories can be altered and weakened to the point that they are no longer intrusive.

Because intrusive memories of trauma are often visual in nature, James and colleagues (2015) sought to explore whether completing a visuospatial task (e.g., Tetris) after a memory was formed would interfere with the storage of that memory, and thereby reduce the frequency of subsequent intrusions. They hypothesized that only participants who complete a visuo-spatial task after reactivation of the traumatic memories would experience a reduction in intrusive memories.

In comparison, simply completing a visuo-spatial task (without reactivation) or reactivation (without a visuo-spatial task), would not reduce the occurrence of intrusive memories.

In other words, if you play Tetris shortly after you were remembering bad memories, playing Tetris might weaken those memories, which could cause you to experience those kinds of intrusive memories less often in the future.

8.1.2 Study Methods

To test their hypothesis, the authors conducted an experiment ($N = 72$, $n = 18$ per condition). The procedure is summarized as follows:

Trauma Film: All participants viewed a series of video clips of graphic violence (e.g., a person getting hit by a van while using his phone as he crosses the road) as a way to create memories that should become intrusive memories. Participants then went home and recorded the number of intrusive memories they experienced over the next 24 hours. Because this is before the experimental manipulations, all groups were predicted to have an equal occurrence of intrusive memories during the first 24-hours (called Day 0).

Experimental Task: After this 24-hour period, the participants returned to the lab and completed the experimental task. The experimenters randomly assigned participants to ONE of the following conditions:

1. No-task control: These participants completed a 10-minute music filler task.
2. Reactivation + Tetris: These participants were shown a series of images from the trauma film to reactivate the traumatic memories (i.e., reactivation task). After a 10-minute music filler task, participants played the video game Tetris for 12 minutes.
3. Tetris Only: These participants played Tetris for 12 minutes, but did not complete the reactivation task.
4. Reactivation Only: These participants completed the reactivation task, but did not play Tetris.

Intrusive Memories: All participants were asked to record the number of intrusive memories that they experienced over the next seven days (Days 1 to 7).

After the seven days had passed, participants completed an Intrusion-Provocation Task, in which they were shown blurred images from the trauma film and asked to indicate whether the blurred image triggered an intrusive memory.

8.2 Lab Skills Learned

8.3 Important Stuff

- citation: James, E. L., Bonsall, M. B., Hoppitt, L., Tunbridge, E. M., Geddes, J. R., Milton, A. L., & Holmes, E. A. (2015). Computer game play reduces intrusive memories of experimental trauma via re-consolidation-update mechanisms. *Psychological Science*, 26, 1201-1215.
- Link to .pdf of article
- Data in .csv format
- Data in SPSS format

8.4 R

8.4.1 Load the data

Remember that any line with a # makes a comment and the code does not run. Below is how to load the .csv data from the online repository, or from a local file (you need to change the file path to where the local file is, if you downloaded it). The data contains all of the measures and conditions from Experiment 2 in the paper.

```
library(data.table)
#fread("https://raw.githubusercontent.com/CrumpLab/statisticsLab/master/data/Jamesetal2015Experiment2.csv")
all_data <- fread("data/Jamesetal2015Experiment2.csv")
```

8.4.2 Inspect the dataframe

This will give you a big picture of the data frame. Click the button to view it in your browser, then take a look to see what is in it.

```
library(summarytools)
view(dfSummary(all_data))
```

8.4.3 Get the data you need

Again we have only the data from Experiment 2, so we don't need to get rid of any rows of the data. But we do need look at the columns to see how the independent variable and dependent variables were coded.

8.4.3.1 The independent variable

There was one important independent variable, and it had four levels. The first column in the data frame is called condition, and it has four levels. The levels are 1, 2, 3, and 4 in the data-frame. These will correspond to the four levels in shown in figure:

- No-task control
- Reactivation Plus Tetris
- Tetris only
- Reactivation only

Each of these refer to what subjects did after they watched the traumatic film. But, which of these correspond to the numbers 1 to 4 in the data-frame? It turns out the are in the right order, and 1 to 4 refer to:

1. No-task control
2. Reactivation Plus Tetris
3. Tetris only
4. Reactivation only

Let's do ourselves a favor and rename the levels of the `Condition` column with words so we know what they refer to. First, convert the `Condition` column to a factor, then rename the levels. Then.

```
all_data$Condition <- as.factor(all_data$Condition)
levels(all_data$Condition) <- c("Control",
                                "Reactivation+Tetris",
                                "Tetris_only",
                                "Reactivation_only")
```

8.4.3.2 The dependent variable

The authors showed two figures, one where they analysed intrusive memory frequency as the mean for the week, and the other where the used intrusive memory frequency on Day 7. In this tutorial, we will do the steps to run an ANOVA on the mean for the week data, and you will do follow these steps to run another ANOVA on the Day 7 data.

The mean for the week data for each subject is apparently coded in the column `Days_One_to_Seven_Number_of_Intrusions`.

8.4.4 Look at the data

Remember before we do any analysis, we always want to “look” at the data. This first pass let's us know if the data “look right”. For example, the data file could be messed up and maybe there aren't any numbers there, or maybe the numbers are just too weird.

In the last two labs, we saw how to show all the data we are interested in looking at in one go. We can do the same things here. For example, in one little piece of code, we can display the means in each condition, the standard errors of the mean, and the individual scores for each subject in each condition. This is a lot to look at, but it's everything we want to look at for starters all in the same place. Looking at the data this way will also give us intuitions about what our ANOVA will tell us before we run the ANOVA. This is helpful for determining

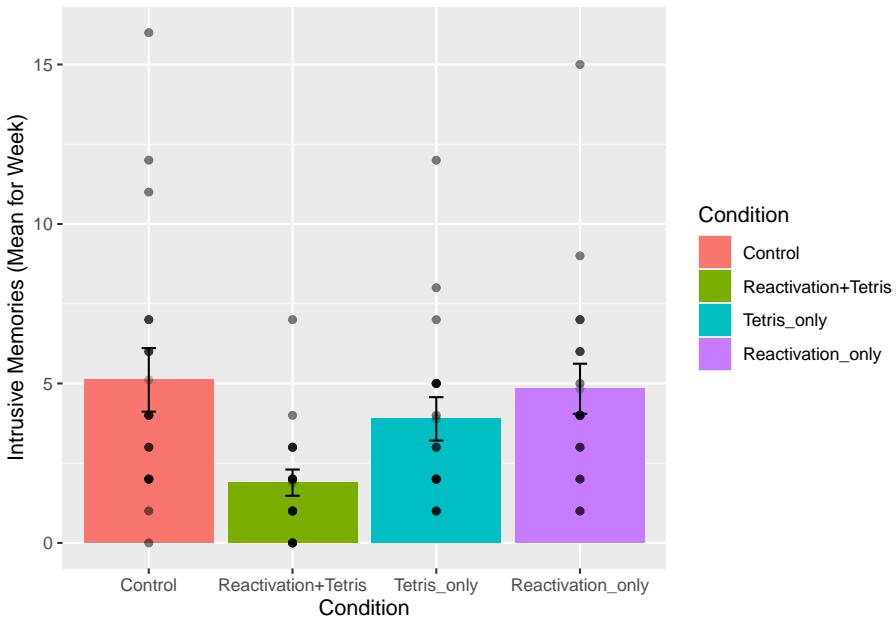
whether the results of your ANOVA are accurate or not. Let's do it...also notice that the code is very similar to what we did for the independent t-test. In fact, all I did was copy and paste that code right here, and edited it a little bit. This is really fast, and shows the efficiency of doing things this way.

```
library(dplyr)

library(ggplot2)

# get means and SEs
descriptive_df <- all_data %>%
  group_by(Condition) %>%
  summarise(means= mean(Days_One_to_Seven_Number_of_Intrusions),
            SEs = sd(Days_One_to_Seven_Number_of_Intrusions)/sqrt(length(Days_0))

# Make the plot
ggplot(descriptive_df, aes(x=Condition, y=means))+
  geom_bar(stat="identity", aes(fill=Condition))+ # add means
  geom_errorbar(aes(ymin=means-SEs, # add error bars
                     ymax=means+SEs), width=.1) +
  geom_point(data=all_data, aes(x=Condition, y=Days_One_to_Seven_Number_of_Intrusions), alpha=.5)
  geom_point(alpha=.25)+
  ylab("Intrusive Memories (Mean for Week)")
```



There we have it. A nice graph showing us everything. We can see there was a lot of differences at the level of individual subjects. Some subjects had a mean

of 15 intrusive memories for the week. Some had 0. Lots of differences.

And, to be completely honest, this data seems a bit weird to me. It might not be weird, but it might be. The wording in the manuscript is that this data is the mean frequency of intrusive memories over the week. For the people who had 15, this means that every day they had on average 15 intrusive memories. Some people had on average 0 per day. This could be the data. But, it also seems like the data might just be frequency counts and not means at all. For example, the data could be that some people had a total of 15 intrusive over the week. The data might make more sense if these were frequency counts. Otherwise the differences between people are truly very large. For example the person who had an average of 15 intrusive memories per week, must have had $15 \times 7 = 105$ intrusive memories, which is a lot more than zero. In any case, this kind of wondering about the data is what happens when you start to notice how numbers work. It's useful develop your data sense.

Let's move on to the ANOVA. By looking at the above graph do you have an intuition about what the ANOVA will tell us? I do, it should easily tell us that we are going to get an F-value that is bigger than one, and a p-value that is probably smaller than .05? How do I know this? I looked at the error bars, which represent the standard errors of the mean. If we look across conditions we can see that that the error bars don't always overlap. This suggests there are differences in the data that don't happen very often by chance. So, we expect a smallish p-value. Why do you think I expect the F-value to be greater than 1? If you can answer this question with a justification and explanation of how F works, pat yourself on the back!

8.4.5 Conduct the ANOVA

Conducting an ANOVA in R is pretty easy. It's one line of code, just like the t-test.

It's worth knowing that there are a few different packages out there to do an ANOVA, for example Mike Lawrence's `ezANOVA` package is pretty popular.

For this tutorial we'll show you how to run the ANOVA using the `aov` function from base R (comes pre-installed). This is pretty "easy" too. I put easy in quotes because the first time I saw this, it was not easy for me. Here's a brief digression for me to tell you that I feel your pain. I was first introduced to R by Patrick Bennett (a professor at McMaster University, where I attended graduate school). Pat, like I am doing now, forced us to use R in his statistics class. I remember having zero clue about so many things, and was often very frustrated. So, I imagine some of you are quite frustrated too. I was luck, like some of you, to have had some previous experience with other programming languages, so I was familiar with what R might be doing. What I was most frustrated with, was learning how to tell R what to do. In other words, I didn't know how to write the commands properly. I didn't understand what we call the **syntax** of R.

This was back many years ago now, well before there was so many helpful examples on Google with working code, showing you how the syntax works. All we had was the R help files, which were a total mystery to me. If you want to see the help file for `aov`, just type `?aov()` into the console, and press enter. You will see an “explanation” of how the `aov` function is supposed to work. You can use the same trick for any R function, like this `?name_of_function()`. To be clear, you have to replace the letters `name_of_function`, with the name of the function. Some of you think that might be super obvious, but that is the kind of thing I did not think was obvious. So, when I read the help file for how to use the `aov` function, that is to learn what to put where, I didn’t feel like it was showing me what I needed to do. Usually, at the bottom of the help file, there are some examples, and these are helpful, but sometimes they are missing the example you need, and you are expected to generalize your knowledge of how `aov` works, to make it work for your problem. This is a catch-22 because if you don’t know how it works, you can’t generalize. IMO, you need a lot of examples of things that work.

So, with that digression, I’m going to explain the syntax for the `aov` function. It looks like this:

```
aov(DV ~ IV, dataframe)
```

That probably looks really foreign. Let me explain. First you need write `aov()`. `aov` is the name of the function, followed by the brackets. The brackets are a sandwich. Sandwiches have a top and a bottom, and they enclose the things you put in the sandwich. We then put things inside the `()`, in specific ways to make the `aov` sandwich. The `DV` stands for the name of the dependent variable in the data frame. For us, this will be `Days_One_to_Seven_Number_of_Intrusions`. So, when we add that, our function will look like:

```
aov(Days_One_to_Seven_Number_of_Intrusions ~ IV, dataframe)
```

Next, the `~` (tilda) stands for the word ‘by’. We are saying we want to analyse the Dependent variable **by** the conditions of the independent variable.

The `IV` stands for the name of the column that is your independent variable. Our’s is called **Condition**. Adding that in, our formula looks like:

```
aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, dataframe)
```

Finally, the last part is the name of the data-frame we are using. The `aov` function only works on long-form data, where each score on the dependent variable has its own row. Our’s is already set up this way! The name of our data-frame is `all_data`. We add that in, and it looks like:

```
aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, all_data)
```

In English, this means, do an ANOVA on the dependent variable as a function of the independent variable, and use the data in my data frame.

This is it for now. The `aov` function is very flexible because you can define

different kinds of formulas (the DV ~ IV part). We'll see other examples in later chapters. For now, this is all we need. Also, what's cool, is that this will work for any single IV with any number of levels (conditions) 2 or greater. Fun. Let's see it in action.

```
aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, all_data)

## Call:
##   aov(formula = Days_One_to_Seven_Number_of_Intrusions ~ Condition,
##     data = all_data)
##
## Terms:
##           Condition Residuals
## Sum of Squares    114.8194  685.8333
## Deg. of Freedom      3          68
##
## Residual standard error: 3.175812
## Estimated effects may be unbalanced
```

What is this garbage? I don't see an ANOVA table, what is this? You are seeing the raw print out of the `aov` function. Clearly, this is not helpful, it's not what we want to see.

Fortunately, R comes with another function called `summary`. What it does is summarize the results of functions like `aov`, and prints them out in a nicer way. Let's see the `summary` function do it's thing:

```
summary(aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, all_data))

##           Df Sum Sq Mean Sq F value Pr(>F)
## Condition     3 114.8    38.27   3.795 0.0141 *
## Residuals    68 685.8    10.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alright, now we have an ANOVA table we can look at. However, it still looks ugly, at least to me. When you are working inside an R Markdown document, you have some more options to make it look nice. We can use the `kable` and `xtable` function together, like this.

```
library(xtable)
aov_out<-aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, all_data)
summary_out<-summary(aov_out)

knitr::kable(xtable(summary_out))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Condition	3	114.8194	38.27315	3.794762	0.0140858
Residuals	68	685.8333	10.08578	NA	NA

Now we see a nicer print out. Especially if we `knit` the document into a webpage.

8.4.6 Reporting the ANOVA results

Refer tot the textbook on ANOVAs for a deeper discussion of all the things in the ANOVA table. We'll remind about some of those things here.

First, let's look at how we might report the results. There are three very important parts to this.

1. Saying what test you did to what data
2. Reporting the inferential statistics
3. Reporting the pattern in the means

Here is part 1, we need to say what data we used, and what kind of test we used on that data:

We submitted the mean intrusive memories for the week from each subject in each condition to a one-factor between-subjects ANOVA, with Intervention type (No-task control, Reactivation Plus tetris, Tetris only, Reactivation only) as the sole independent variable.

Part 2 is saying what the results of the test were. Specifically, we report the values from the inferential test (see the textbook for why these values). Also, you should be able to answer this question: why do we report the values that we do?

We found a main effect of Intervention type, $F(3, 68) = 3.79$, $MSE = 10.09$, $p = 0.014$.

Part 3 is saying what the pattern was in the means. Remember, that in the ANOVA, a significant effect refers to the global variation among the means. In other words, we can say that there are some differences between the means, but we can't specifically say which pairs of means are different, or which groups of means are different from one another. How can we report this, where are the means? In fact, we already found them when we plotted the data earlier. So, we can copy paste that code, and print out the means, rather than the figure:

```
# get means and SEs
descriptive_df <- all_data %>%
  group_by(Condition) %>%
  summarise(means= mean(Days_One_to_Seven_Number_of_Intrusions),
            SEs = sd(Days_One_to_Seven_Number_of_Intrusions)/sqrt(length(Days_One_to_Seven_Number_of_Intrusions)))
```

`knitr::kable(descriptive_df)`

Condition	means	SEs
Control	5.111111	0.9963623
Reactivation+Tetris	1.888889	0.4113495
Tetris_only	3.888889	0.6806806
Reactivation_only	4.833333	0.7848650

No we have to use a sentence to describe these means.

Refer to table 1 for the means and standard errors of the mean in each condition

or,

Mean intrusive memories per week were 5.11 (SE = .99); 1.89 (SE = .41); 3.89 (SE = .68); and 4.83 (SE= .78), in the Control, Reaction plus Tetris, Tetris Only, and Reactivation only conditions, respectively

Ugh, what a mouthful. Be mindful of how you write results. The above is not helpful because you see a list of numbers, and then a list of conditions, and the reader has to do a lot of work to keep everything straight. I like the table option.

I also like this other kind of option:

Mean intrusive memories were significantly different between the Control ($M = 5.11$, SE = .99), Reactivation plus Tetris ($M = 3.89$, SE = .68), Tetris only ($M= 3.89$, SE = .68), and Reactivation only ($M = 4.83$, .78) conditions.

That's a little bit better. Let's put it all in one place to see what it look like:

We submitted the mean intrusive memories for the week from each subject in each condition to a one-factor between-subjects ANOVA, with Intervention type (No-task control, Reactivation Plus tetris, Tetris only, Reactivation only) as the sole independent variable. We found a main effect of Intervention type, $F(3, 68) = 3.79$, $MSE = 10.09$, $p = 0.014$. Mean intrusive memories were significantly different between the Control ($M = 5.11$, SE = .99), Reactivation plus Tetris ($M = 3.89$, SE = .68), Tetris only ($M= 3.89$, SE = .68), and Reactivation only ($M = 4.83$, .78) conditions.

8.4.7 Individual comparisons

This next part is complicated, so we intentionally simplify it. There are these things (remember from the textbook), called comparisons. We use them to compare differences between specific conditions of interest. That part isn't very complicated. You just pick the things you want to compare, and compare them.

What is complicated is exactly what you “should” do to make the comparison. It turns out there are lots of recommendations and disagreements about what

you should do. There are also lots of tests that you can do, so you have a lot of options. We are not going to show you here all of the tests. This is beyond the scope of what we are trying to teach you. Instead, we will use tests that you already know, the t-test for independent samples from the last lab. It will do the job for this data.

We think that before you can make good decisions about the kind of comparison test you want to use, you have to have a solid understanding of **what you are comparing** and **what you are not comparing** when you use different tests. We think this understanding is more important than what test you use. More important, is that you know what means you want to compare. In this case, we will talk about what means we want to compare, and then just do a t-test.

8.4.7.1 What did the ANOVA tell us

Remember, the ANOVA we conducted is termed the **omnibus** test in the textbook. What means was it comparing? It wasn't comparing specific means. It was asking a kind of blind and very general question: Are any of these means different. Our answer was yes. Our next question is: What means were different from what other means? The ANOVA doesn't know the answer to this question. It just says I don't know...

8.4.7.2 Comparing specific means and the experimental question

Notice there are 4 means to compare. So, there are $(4-1)! = 3! = 3 \times 2 \times 1 = 6$ total different comparisons. The ! stands for factorial. What's more important is recognizing that when you have more than two conditions (where you can only make one comparison, A vs. B), you get increasingly more comparisons. For four conditions, A, B, C, D, you get six comparisons, they are:

AB, AC, AD, BC, BD, and CD where each letter pair refers to A compared to B (AB), A compared to C (AC), and so on.

Do we actually care about all of these comparisons? Perhaps. What was the point of the experiment? Remember, the experiment was asking a questions, that's why they set-up the condition necessary to produce these means. What question were they asking? What did they want to know?

They wanted to find out if various interventions after watching the scary movie, would change how many bad intrusive memories people would experience in the week following the movie. They discuss the idea that a memory can become malleable when it is re-activated. They want to "Re-activate" the bad memories, and then while they were changeable, do something to them to make them less likely to be intrusive later on. The big idea was that doing a bit of "re-activation" AND then playing Tetris (which takes up visual cognitive bandwidth) could cause changes to the re-activated memories, that would decrease the number of intrusive memories later on. With that reminder, let's go through some different comparisons that we can make, and talk about what they might mean.

8.4.7.3 Control vs. Reactivation_only

There was a control group that did not do anything special after watching the traumatic movie. The mean number of intrusive memories for the control group, gives some idea of the amount of intrusive memories we would expect to measure, when you do nothing to change that number.

Comparing to the control group is a very sensible thing to do for each of the other groups. If one of the manipulations worked, it should show a different mean (something changed) than the control group.

So, did the Reaction_only group have less intrusive memories over the control group? First we use the `filter` function from `dplyr` to select only the rows from the data frame that have the Control and Reactivation_only conditions. Then we run a t-test

```
comparison_df <- all_data %>%
  filter(Condition %in% c('Control', 'Reactivation_only') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

## 
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
## t = 0.219, df = 34, p-value = 0.828
## alternative hypothesis: true difference in means between group Control and group Reac
## 95 percent confidence interval:
## -2.299851 2.855406
## sample estimates:
##          mean in group Control mean in group Reactivation_only
##                  5.111111           4.833333
```

The means are both basically 5, not a big difference!. The p-value is large, suggesting that change could easily have produced the tiny differences between the means. In other words, it doesn't look like the "re-activation" phase did anything to suppress the amount of intrusive memories that people would have over one week, compared to control.

Notice, this was a comparison we could make. But, was it an informative one about the goal of the study? Not really.

8.4.7.4 Control vs. Reactivation+Tetris

What we really want to know is if Reactivation+Tetris cause fewer intrusive memories...but compared to what? Well, if it did something, it should have a smaller mean than the Control group. So, let's do that comparison:

Note: we just change the one level name to the level we want Reactivation+Tetris.

```
comparison_df <- all_data %>%
    filter(Condition %in% c('Control', 'Reactivation+Tetris') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

##
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
## t = 2.9893, df = 34, p-value = 0.005167
## alternative hypothesis: true difference in means between group Control and group Reactivation+
## 95 percent confidence interval:
## 1.031592 5.412852
## sample estimates:
## mean in group Control mean in group Reactivation+Tetris
## 5.111111 1.888889
```

There is a bigger difference now, roughly 5.1 intrusive memories for control, and 1.9 for Reactivation+Tetris. The p-value is quite small, indicating this difference was not likely produced by chance. Now, we have some evidence that Reactivation+Tetris caused something to change, that condition produced fewer intrusive memories than control.

8.4.7.5 Control vs. Tetris_only

Now we can really start wondering what caused the difference. Was it just playing Tetris? It wasn't just doing the reactivation, we already found out that didn't do anything. Does just playing Tetris reduce the number of intrusive memories during the week? Let's compare that to control:

```
comparison_df <- all_data %>%
    filter(Condition %in% c('Control', 'Tetris_only') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

##
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
## t = 1.0129, df = 34, p-value = 0.3183
## alternative hypothesis: true difference in means between group Control and group Tetris_only is
## 95 percent confidence interval:
```

```
##   -1.230036  3.674480
## sample estimates:
##   mean in group Control mean in group Tetris_only
##                  5.111111      3.888889
```

There's mean difference of about 1, but the p-value isn't very small. This suggests chance produces a difference of this size fairly often. If we claimed that just playing Tetris caused a difference based on this data, we could easily be making a type I error (claiming the result is real when it is not, a false-positive). Still, the difference was in the right direction wasn't it.

8.4.7.6 Tetris_only vs. Reactivation + Tetris

Finally, we might ask if the Reactivation+Tetris group had fewer unwanted memories than the Tetris_only group. Did putting the two things together (reactivation AND Tetris) really do something special here, beyond just playing Tetris.

```
comparison_df <- all_data %>%
  filter(Condition %in% c('Tetris_only', 'Reactivation+Tetris') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

##
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
## t = -2.5147, df = 34, p-value = 0.01681
## alternative hypothesis: true difference in means between group Reactivation+Tetris and group Tetris_only is not equal to 0
## 95 percent confidence interval:
##   -3.6162855 -0.3837145
## sample estimates:
##   mean in group Reactivation+Tetris           mean in group Tetris_only
##                      1.888889                     3.888889
```

Well, according to the t-test, the p-value is again fairly small. Suggesting that the difference between Reactivation+Tetris ($M=1.89$) and Tetris_only (3.89), was not likely to be produced by chance. So, on the basis of this, there is some evidence that Reactivation+Tetris, really does cause fewer intrusive memories.

8.4.8 Writing it all up.

Because we have spent so much time on individual comparisons, we won't do a full write up of the results. A full write-up would include telling the reader what data was used, what test was conducted, the results of the test, and the

pattern of the means, AND then, the results of specific comparisons of interest. You can read the paper to see how the authors did it.

8.4.9 Food for thought

Think about what we did here. We almost blindly just downloaded the data and ran the same analysis as the authors did. Sure, we looked at the data first, and then did the analysis. But, did we really look? Did you notice anything about what you looked at? What did we not look closely at that might make you skeptical of the conclusions from the research...

Here's a hint. Sample-size. We know that is important. Let's ask the question, was there an equal number of participants in each of the 4 conditions. We can use `dplyr` again to do this. We'll add the `length` function, which counts the number of subjects in each condition:

```
descriptive_df <- all_data %>%
  group_by(Condition) %>%
  summarise(means = mean(Days_One_to_Seven_Number_of_Intrusions),
            SEs = sd(Days_One_to_Seven_Number_of_Intrusions)/sqrt(length(Days_One_to_Seven_Number_of_Intrusions)),
            count = length(Days_One_to_Seven_Number_of_Intrusions))

knitr::kable(descriptive_df)
```

Condition	means	SEs	count
Control	5.111111	0.9963623	18
Reactivation+Tetris	1.888889	0.4113495	18
Tetris_only	3.888889	0.6806806	18
Reactivation_only	4.833333	0.7848650	18

The answer is YES, there were an equal number of subjects. That's good. We should have checked that before. Lesson for next time. For example, if there were only 9 subjects in the Reactivation+Tetris group, we might be suspicious that they got lucky, and accidentally (by chance) assigned people to that group who are unlikely to report having intrusive memories. After all, different people are different, and not everybody is as susceptible to intrusive memories.

Let's do one more thing for fun, and to see everything in action all in one place. Let's consider the role of outliers. Looking at the first graph we can see that most people in all the groups had fewer than 10 intrusive memories (mean we assume) per week. It looks like 5 people had more than that, and they just happened to be in the other groups. Maybe Reactivation+Tetris made those people have way less intrusive memories (which is why no one is above 10), or maybe the researchers got a little bit unlucky, and accidentally didn't get any "outliers" (people with extreme values on the measure) in that group.

Let's re-run the analysis, but remove anybody with a mean higher than 10. This will only remove 5 subjects, so we will still have a lot left. What happens?

Before we find out, let me point out again the beauty of R. All we need to do is copy and paste our previous code. Then, just filter the data once to remove the outliers, then voila, we redo everything all in one go. It's much more complicated and time consuming to do this in many other software programs. You are lucky to be learning R.

```
# get rid out of outliers

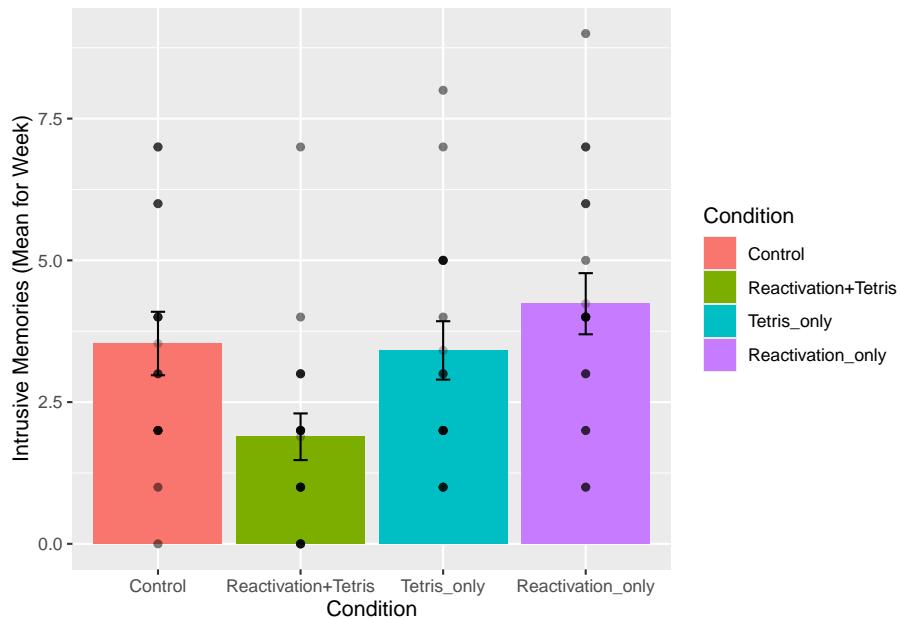
all_data <- all_data %>%
  filter(Days_One_to_Seven_Number_of_Intrusions < 10)

# get means and SEs

descriptive_df <- all_data %>%
  group_by(Condition) %>%
  summarise(means= mean(Days_One_to_Seven_Number_of_Intrusions),
            SEs = sd(Days_One_to_Seven_Number_of_Intrusions)/sqrt(len(Condition)))

# Make the plot

ggplot(descriptive_df, aes(x=Condition, y=means))+
  geom_bar(stat="identity", aes(fill=Condition))+ # add means
  geom_errorbar(aes(ymin=means-SEs, # add error bars
                     ymax=means+SEs), width=.1) +
  geom_point(data=all_data, aes(x=Condition, y=Days_One_to_Seven_Number_of_Intrusions))
  geom_point(alpha=.25)+
  ylab("Intrusive Memories (Mean for Week)")
```



```
# run and report the ANOVA

aov_out<-aov(Days_One_to_Seven_Number_of_Intrusions ~ Condition, all_data)
summary_out<-summary(aov_out)

knitr::kable(xtable(summary_out))



|           | Df | Sum Sq    | Mean Sq   | F value  | Pr(>F)    |
|-----------|----|-----------|-----------|----------|-----------|
| Condition | 3  | 51.49152  | 17.163841 | 4.024458 | 0.0110347 |
| Residuals | 63 | 268.68758 | 4.264882  | NA       | NA        |



# conduct critical comparisons

## control vs reactivation+Tetris

comparison_df <- all_data %>%
  filter(Condition %in% c('Control','Reactivation+Tetris') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

## 
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
```

```

## t = 2.4159, df = 31, p-value = 0.02178
## alternative hypothesis: true difference in means between group Control and group Reactivation+Tetris
## 95 percent confidence interval:
##  0.2562181 3.0326708
## sample estimates:
##               mean in group Control mean in group Reactivation+Tetris
##                         3.533333                           1.888889

## Tetris_only vs reactivation+Tetris

comparison_df <- all_data %>%
  filter(Condition %in% c('Tetris_only', 'Reactivation+Tetris') == TRUE)

t.test(Days_One_to_Seven_Number_of_Intrusions ~ Condition,
       comparison_df,
       var.equal=TRUE)

## 
## Two Sample t-test
##
## data: Days_One_to_Seven_Number_of_Intrusions by Condition
## t = -2.3239, df = 33, p-value = 0.02643
## alternative hypothesis: true difference in means between group Reactivation+Tetris and group Tetris_only
## 95 percent confidence interval:
##  -2.8561054 -0.1896463
## sample estimates:
## mean in group Reactivation+Tetris      mean in group Tetris_only
##                         1.888889                           3.411765

```

The take home is that yes, even after removing outliers, the same basic pattern in the data is observed. Overall, this is a small n study, and ideally the basic findings should be replicated in another lab before we really have full confidence in them. But, I'd say the trends here look promising.

That's ANOVA. Come back next week for another ANOVA tutorial, this time using within-subject data. It's called a repeated measures ANOVA, and it's what's happening next week.

8.4.10 Generalization Exercise

Your task is to conduct the ANOVA using Day_Zero_Number_of_Intrusions as the dependent variable. Report the ANOVA table, a figure to show the means, and a short write-up of the results.

8.4.11 Writing assignment

(2 points - Graded)

Answer the following questions:

1. Explain why the ANOVA is called an omnibus test, and how the omnibus test is different from comparing specific means with t-tests. (1 point)
2. Explain the general similarity between the ratios that are used to calculate the F value and the t-value (1 point)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

8.5 Excel

How to do it in Excel

8.6 SPSS

In this lab, we will use SPSS to:

1. Perform a One-Factor Analysis of Variance (ANOVA) and Graphing the data
2. Planned comparisons: Use t-tests to locate differences
3. Unplanned comparisons: Use a post-hoc test to locate differences

8.6.1 Experiment Background

As described at the beginning of this lab manual section, researchers in the James et al. (2015) study sought to explore whether completing a visuospatial task (e.g., Tetris) after a memory was formed would interfere with the storage of that memory, and thereby reduce the frequency of subsequent intrusions. They hypothesized that only participants who complete a visuo-spatial task after reactivation of the traumatic memories would experience a reduction in intrusive memories. In comparison, simply completing a visuo-spatial task (without reactivation) or reactivation (without a visuo-spatial task), would not reduce the occurrence intrusive memories. In other words, if you play Tetris shortly after you were remembering bad memories, you will experience intrusive memories less often in the future.

The authors conducted an experiment (N = 72, n = 18 per condition) wherein all participants viewed a series of video clips of graphic violence (e.g., a person

getting hit by a van while using his phone as he crosses the road). Over the next 24 hours, participants recorded their intrusive memories. All groups were predicted to have an equal occurrence of intrusive memories during this pre-manipulation period (called Day 0).

Experimental Task: After this 24-hour period, the participants returned to the lab and completed the experimental task. The experimenters randomly assigned participants to ONE of the following conditions:

1. No-task control: These participants completed a 10-minute music filler task.
2. Reactivation + Tetris: These participants were shown a series of images from the trauma film to reactivate the traumatic memories (i.e., reactivation task). After a 10-minute music filler task, participants played the video game Tetris for 12 minutes.
3. Tetris Only: These participants played Tetris for 12 minutes, but did not complete the reactivation task.
4. Reactivation Only: These participants completed the reactivation task, but did not play Tetris.

Dependent variable: Intrusive Memories: All participants were asked to record the number of intrusive memories that they experienced over the next seven days (Days 1 to 7).

8.6.2 Performing a One-Factor Analysis of Variance (ANOVA) & Graphing the data

First, let's open the relevant data file; Here is the link. It's called "James2015_Experiment2.sav" Open this file in SPSS. Your data should look like this:

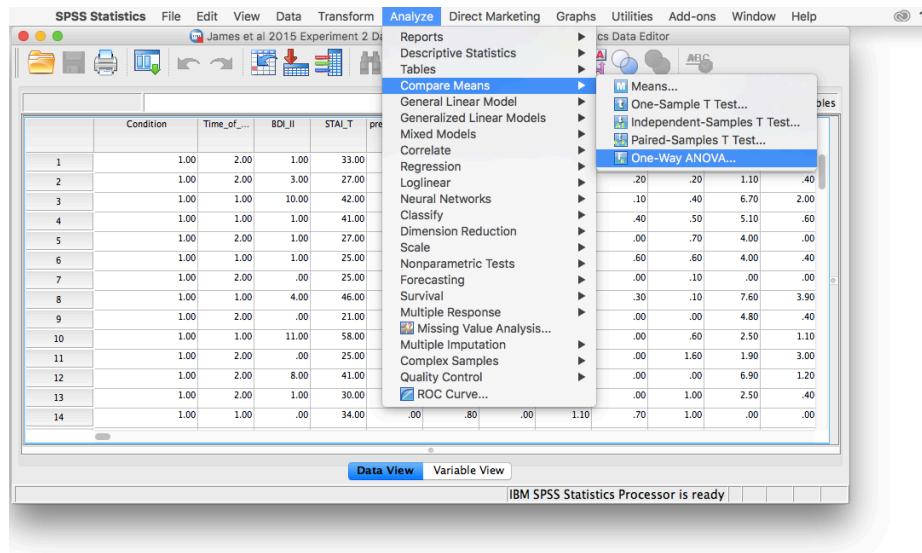
The screenshot shows a data editor window for SPSS. The title bar reads "James et al 2015 Experiment 2 Data Set.sav [DataSet3] - IBM SPSS Statistics Data Editor". The main area displays a data table with 14 rows and 16 columns. The columns are labeled as follows: Condition, Time_of..., BDI_II, STAI_T, pre_film_VAS_Sad, pre_film_VAS_Hopeless, pre_film_VAS_Depressed, pre_film_VAS_Fear, pre_film_VAS_Horror, pre_film_VAS_Anxious, post_film_VAS_Sad, post_film_VAS_Hopeless. The 'Condition' column contains values 1 through 14. The 'Time_of...' column contains values 1.00 and 2.00. The other columns contain various numerical values. The status bar at the bottom indicates "IBM SPSS Statistics Processor is ready".

In this experiment, there was one important independent variable, called **condition**, and it has four levels. The levels are 1, 2, 3, and 4 in the spreadsheet. These will correspond to the four levels in shown below:

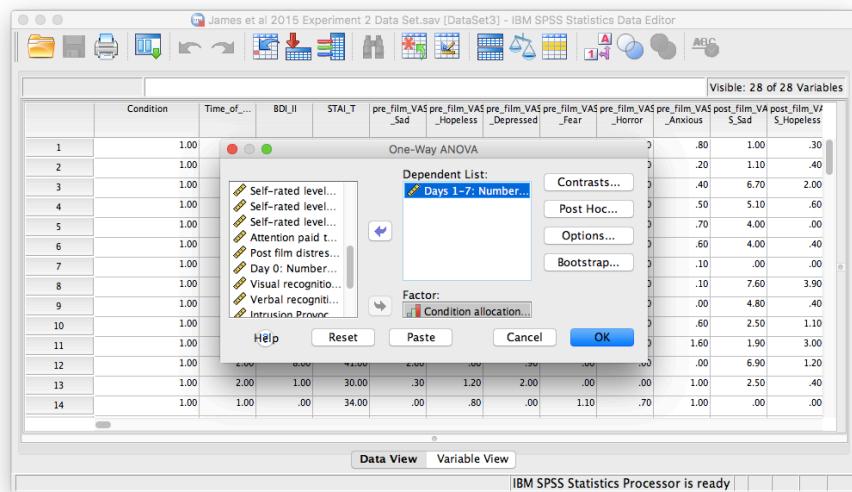
1. No-task control
2. Reactivation Plus Tetris
3. Tetris only
4. Reactivation only

The dependent variable is coded in the column **Days_One_to_Seven_Number_of_Intrusions**. This represents the mean number of intrusions for that week. We are testing the hypothesis that the number of intrusive memories will be different among the groups (1-4). For now, we will not specify which group we think is going to have the highest or lowest number of memories, but rather we will test if there is a difference SOMEWHERE between these 4 groups.

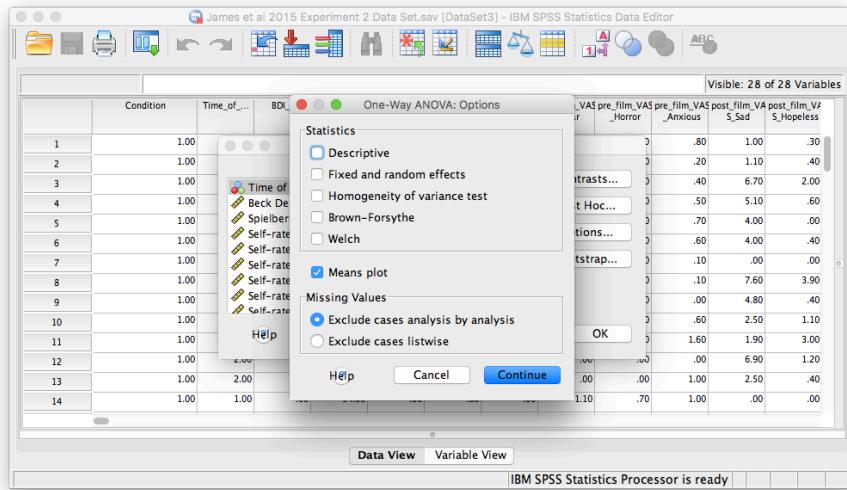
Let's begin. Go to Analyze, then Compare Means, then select One-Way ANOVA...



The next window will ask you to specify the dependent variable (move the variable Days_One_to_Seven_Number_of_Intrusions from the list on the left to the field labeled Dependent Variable) and the “Factor”, which is the same thing as the independent variable or grouping variable (remember back to t-tests!). Move the variable Condition from the list into the field labeled “Factor”. Your window should look like this:



Now, click the Options button, and then choose Means plot.



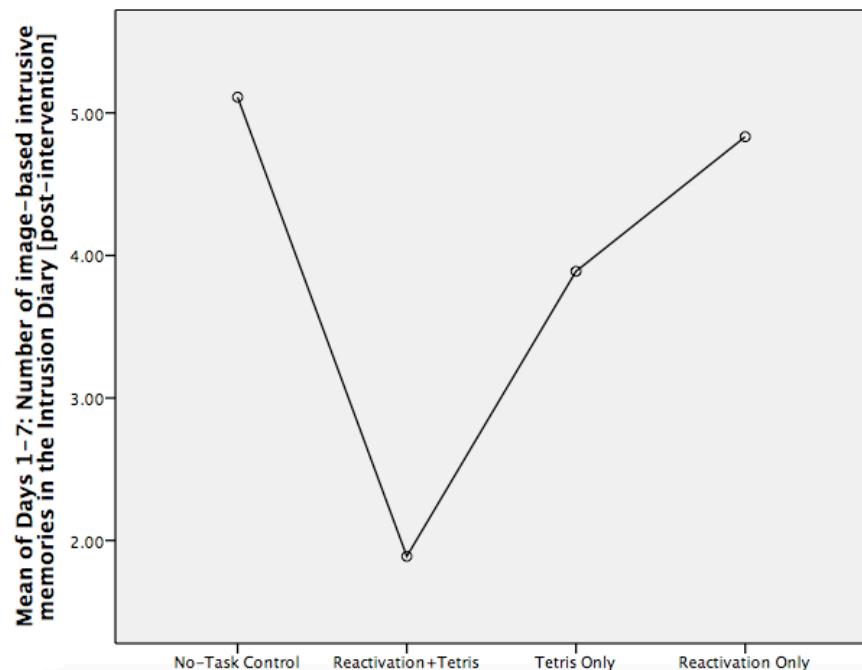
Then, click Continue and OK.

SPSS will produce an output window containing the results of the ANOVA (an ANOVA table), as well as a graph of the group means. It should look as follows:

ANOVA

Days 1-7: Number of image-based intrusive memories in the Intrusion Diary [post-intervention]

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	114.819	3	38.273	3.795	.014
Within Groups	685.833	68	10.086		
Total	800.653	71			

Means Plots

First, let's look at our ANOVA table: We see our F-statistic listed is 3.795, with a p-value of .014. This tests the null hypothesis that there is no difference among the groups. Since p is less than our likely alpha level (.05), we can reject the null hypothesis and conclude that there IS a difference somewhere among these groups. We may say it more formally as follows:

The effect of condition on mean intrusive memories over days 1-7 was significant, $F(3, 68)=3.795, p<.05$.

To examine the data further, let's refer to the graph created by SPSS: we can see that the Reactivation + Tetris condition appears to have the smallest mean. As for the others, they are all greater than Reactivation + Tetris, but differ somewhat from each other as well. It is important to know that at this point, we do not know where the significant differences could be. For example, it could

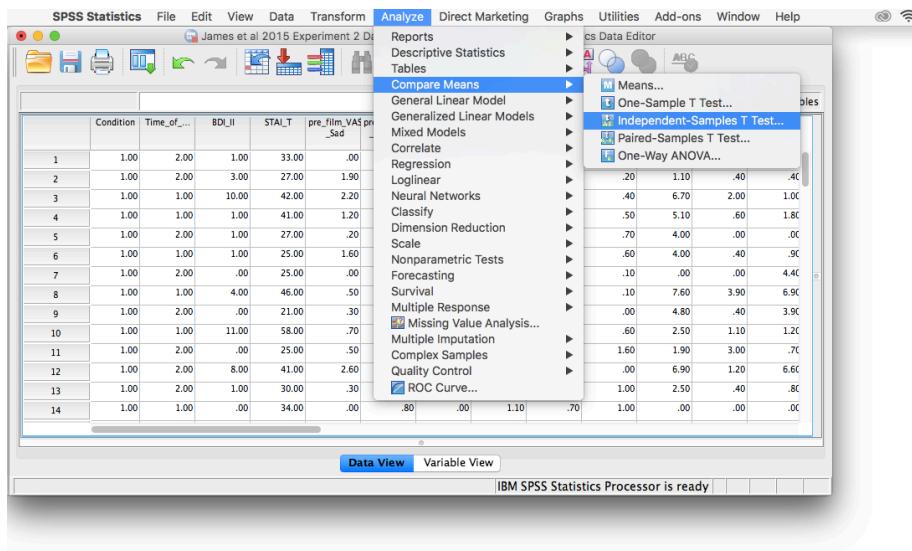
be that only Reactivation + Tetris is different from No-Task Control, or it could be that there are other groups different from one another as well. In order to locate where the differences are, we must conduct additional analyses called either planned or unplanned comparisons.

8.6.3 Planned Comparisons: T-tests

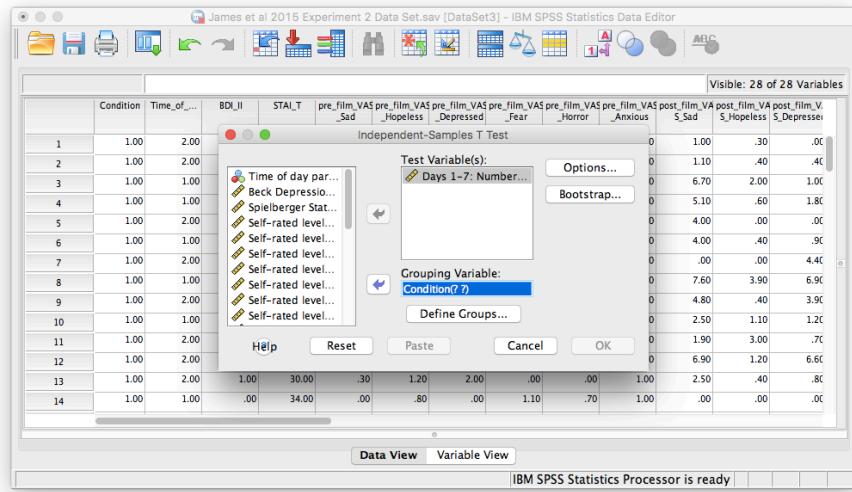
To find out where the differences are, one approach is to conduct as many t-tests as 2-group comparisons (also called pairwise comparisons) you can make out of your groups. That is, if you have 4 groups like we do, we can make the following pairwise comparisons:

1. groups 1 vs. 2
2. groups 1 vs. 3
3. groups 1 vs. 4
4. groups 2 vs. 3
5. groups 2 vs. 4
6. groups 3 vs. 4

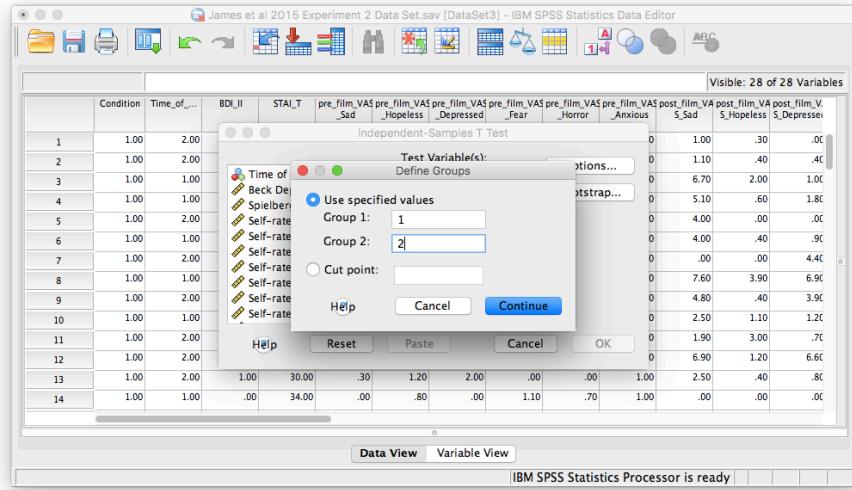
For each of these, we can run a separate independent-samples t-test. Let's start with the first comparison (Groups 1 and 2). Go to Analyze, then Compare Means, then Independent-Samples t-test...



Let's place our dependent variable (`Days_One_to_Seven_Number_of_Intrusions`) in its appropriate field. Then, place `Condition` into the Factor field.



Remember here that we have to click Define Groups to specify which groups we are comparing. Let's put 1 and 2 in the fields from Group 1 and Group 2, as follows:



Click Continue and then OK. SPSS will produce the standard output tables for an independent-samples t-test.

T-Test

Group Statistics						
Condition allocation	N	Mean	Std. Deviation	Std. Error Mean		
Days 1-7: Number of image-based intrusive memories in the Intrusion Diary [post-intervention]	No-Task Control	18	5.1111	4.22721	.99636	
	Reactivation+Tetris	18	1.8889	1.74521	.41135	

Independent Samples Test								
	Levene's Test for Equality of Variances			t-test for Equality of Means				
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference
	8.956	.005	2.989	34	.005	3.22222	1.07794	1.03159 5.41285
Days 1-7: Number of image-based intrusive memories in the Intrusion Diary [post-intervention]	Equal variances assumed		2.989	22.632	.007	3.22222	1.07794	.99033 5.45411
	Equal variances not assumed							

As we can see, our t-test is significant; there is indeed a difference between groups 1 and 2, $t(22.632) = 2.989$, $p < .05$.

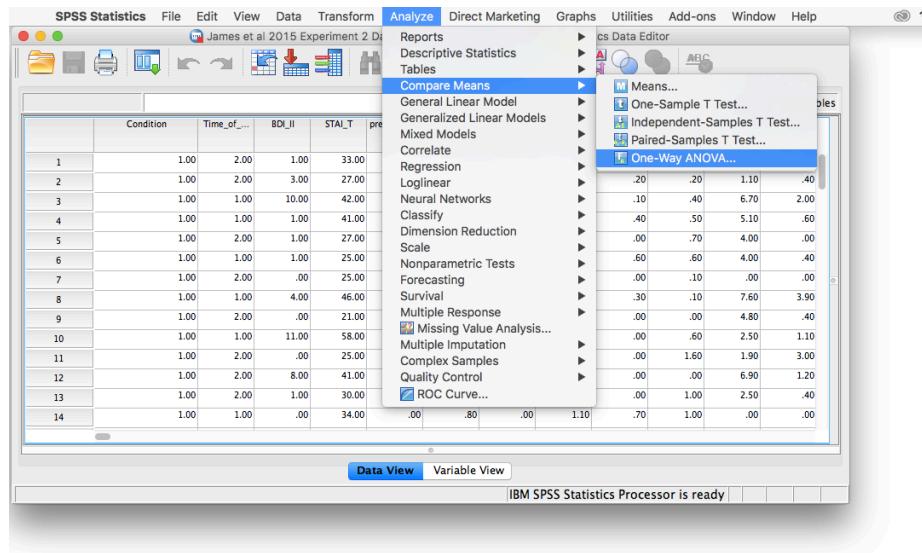
Now, that covers the comparison between groups 1 and 2. We now need to do the same thing for the rest of the comparisons (there are 5 comparisons left to conduct). The procedure for conducting these t-tests is the same as described here, and will only differ in the Define Groups stage of setting up the analysis; in this step, we used the numbers 1 and 2 to represent groups 1 and 2, but for the remainder of the comparisons you will use 1 and 3, 1 and 4, 2 and 3, 2 and 4, and so on.

8.6.4 Unplanned Comparisons: Post-hoc tests

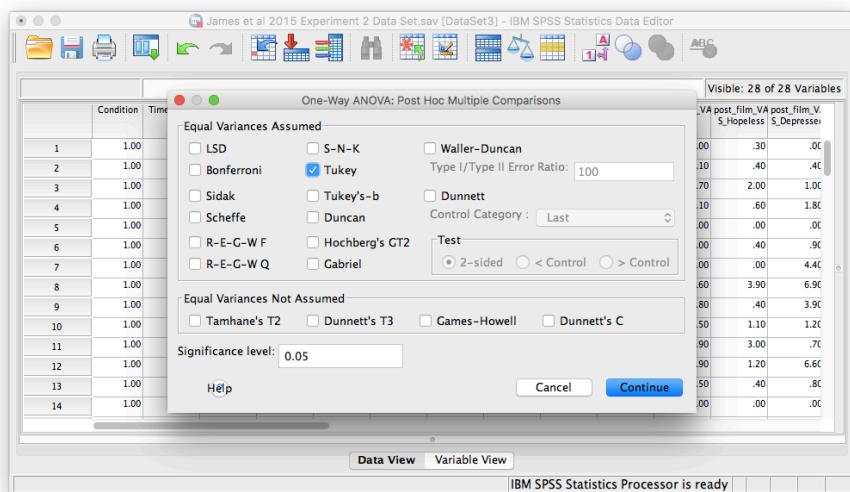
Another way to find the differences between groups in a One-Factor ANOVA is to conduct something called a post-hoc test. This test is referred to as “unplanned” because you only conduct one when you have a significant overall ANOVA. The post-hoc test is neat because it is conveniently summoned during the process of conducting an ANOVA, and presents its results in a compact table.

Let’s use a post-hoc test to look at this data. Although there are many types of post-hoc tests, we’re going to conduct the commonly-used Tukey post-hoc test in this example. Let’s begin.

We are going to go through the same steps as before in order to conduct our ANOVA: Go to Analyze, then Compare Means, then One-Way ANOVA...



Place the dependent variable (Days_One_to_Seven_Number_of_Intrusions) in its appropriate field. Then, place Condition into the “Factor” field. Now, click the button labeled Post-hoc. A window will appear with options for various post-hoc tests. We are going to choose Tukey:



Click Continue, and then OK. SPSS will produce the ANOVA table once again, followed by two tables that refer to the post-hoc test. Let’s look at the first table, named “Multiple Comparisons.”

8.6.4.1 Multiple Comparisons

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Days 1–7: Number of image-based intrusive memories in the Intrusion Diary [post-intervention]
Tukey HSD

(I) Condition allocation	(J) Condition allocation	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
No-Task Control	Reactivation+Tetris	3.22222*	1.05860	.017	.4342	6.0103
	Tetris Only	1.22222	1.05860	.657	-1.5658	4.0103
	Reactivation Only	.27778	1.05860	.994	-2.5103	3.0658
Reactivation+Tetris	No-Task Control	-3.22222*	1.05860	.017	-6.0103	-.4342
	Tetris Only	-2.00000	1.05860	.242	-4.7881	.7881
	Reactivation Only	-2.94444*	1.05860	.034	-5.7325	-.1564
Tetris Only	No-Task Control	-1.22222	1.05860	.657	-4.0103	1.5658
	Reactivation+Tetris	2.00000	1.05860	.242	-.7881	4.7881
	Reactivation Only	-.94444	1.05860	.809	-3.7325	1.8436
Reactivation Only	No-Task Control	-.27778	1.05860	.994	-3.0658	2.5103
	Reactivation+Tetris	2.94444*	1.05860	.034	.1564	5.7325
	Tetris Only	.94444	1.05860	.809	-1.8436	3.7325

*. The mean difference is significant at the 0.05 level.

In this table, every possible group of 2 conditions is being compared. Notice that the table is redundant; it considers the comparison between no-task control and reactivation + tetris to be different from reactivation + tetris vs. no-task control. However, we know these are the same comparisons. So, we only need to look at half of the table, in essence.

If you start with row 1, this row compares no-task control with reactivation + tetris. If we find the Sig. column, we see the corresponding p-value for this row is .017. Because this is less than our likely alpha level (.05), this means that these two conditions ARE significantly different from one another. Where else is there a Sig. that is less than alpha (.05)?

Reactivation + Tetris compared with Reactivation only has a p-value of .034. This is also a significant difference. These are the only significant differences we see here. This means that, among the 4 groups, only two differences were found: Reactivation + tetris is different from no-task control, and reactivation + tetris is different from reactivation only.

8.6.4.2 Homogeneous Subsets

Homogeneous Subsets

Days 1–7: Number of image-based intrusive memories in the Intrusion Diary [post-intervention]

Tukey HSD^a

Condition allocation	N	Subset for alpha = 0.05	
		1	2
Reactivation+Tetris	18	1.8889	
Tetris Only	18	3.8889	3.8889
Reactivation Only	18		4.8333
No-Task Control	18		5.1111
Sig.		.242	.657

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 18.000.

This table actually gives you the same kind of information as the above table, but is more compact and easier to use. In this table you will see the conditions listed on the left, and a number of “subsets”—in this case there are 2 subsets (2 columns labeled 1 and 2, respectively). In these columns, the means of the conditions are listed (in this case, mean number of intrusive memories).

The way to read a homogeneous subsets table is as simple as follows: If the means of two groups ever appear in the same column together, they are not significantly different from one another. If the means of two groups NEVER appear in the same column together, then they ARE significantly different from one another.

Let's start with Reactivation + Tetris and Tetris Only. Look at the table to determine if the means of these conditions ever appear in the same column together. The answer should be yes. The Reactivation + tetris mean (1.89) and Tetris only mean (3.89) appear in column 1 together. Now, let's move on to Reactivation + Tetris as compared to Reactivation only. Do the means of these two groups ever appear in the same column together? No. The Reactivation + Tetris mean (1.89) only appears in column 1, and the reactivation only mean (4.83) only appears in column 2. This means they are different.

If you go through every possible comparison of two groups, you will find that the only differences are between:

1. reactivation + tetris vs. reactivation only
2. reactivation + tetris vs. no-task control

These are the same pairwise differences we noted using the multiple comparisons table.

8.6.5 Practice Problems

Using this lab's data set, think about the following question: What if participants' intrusive memories were not actually affected by the manipulation (tetris and reactivation), but rather the group with the most intrusive memories just happened to react more strongly to the disturbing films? What if there were differences in "level of disturbance" (hint hint: post-film distress rating) among the 4 groups?

1. Test this hypothesis using an alpha level of .05. Report your result in proper statistical reporting format. What do you conclude?
2. Do the same type of analysis to determine whether the amount of attention paid to the film differed between the 4 groups. Report your result in proper statistical reporting format. What do you conclude?

8.7 JAMOVI

How to do it in JAMOVI

Chapter 9

Lab 9 Repeated Measures ANOVA

However, perhaps the main point is that you are under no obligation to analyse variance into its parts if it does not come apart easily, and its unwillingness to do so naturally indicates that one's line of approach is not very fruitful. —R. A. Fisher

9.1 Betcha can't type JHDBZKCO very fast on your first try

This lab activity uses the data from Behmer & Crump (2017) to teach one-factor repeated measures ANOVA with-up follow comparisons

9.1.1 STUDY DESCRIPTION

Behmer & Crump (2017) used the everyday task of typing on a computer keyboard to ask questions about how people learn to put sequences of actions together. Whenever you type a series of letters on the keyboard, you are putting a sequence of actions together, so typing is task that could be used to measure skilled sequencing. Typing also happens to be a convenient task for measuring sequencing. For example, every time a person types a letter, the timing of the button press and the letter pressed can be measured and stored for later analysis.

Behmer & Crump were interested in asking a few different questions, however, we will simplify everything and talk about replication. First we describe an interesting finding from previous research. Behmer & Crump repeated an experiment that should also produce this same finding. If they succeed in doing

this, it means the finding can be replicated, and that it happens in more than one lab.

Finding from previous research: Prior research showed that typists do something funny. Skilled typists can type normal words very fast. This suggests they know how to locate all of the letters on the keyboard, and can press each letter very quickly to type words. That part isn't particularly funny. However, if you take really skilled typists and make them type random letters like this: kwitb dhhgjtryq xkldpt mazhyffdt, guess what happens? They slow down a lot. It's kind of weird that a typist would slow down, after all they can type letters really fast when they appear in words, but not when they appear in random orders...what gives? Last, it turns out that typists are kind of in the middle in terms of speed, if you ask them to type non-words that have similar properties to words, such as: quenp hamlke phwempy.

To summarize, prior research showed that typing speed changes as a function of the structure of the text, roughly in this order from fastest to slowest.

(FASTEST) Normal Words < Word-like Non-words < Random strings (SLOWEST)

Replication question: Behmer & Crump also measured typists while they typed words, non-words that were English-like, and random strings. They had some additional things they were interested in, but for us, we are interested in whether they would show the same effect. Would they replicate the pattern: Normal words (Fastest) < Word-like Non-words (medium) <- Random strings (Slowest)?

9.1.2 Study Methods

The authors conducted a repeated measures experiment. A total of 38 subjects were used for the analysis.

Independent Variable: The IV Stimulus or typing material had three levels: Normal, Bigrams, and Random. Normal refers to normal 5 letter English words (like truck, or plant). Bigrams refers to non-words that have properties similar to words (e.g., phemt quilp). Random refers to 5 letter strings whose letters were totally random (qmklt gdrzn lprni).

Dependent Variables: There were three dependent variables, that all measured different aspects of typing performance. Reaction times (RTs) were defined as the temporal interval between seeing a stimulus (to type), and then starting to type it (first key press). Inter-keystroke intervals (IKSIs) are the times between each key-press. Last, accuracy was also measured (correct or incorrect key-presses)

The task: Participants (who happened to also be students from Brooklyn College) sat in front a computer. They were presented with one stimulus (word, bigrams, or random) at a time. As soon as they saw the string of letters, they

typed it as quickly and accurately as they could, then they moved on to the next trial.

Reminder, this is a repeated measures design because each participant typed letter strings from the word, bigrams, and random conditions.

9.2 Lab Skills Learned

- Conducting a one-factor repeated measures ANOVA
- Conducting follow-up comparisons

9.3 Important Stuff

- citation: Behmer, Lawrence P., Crump, M. J. C. (2017). Spatial Knowledge during Skilled Action Sequencing: Hierarchical versus Non-Hierarchical Representations. *Attention, Perception & Psychophysics*, 79, 2435-2448.
- Link to .pdf of article
- Data in .csv format

9.4 R

9.4.1 Load the data

Remember that any line with a # makes a comment and the code does not run. Below is how to load the .csv data from the online repository, or from a local file (you need to change the file path to where the local file is, if you downloaded it). The data contains all of the measures and conditions from Experiment 1 in the paper.

```
library(data.table)
#all_data <- fread("https://github.com/CrumpLab/statisticsLab/raw/master/data/exp1_BehmerCrumpAPP.csv")
all_data <- fread("data/exp1_BehmerCrumpAPP.csv")
```

9.4.2 Inspect the dataframe

This will give you a big picture of the data frame. Click the button to view it in your browser, then take a look to see what is in it.

```
library(summarytools)
view(dfSummary(all_data[,c(1:7,10:20)]))
```

Note, there is some weird stuff in code above. Normally, we would just write `view(dfSummary(all_data))`, why we add this: `all_data[,c(1:7,10:20)]?` It turns out the `dfSummary` function didn't like some of the data. In particular it didn't like the data in columns 8 and 9 (notice those numbers are missing, the

range inside `c` is 1 to 7 and 10 to 20). It doesn't mean the data isn't there, just that it didn't want to display it in the viewer.

9.4.3 Get the data you need

This data file contains all of the data from Experiment 1 in the paper. So, we don't need to get rid of any rows.

There are numerous columns, some of them we don't need for the analysis. But, we'll just ignore these later when we use `dplyr` to group by the columns we want.

The structure of this data a file is in long form. Every row described a measurement for a single key-press. For example, the first 5 rows, have data for the timing of the first 5 key-presses, that the first subject made to type the first string of letters they saw. In total there were 85,410 key-presses made. That's quite a lot.

9.4.3.1 The independent variable

The important independent variable is in the column `Stimulus`.

- Normal (5 letter English words)
- Bigrams (5 letter strings that kind of looked like words)
- Random (5 letter strings that were random)

It is also important to know that the `Order` column codes the position for each letter, from 1 to 5.

Note: there was another independent variable in the study as well. We talk about this later. The second IV is coded in the `Block` column.

- Baseline (normal typing, keyboard is visible while typing)
- Manipulation (occluded typing, keyboard is covered while typing)

9.4.3.2 The dependent variables

1. `TimeFromOnset` : This column records the temporal interval in milliseconds between the onset of the word and each key-press. When order is 1 (first keystroke), the number here is the reaction time to start typing.
2. `PureRTs` : This column contains keystroke intervals. The first interval is between the onset of the word and the first key-press (order 1), the second interval is between the first and second key-press (order 2), and so on. `PureRTs` for orders 2 to 5, represent the inter-keystroke intervals reported in paper.
3. `AllCorrect` : 0 means incorrect (wrong letter was typed), 1 means correct (correct letter was typed)

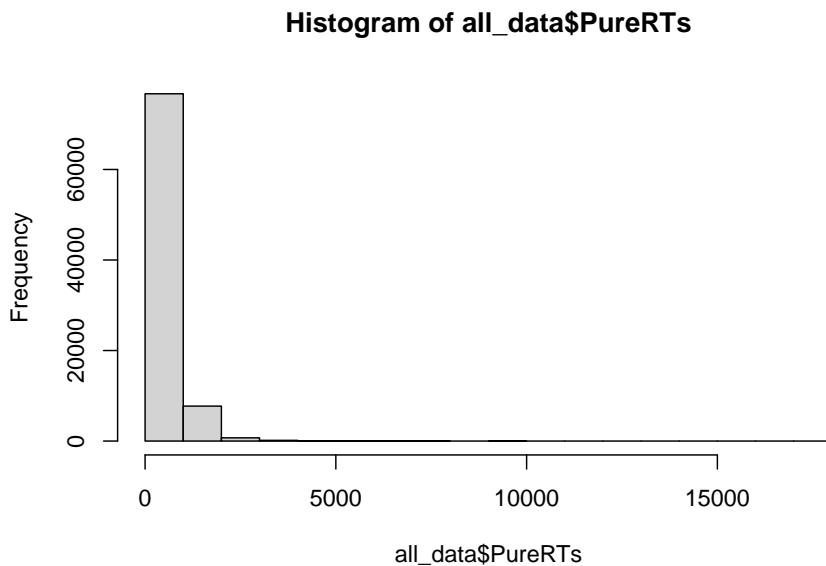
9.4.4 Look at the data

Remember before we do any analysis, we always want to “look” at the data. This first pass let’s us know if the data “look right”. For example, the data file could be messed up and maybe there aren’t any numbers there, or maybe the numbers are just too weird.

For example, this study involves reaction times: the time between seeing something and responding to it. If you had done a study like this before, you would know that it usually doesn’t take people that long to start responding. Most reaction times will be under a second (or 1000 milliseconds). But, sometimes people are little slow, and sometimes they do funny things like check their phone in the middle of an experiment.

Before I analyze reaction time data, I often make a histogram of all of the RT data, like this:

```
hist(all_data$PureRTs)
```



We can see that almost all of the reaction times are well below 5000 milliseconds (5 seconds), which is good. Most of the time people were paying attention and not “checking their phone”. Notice, the range of the histogram goes out to 15,000 milliseconds. You can’t see any bars out there (too small to notice), but there must be at least a few trials where somebody took 15 seconds to start responding. These are called outliers. We will remove them before we conduct our analysis.

9.4.5 Look at the means

As part of looking at the data, we might as well make a figure that shows the mean reaction times in each condition, and some error bars to look at the spread in each condition. The following code takes three important steps:

1. Get the means for each subject in each condition. These are put into the data frame called `subject_means`.
2. Get the means for each condition, by averaging over the means for each subject. These are put into the data frame called `plot_means`.
3. Make a graph with the `plot_means` data frame using ggplot2.

```
library(dplyr)
library(ggplot2)

all_data$Block<-as.factor(all_data$Block)
levels(all_data$Block) <- c("Visible keyboard", "Covered Keyboard")

## get subject mean RTs

subject_means <- all_data %>%
  filter(Order==1, Correct==1, PureRTs<5000) %>%
  group_by(Subject, Block, Stimulus) %>%
  summarise(mean_rt = mean(PureRTs))

subject_means$Subject<-as.factor(subject_means$Subject)
subject_means$Block<-as.factor(subject_means$Block)
subject_means$Stimulus<-as.factor(subject_means$Stimulus)

## get condition mean RTs

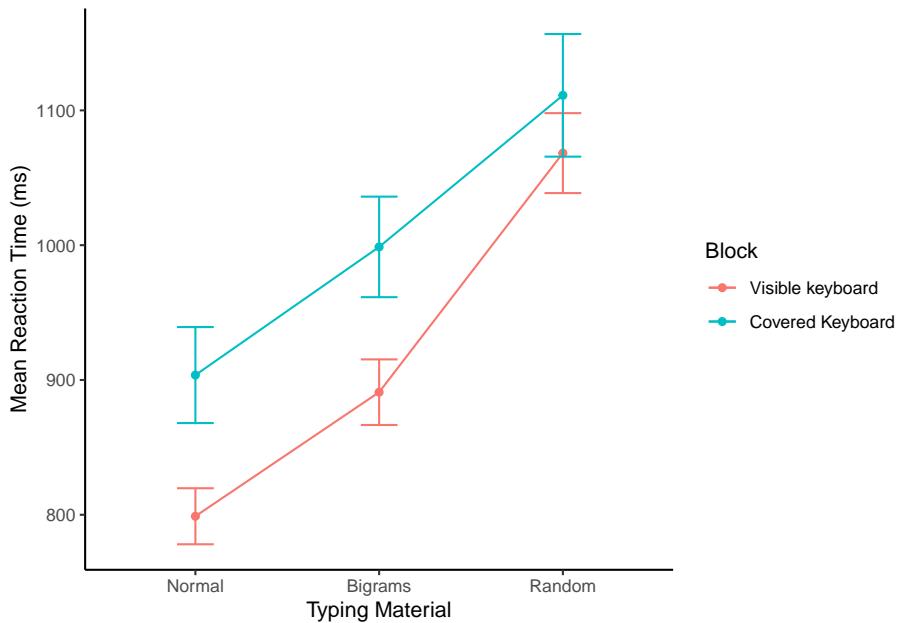
plot_means <- subject_means %>%
  group_by(Block, Stimulus) %>%
  summarise(means = mean(mean_rt),
            SEs = sd(mean_rt)/sqrt(length(mean_rt)))

## plot the condition means

# re-order stimulus factor for plotting
plot_means$Stimulus <- factor(plot_means$Stimulus, levels = c("Normal", "Bigrams", "Ra
```

```
ggplot(plot_means, aes(x=Stimulus, y=means, group=Block, color=Block))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=means-SEs, ymax=means+SEs), width=.2)+
  theme_classic()+
  ylab("Mean Reaction Time (ms)")+
```

```
xlab("Typing Material")
```



Alright, we made things a little bit more complicated than they need to be. Our primary question is whether reaction times followed this pattern: Normal < Bigrams < Random. We can see the means do follow this pattern. However, shouldn't we only be looking at three means, why are their six means, and two lines?

The above code included the second independent variable `Block`. As a result, you are seeing the means for Typing material when subjects could see the keyboard, and when they couldn't see the keyboard. We will come back to this later. For now, let's ignore the Block condition, and find the means for the Typing Material IV by averaging over the Block conditions. We run the same code as above, by take out `Block`, in the `group_by` function. We also take `Block` out the `ggplot` function.

VERY IMPORTANT: We did something in the above code that we didn't point out. We filtered the data before we found the means. For most of the data sets in other labs, we given you data that is more or less ready to analyse. More often than not data needs to be pre-processed, or filtered before you analyze it. We can use the `filter` function in `dplyr` to do our filtering. `filter` filters the rows for us, so we will only include the rows that we want.

1. We want to analyze the time between the onset of the stimulus and the first keystroke. The reaction times for this value are in the `PureRTs` column, but this column contains other RTs that we do not want to analyse. For

example, the `Order` column codes for the letter position in the string. We only want to analyze the rows that contain a 1, for the first position. So, that is why we add `Order==1` to the filter function below.

2. We want to analyze only the reaction times that are correct. That is, when the subject typed the first letter correctly, and did not make a typo. Accuracy is coded in the `Correct` column, with 1 = correct, and 0 = incorrect. We add `Correct==1` to the filtering function.

Note the use of `==`, that is two equal signs in a row. In R, two equal signs in a row has a special meaning. It means conduct a `logic` test to determine if one thing is the same as another.

3. We want to analyze only reaction times that are “sensible” to analyze. What does sensible mean? We don’t want to analyze data that is clearly garbage data. For example, if someone fell asleep at the computer and didn’t respond for 15 seconds, that kind of data is not what we want to analyze. If we were to filter the data, and exclude these kinds of `outliers`, we would be conducting an outlier elimination procedure. Behmer & Crump (2017) did this, and it is commonly done in many different kinds of studies. We skip an extended discussion of outlier elimination for this lab. But, we do introduce the idea of doing it. We want to keep as much of the data as possible. So, what we do is keep all of the RTs that are less than 5000 ms (that’s 5 seconds). To do this, we add `PureRTs<5000` to the filter function.

```
## get subject mean RTs

subject_means <- all_data %>%
  filter(Order==1, Correct==1, PureRTs<5000) %>%
  group_by(Subject, Stimulus) %>%
  summarise(mean_rt = mean(PureRTs))

subject_means$Subject<-as.factor(subject_means$Subject)
subject_means$Stimulus<-as.factor(subject_means$Stimulus)

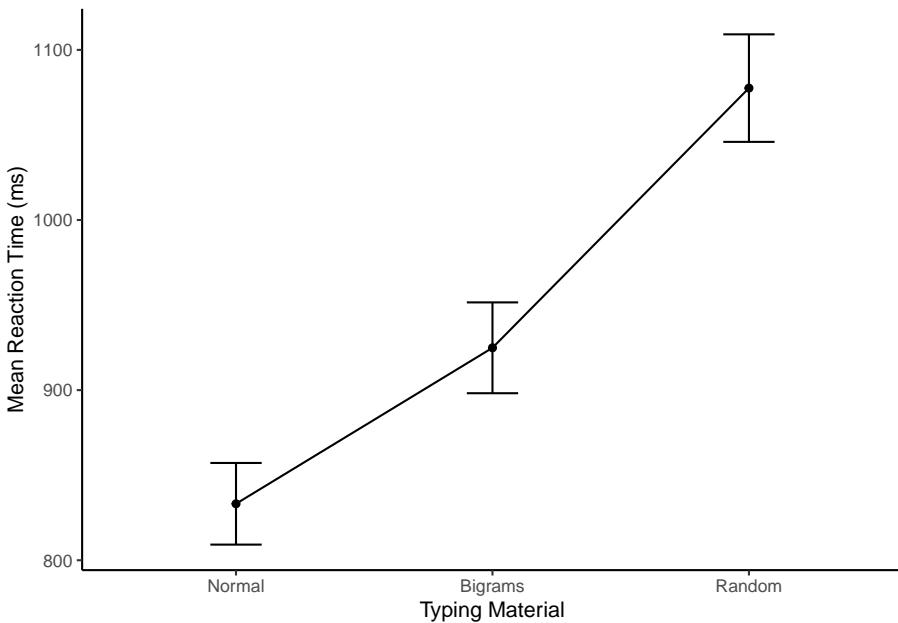
## get condition mean RTs

plot_means <- subject_means %>%
  group_by(Stimulus) %>%
  summarise(means = mean(mean_rt),
            SEs = sd(mean_rt)/sqrt(length(mean_rt)))

## plot the condition means

# re-order stimulus factor for plotting
plot_means$Stimulus <- factor(plot_means$Stimulus, levels = c("Normal", "Bigrams", "Ra
```

```
ggplot(plot_means, aes(x=Stimulus, y=means, group=1))+  
  geom_point() +  
  geom_line(stat="identity") +  
  geom_errorbar(aes(ymin=means-SEs, ymax=means+SEs), width=.2) +  
  theme_classic() +  
  ylab("Mean Reaction Time (ms)") +  
  xlab("Typing Material")
```



9.4.6 Conduct the repeated Measures ANOVA

We use the same `aov` function as we used last time. The only difference is that we add in a new part to the formula. Remember the formula for a one-factor between subjects ANOVA looked like this:

`aov(DV ~ IV , dataframe)`, where DV is the name of the column with your independent variable, IV is the name of the column with your independent variable, and `dataframe` is the name of your data frame containing the means in each condition.

The formula for a repeated-measures ANOVA looks like this:

`aov(DV ~ IV + Error(Subject/IV) , dataframe)`. We have added `+ Error(Subject/IV)`. This tells R to use the appropriate error term for the repeated measures ANOVA. In the formula, `Subject` refers to the name of the column coding your subjects (make sure this is a factor in R), and `IV` is the name of the column for your independent variable.

The formula for our data would be: `aov(mean_rt ~ Stimulus + Error(Subject/Stimulus), subject_means)`.

Here is the code below. Just as reminder, the raw data codes every single key press on each row. We don't want to submit this as the data frame to the `aov` function. Instead, we need to calculate the data frame for the subject means in each condition. We did that above as a step toward making the graphs. We do it again here to remind you that you need to do this.

```
# get subject means

subject_means <- all_data %>%
  filter(Order==1, Correct==1, PureRTs<5000) %>%
  group_by(Subject, Stimulus) %>%
  summarise(mean_rt = mean(PureRTs))

# Make sure IV and Subject are coded as factors
subject_means$Subject <- as.factor(subject_means$Subject)
subject_means$Stimulus <- as.factor(subject_means$Stimulus)

# Conduct the anova

aov_out <- aov( mean_rt ~ Stimulus + Error(Subject/Stimulus), subject_means)
summary_out <- summary(aov_out)

library(xtable)
knitr::kable(xtable(summary_out))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Residuals	37	3030779.9	81912.970	NA	NA
Stimulus	2	1157965.2	578982.606	230.5806	0
Residuals1	74	185812.3	2510.977	NA	NA

Great, we have conducted the ANOVA. We could write up the results of the ANOVA like this:

For each subject we computed mean reactions for correct keystrokes in each condition of the Stimulus factor. These means were submitted to a one-factor repeated-measures ANOVA, with Stimulus (Normal, Bigrams, and Random) as the sole factor. The effect of Stimulus was significant, $F(2, 74) = 230.58$, $MSE = 2510.98$, $p < 0.001$.

Note, the p-value shows up as a zero, that's because it is so small that R doesn't want to print the actual number 0.000000000000000...1.

What does this tell us?

1. The F value we obtained (230.58) almost never occurs by chance. More

specifically, the sampling distribution of F from the distribution of no differences virtually never produces a huge F like 230.58

2. It is super-duper unlikely that chance (sampling error) could have produced the difference we observed.
3. We reject the idea that chance caused the differences, and are very confident that the manipulation (changing the kinds of letters that people have to type), has a causal influence on reaction time in typing.

9.4.6.1 Report the means too

Remember, the important goal when conducting analyses, and then writing about them, is to tell people what you did and what you found. This involves more than one step. For this example, we might do three basic things. 1) make a figure to show the means, 2) report the ANOVA so people know if there is support for the inference that the differences between the means are not caused by chance, and 3) report descriptives for the means, so people know what the numbers are (the figure doesn't show the exact values).

We've already made the figure and done the ANOVA, let's report the condition means. To do this, we need to find the means for each condition, collapsing over the means for each subject in each condition. Note that, we already did this to make the figure. Here's the code again:

```
## get subject mean RTs

subject_means <- all_data %>%
  filter(Order==1, Correct==1, PureRTs<5000) %>%
  group_by(Subject, Stimulus) %>%
  summarise(mean_rt = mean(PureRTs))

subject_means$Subject<-as.factor(subject_means$Subject)
subject_means$Stimulus<-as.factor(subject_means$Stimulus)

## get condition mean RTs

plot_means <- subject_means %>%
  group_by(Stimulus) %>%
  summarise(means = mean(mean_rt),
            SEs = sd(mean_rt)/sqrt(length(mean_rt)))

knitr::kable(plot_means)
```

Stimulus	means	SEs
Bigrams	924.8764	26.69375
Normal	833.1872	24.00055
Random	1077.5361	31.60979

Now, our full write-up of the results would look like this.

For each subject we computed mean reactions for correct keystrokes in each condition of the Stimulus factor. These means were submitted to a one-factor repeated-measures ANOVA, with Stimulus (Normal, Bigrams, and Random) as the sole factor. The effect of Stimulus was significant, $F(2, 74) = 230.58$, $MSE = 2510.98$, $p < 0.001$. The mean reaction time was fastest in the Normal condition ($M = 833$ ms, $SE = 24$ ms), followed by the Bigram condition, ($M = 924$ ms, $SE = 27$ ms) and slowest in the Random Condition ($M = 1078$ ms, $SE = 32$ ms).

9.4.7 Follow-up comparisons

The ANOVA tells us that the differences between the means are unlikely to be due to chance. But, remember, this is an omnibus test. It does not tell us if specific pairs of means are different from one another. To determine whether the difference between two specific means is not likely due to chance, we need to conduct follow-up tests.

Because this is a repeated-measures design, we can use the paired-samples t-test for follow-up tests. Let's do two follow-up tests to confirm that the RTs for Normal words were indeed faster than the RTs for the Bigram condition (word-like non-words); and then, let's confirm that the RTs for the Bigram condition were indeed faster than the RTs for the Random condition.

9.4.7.1 Normal vs Bigrams

We use the `subject_means` data frame. But, we want to rid of all the rows containing the means from the Random condition. We use `filter` to do that, then we conduct the paired-samples t-test.

```
comparison_df <- subject_means %>%
  filter(Stimulus != "Random")

t.test(mean_rt ~ Stimulus,
      paired=TRUE,
      var.equal=TRUE,
      data = comparison_df)

## 
##  Paired t-test
## 
##  data:  mean_rt by Stimulus
##  t = 12.14, df = 37, p-value = 1.807e-14
##  alternative hypothesis: true mean difference is not equal to 0
##  95 percent confidence interval:
##    76.38601 106.99253
```

```
## sample estimates:
## mean difference
##      91.68927
```

9.4.7.2 Bigrams vs Random

We use the `subject_means` data frame. But, we want to rid of all the rows containing the means from the Normal condition. We use filter to do that, then we conduct the paired-samples t-test.

```
comparison_df <- subject_means %>%
  filter(Stimulus != "Normal")

t.test(mean_rt~Stimulus,
  paired=TRUE,
  var.equal=TRUE,
  data = comparison_df)

##
## Paired t-test
##
## data: mean_rt by Stimulus
## t = -14.212, df = 37, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
## -174.4245 -130.8949
## sample estimates:
## mean difference
##      -152.6597
```

9.4.8 Reporting everything

Now we can look at some write-ups that report everything we did, and everything we want to know. I'll show you two ways to do it.

9.4.8.1 First way

In the first way, we embed the results of the t-test into the description of the mean reaction times.

For each subject we computed mean reactions for correct keystrokes in each condition of the Stimulus factor. These means were submitted to a one-factor repeated-measures ANOVA, with Stimulus (Normal, Bigrams, and Random) as the sole factor. The effect of Stimulus was significant, $F(2, 74) = 230.58$, $MSE = 2510.98$, $p < 0.001$. The mean reaction time was significantly faster in the Normal condition ($M = 833$ ms, $SE = 24$ ms), compared to the Bigram condition, ($M = 924$ ms, $SE = 27$ ms), $t(37) = 12.14$, $p < 0.001$. Additionally, mean reactions

in the Bigram condition were significantly faster than the Random Condition ($M = 1078$ ms, $SE = 32$ ms), $t(37) = 14.21$, $p < 0.001$.

9.4.8.2 Second way

In the second way, we first report the means as we did the very first time, and then after that we report the t-test results to highlight the size of the differences between each comparison.

For each subject we computed mean reactions for correct keystrokes in each condition of the Stimulus factor. These means were submitted to a one-factor repeated-measures ANOVA, with Stimulus (Normal, Bigrams, and Random) as the sole factor. The effect of Stimulus was significant, $F(2, 74) = 230.58$, $MSE = 2510.98$, $p < 0.001$. The mean reaction time was fastest in the Normal condition ($M = 833$ ms, $SE = 24$ ms), followed by the Bigram condition, ($M = 924$ ms, $SE = 27$ ms) and slowest in the Random Condition ($M = 1078$ ms, $SE = 32$ ms). Mean reaction times were significantly faster ($M = 91$ ms) in the Normal than Bigrams condition, $t(37) = 12.14$, $p < 0.001$. And, mean reaction times were significantly faster ($M = 152$ ms) in the Bigrams than Random condition, $t(37) = 14.21$, $p < 0.01$.

There are other ways to write-up statistical results. These are just some example recipes. The important thing is to:

1. Say what the numbers were that you are analyzing
2. Say what the statistical test was
3. Say the results of the statistical test
4. Say what the patterns of means were
5. Say what the follow-up tests were when you test differences between specific means.
6. Add a table or figure so it is easier to “see” the results.

9.4.9 Generalization Exercise

Your task is to conduct another repeated-measures ANOVA. Rather than using the reaction time for the first-keystroke as the dependent measure, you will use the reaction times between all of the keystrokes in each word, these are called interkeystroke intervals. The `Order` variable is used to code keystroke position (1 to 5). You will want to analyze only the `PureRTs` that have an `Order` greater than 1. For example, you could use the following code to get the `subject_means` for the mean interkeystroke intervals.

```
subject_means <- all_data %>%
  filter(Order > 1, Correct==1, PureRTs<5000) %>%
  group_by(Subject, Stimulus) %>%
  summarise(mean_rt = mean(PureRTs))
```

- A. Make a figure for the new DV B. Report the ANOVA table for the new

repeated measures ANOVA C. Discuss whether the general pattern is the same as before.

9.4.10 Writing assignment

(2 points - Graded)

1. Explain the concept of SS_{Total} (.5 points)
2. Explain the concept of partitioning SS_{Total} into smaller pieces. What is the goal of the splitting? (.5 points)
3. Explain the major difference between a between-subjects ANOVA and repeated-measures ANOVA in terms of what is being partitioned. (1 point)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

9.5 Excel

How to do it in Excel

9.6 SPSS

In this lab, we will use SPSS to:

1. Produce a frequency histogram and remove outliers
2. Conduct and graph One-Factor Repeated Measures ANOVA
3. Conduct planned comparisons using a paired-samples t-test

9.6.1 Experiment Background

In this experiment, Behmer and Crump (2017) recruited 38 subjects to type words presented on a screen as quickly as they could. The independent variable was typing material and it had 3 levels: Normal (5 letter English words), Bigrams (5 letter non-words that have properties like real words), and Random (random 5 letter strings). The authors wanted to know whether reaction time (RT) was different according to the type of word being typed.

Dependent Variables: There were three dependent variables, that all measured different aspects of typing performance. Reaction times (RTs) were defined as

the temporal interval between seeing a stimulus (to type), and then starting to type it (first key press). Inter-keystroke intervals (IKSIs) are the times between each key-press. Last, accuracy was also measured (correct or incorrect key-presses). For this analysis we will use Pure RTs as our one DV.

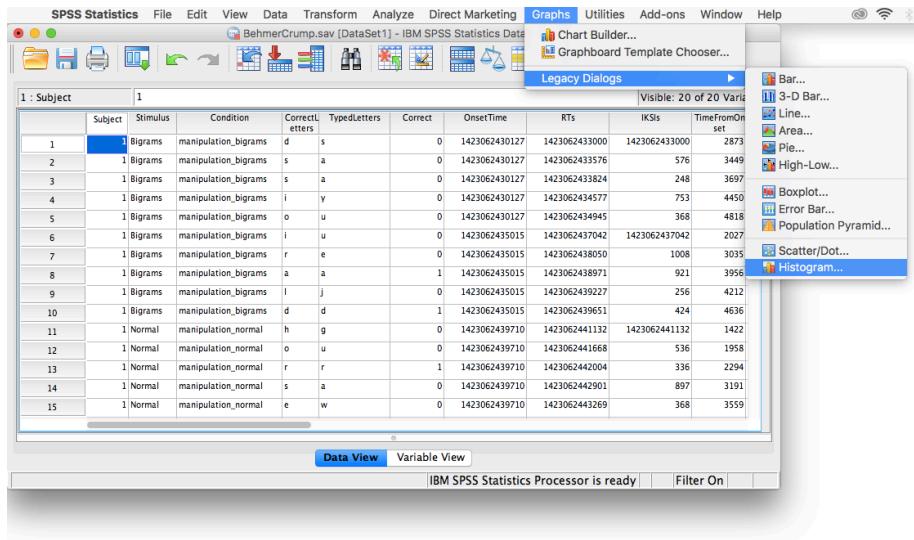
Nota bene: This is a repeated measures design because each participant typed letter strings from the word, bigrams, and random conditions.

9.6.2 Produce a frequency histogram and remove outliers

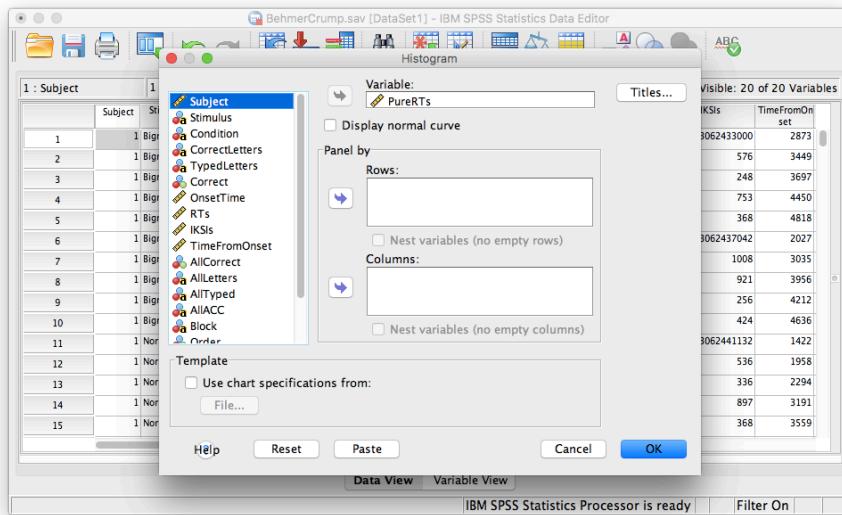
Here is a link to the data file. It is named BehmerCrump.sav. Your data should look like this:

	V1	Subject	Condition	CorrectLetters	TypedLetters	Correct	OnsetTime	RTs	IKSIs	TimeFromOn set
1	1	1	manipulation_bigrams	d	s	0	1423062430127	1423062433000	1423062433000	2873
2	2	1	manipulation_bigrams	s	a	0	1423062430127	1423062433576	576	3449
3	3	1	manipulation_bigrams	s	a	0	1423062430127	1423062433824	248	3697
4	4	1	manipulation_bigrams	i	y	0	1423062430127	1423062434577	753	4450
5	5	1	manipulation_bigrams	o	u	0	1423062430127	1423062434945	368	4818
6	6	1	manipulation_bigrams	i	u	0	1423062435015	1423062437042	1423062437042	2027
7	7	1	manipulation_bigrams	r	e	0	1423062435015	1423062438050	1008	3035
8	8	1	manipulation_bigrams	a	a	1	1423062435015	1423062438971	921	3956
9	9	1	manipulation_bigrams	l	j	0	1423062435015	1423062439227	256	4212
10	10	1	manipulation_bigrams	d	d	1	1423062435015	1423062439851	424	4636
11	11	1	manipulation_normal	h	g	0	1423062439710	1423062441132	1423062441132	1422
12	12	1	manipulation_normal	o	u	0	1423062439710	1423062441668	536	1958
13	13	1	manipulation_normal	r	r	1	1423062439710	1423062442004	336	2294
14	14	1	manipulation_normal	s	a	0	1423062439710	1423062442901	897	3191
15	15	1	manipulation_normal	e	w	0	1423062439710	1423062443269	368	3559

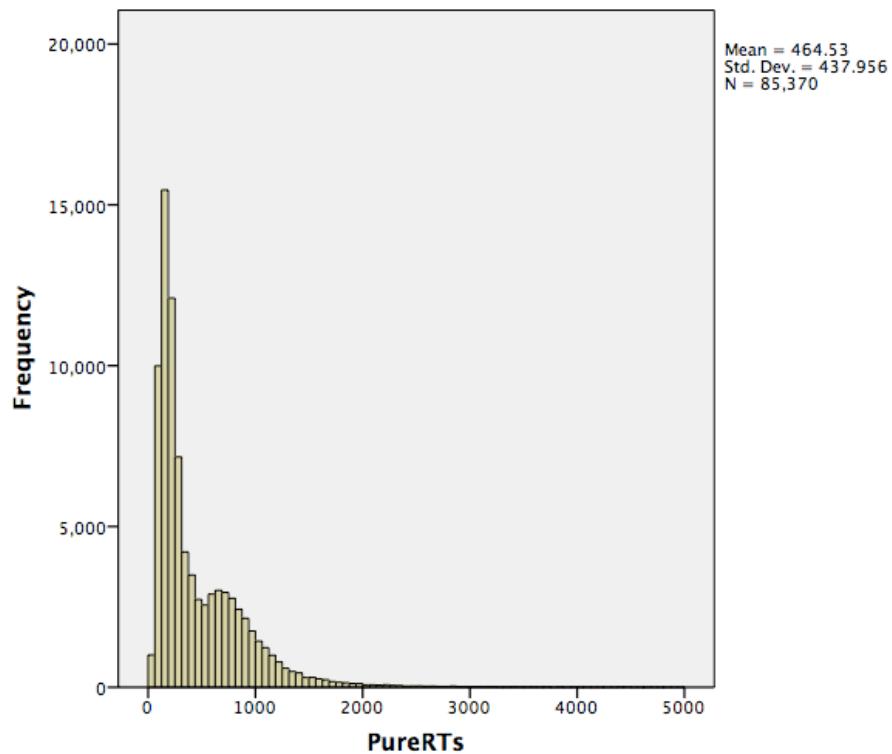
First, we'll be creating a histogram to look at our data. Go to Graphs, then Legacy Dialogs, then Histogram...



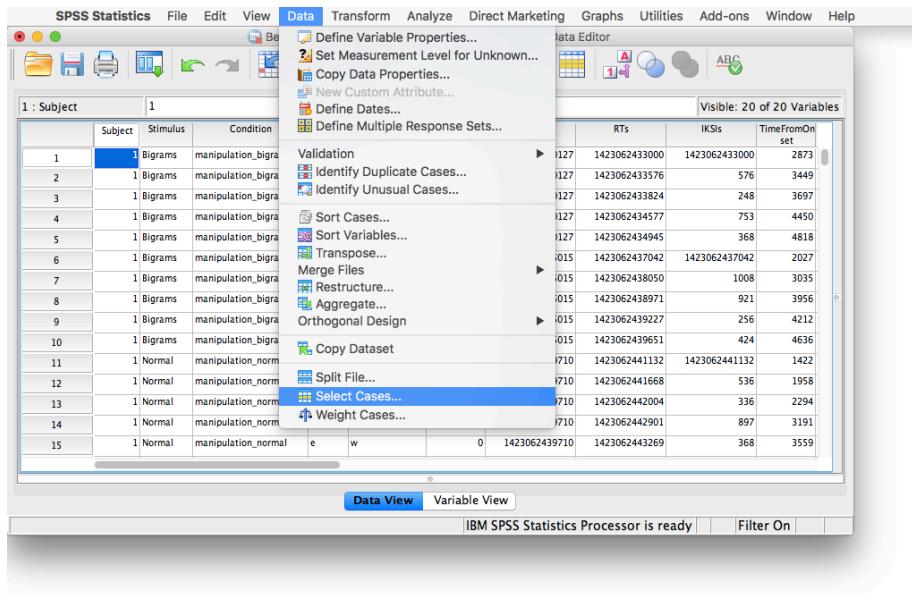
The next window will ask you to specify the variable you would like graphed. Choose PureRTs from the list on the left and move it into the “Variable” field using the arrow.



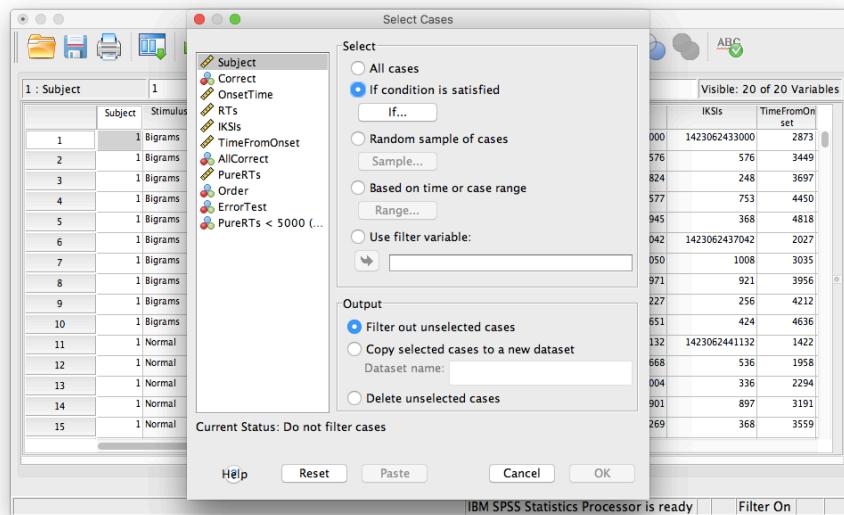
Click OK. SPSS will produce a frequency histogram of your data.

Graph

Notice that the RTs form a positively skewed distribution. Most RT measurements are on the lower end of the scale, and there are very few RTs that are higher than 5000ms. These measurements could have resulted from subjects falling asleep, looking at their phones, or being otherwise distracted in the middle of the experiment. We want to remove those values so they do not influence and mischaracterize our results. To do this, go to the top menu and choose Data, then Select Cases...

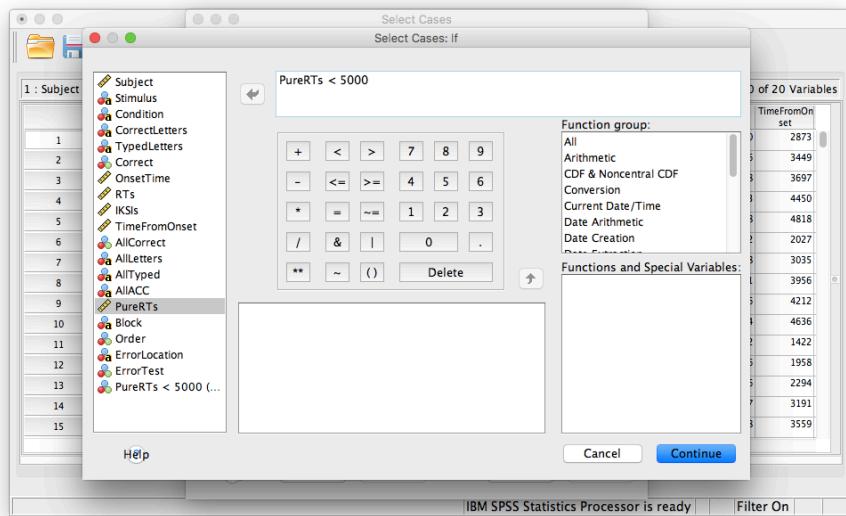


The next window will ask you which cases to select. We specifically want to remove any cases larger than 5000, so check off If condition is satisfied:



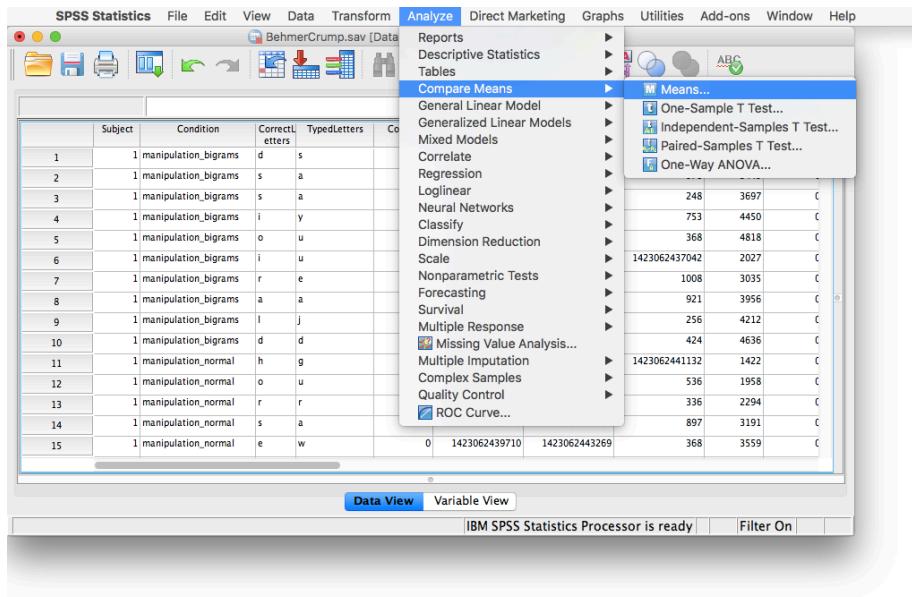
Then click If... The window that follows will ask you to specify the condition for inclusion. Find the variable PureRTs in the left-hand list and move it to the field on top using the arrow. Then, type "< 5000" to specify that we want to

keep only values under 5000ms.

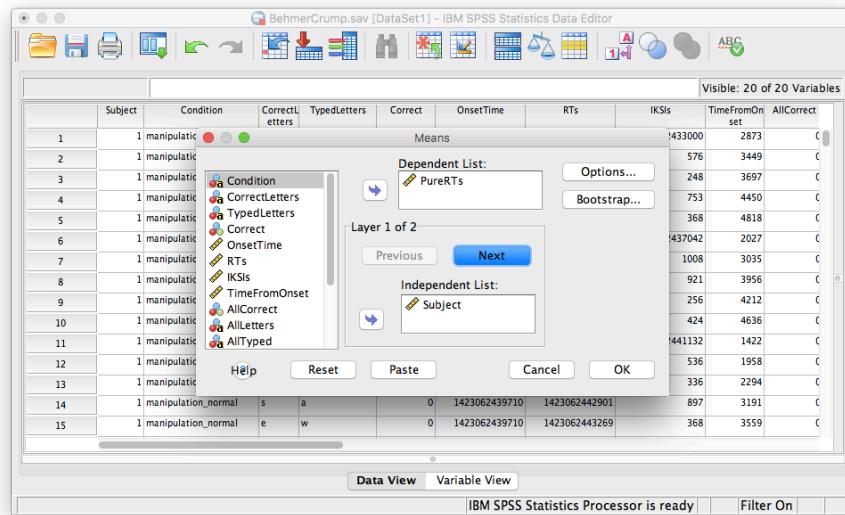


Click Continue. Your data has now been edited so that trials on which the PureRTs value was greater than 5000 have been removed.

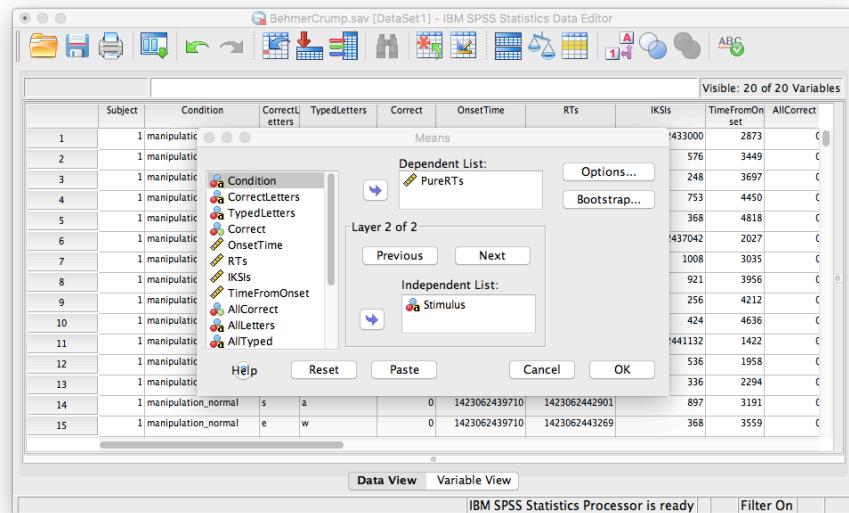
Let's take some time to think about this data. We have measurements taken from many participants across multiple trials and multiple conditions. Our goal is to be able to come up with a single RT for each person per condition. The only way to do this is to take an individual's RTs for a particular condition and average them. To do this, we need to calculate means. First, go to Analyze, then Compare Means, then Means...



In the next window, move the **PureRTs** variable into the “Dependent List” field. In the field below, labeled “Independent List”, place the **Subject** variable:



Then, click Next. The bottom field will become blank again, and you can now place the **Stimulus** variable into it.



Click OK. SPSS will produce an output table containing each subject's average RT per condition:

Report**PureRTs**

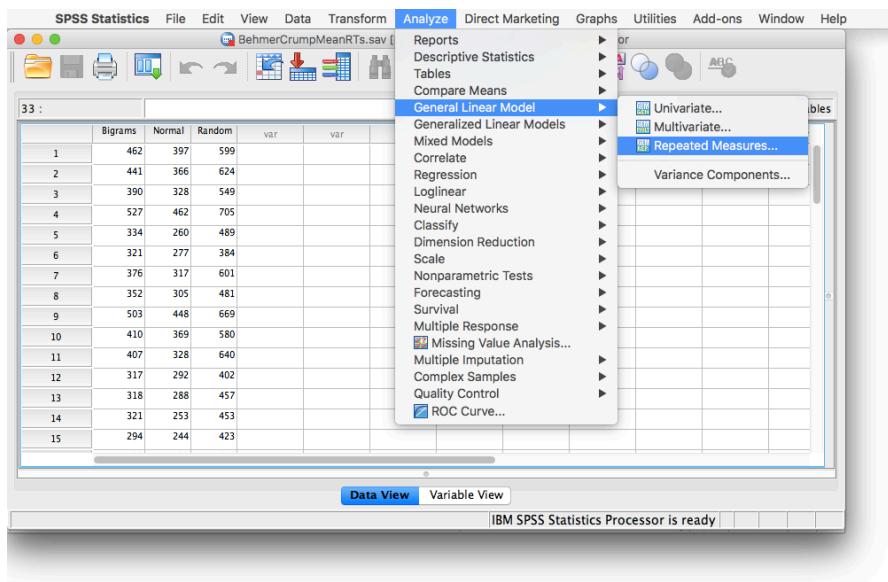
Subject	Stimulus	Mean	N	Std. Deviation
1	Bigrams	462.42	735	374.251
	Normal	397.40	750	321.305
	Random	598.83	740	374.474
	Total	485.87	2225	366.958
2	Bigrams	440.94	750	430.439
	Normal	366.36	745	358.975
	Random	624.19	740	505.774
	Total	476.75	2235	448.721
5	Bigrams	389.58	747	409.947
	Normal	328.07	749	386.280
	Random	548.98	744	442.168
	Total	421.96	2240	423.535
6	Bigrams	527.27	750	538.729
	Normal	462.13	740	414.229
	Random	704.91	724	585.611
	Total	563.59	2214	527.248
7	Bigrams	334.29	750	336.467
	Normal	260.28	749	258.486
	Random	488.82	749	425.538
	Total	361.12	2248	359.618
8	Bigrams	320.68	750	268.580
	Normal	277.22	750	233.588
	Random	383.95	750	274.545
	Total	327.29	2250	263.095
9	Bigrams	375.73	745	316.540
	Normal	317.15	745	319.449
	Random	600.84	750	468.480
	Total	431.62	2240	394.481
11	Bigrams	352.34	750	282.987
	Normal	304.88	750	313.460
	Random	480.92	750	332.309

9.6.3 Conduct and graph One-Factor Repeated Measures ANOVA

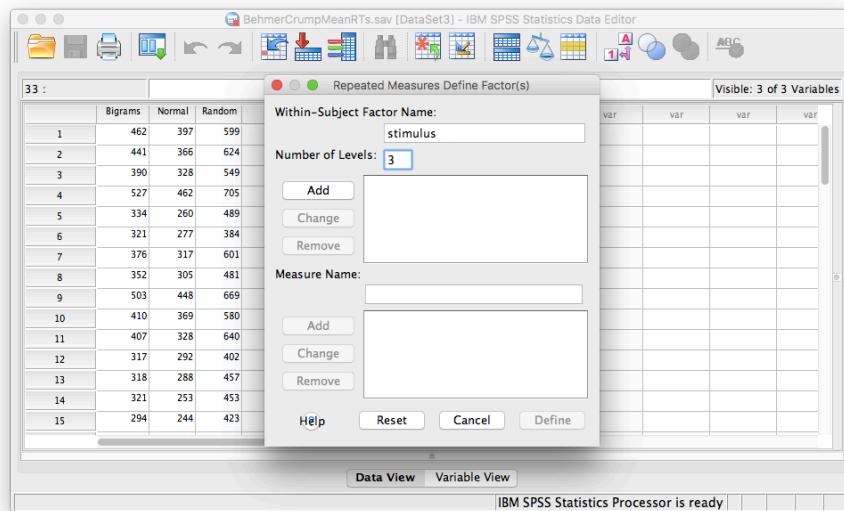
For this part of the tutorial, we will switch over to a new SPSS file. Here is the link; it's called BehmerCrumpMeanRTs.sav. When you open the file, it should look like this:

	Bigrams	Normal	Random	var							
1	462	397	599								
2	441	366	624								
3	390	328	549								
4	527	462	705								
5	334	260	489								
6	321	277	384								
7	376	317	601								
8	352	305	481								
9	503	448	669								
10	410	369	580								
11	407	328	640								
12	317	292	402								
13	318	288	457								
14	321	253	453								
15	294	244	423								

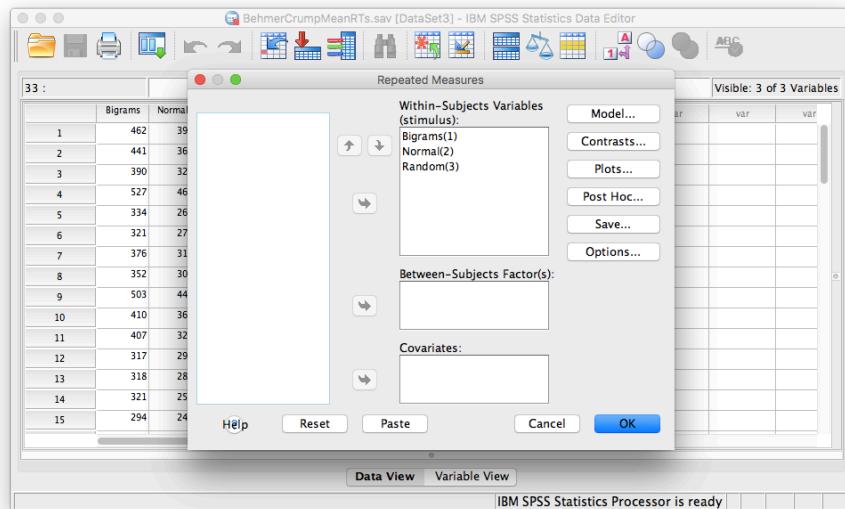
Notice that this file contains the means from the table we just created. Each person's data is contained within a row: there are 3 measurements corresponding to all stimulus conditions. To calculate a One-Factor Repeated Measures ANOVA, go to Analyze, then General Linear Model, then Repeated Measures...



The next window will ask you to label your within-subjects variable. Type **Stimulus** into the “Within-Subjects Factor Name” field. Indicate that this variable has 3 levels. Then click Add, and then Define.



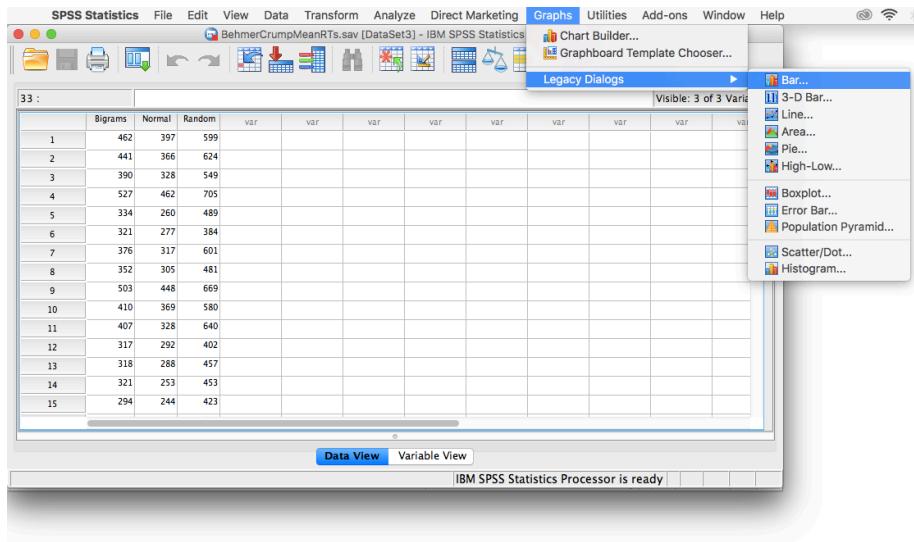
In the next window, move all 3 conditions listed on the left into the “Within-Subjects Variables” field.



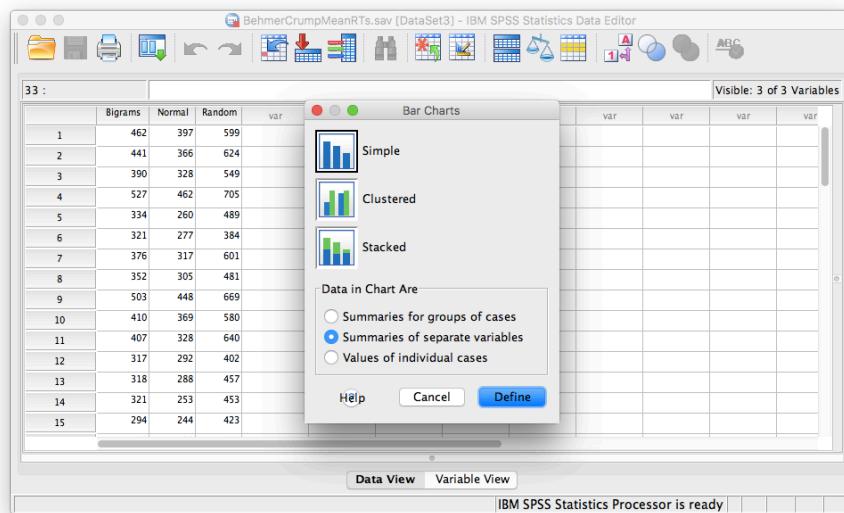
Click OK. SPSS will produce an output table labeled “Tests of Within-SUBjects Effects” that contains The F statistic and its corresponding p-value.

Tests of Within-Subjects Effects						
Measure: MEASURE_1		Type III Sum of Squares	df	Mean Square	F	Sig.
stimulus	Sphericity Assumed	1070038.49	2	535019.247	420.334	.000
	Greenhouse-Geisser	1070038.49	1.174	911071.396	420.334	.000
	Huynh-Feldt	1070038.49	1.190	899233.864	420.334	.000
	Lower-bound	1070038.49	1.000	1070038.49	420.334	.000
Error(stimulus)	Sphericity Assumed	94190.478	74	1272.844		
	Greenhouse-Geisser	94190.478	43.456	2167.496		
	Huynh-Feldt	94190.478	44.028	2139.334		
	Lower-bound	94190.478	37.000	2545.689		

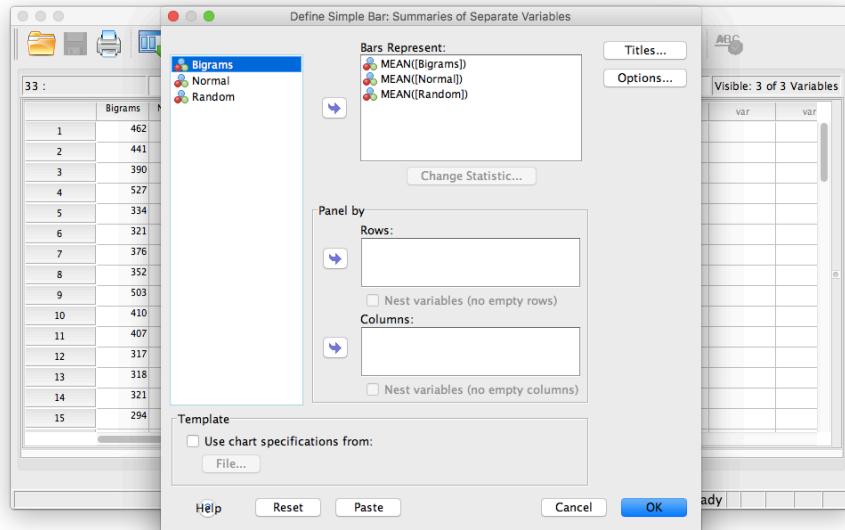
From this table, we see that there is a significant effect of stimulus on RT, $F(2, 74) = 420.33$, $p < .05$. In order to graph this data and look at the means, go to Graphs, then Legacy Dialogs, then Bar...



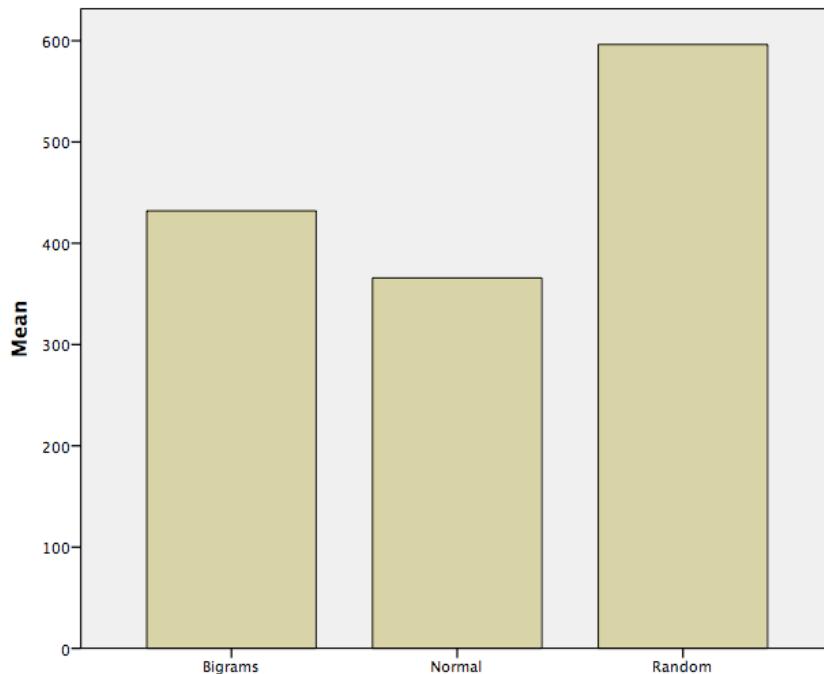
In the next window, choose Simple, but this time select Summaries of separate variables, then Define:



In the next window, move the three stimulus conditions listed in the field on the left into the field labeled “Bars Represent.”



Then click OK. SPSS will produce a bar graph with each bar representing the mean of each of the conditions. We can see that the Random condition has the longest RT, followed by Bigrams, and then Normal. However, at this point we do not know which groups are significantly different from the others.

Graph

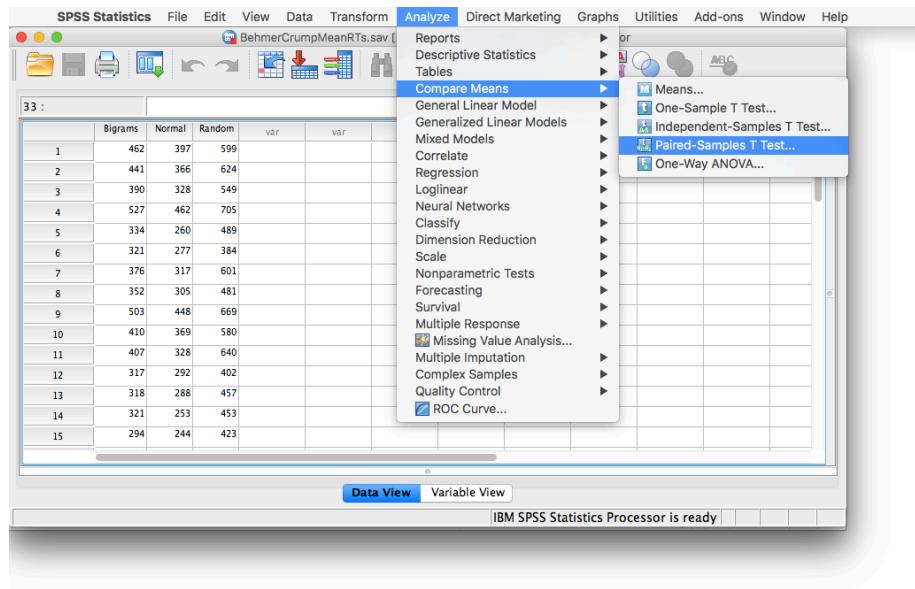
9.6.4 Conduct planned comparisons using a paired-samples t-test

To find out where the difference among these 3 conditions exists, we will use a planned comparison in the form of a paired-samples t-test. This method takes 2 groups at a time and tests for pairwise differences. There are three comparisons that can be made here:

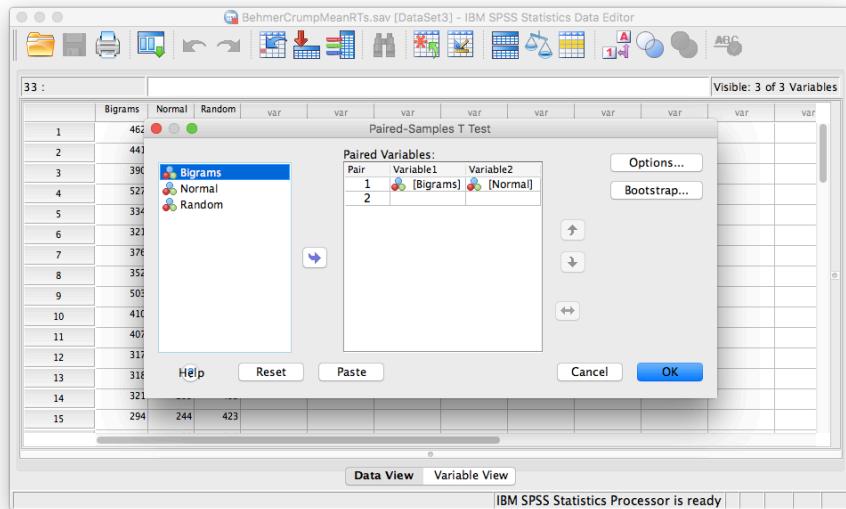
1. normal vs. bigram
2. bigram vs. random
3. random vs. normal

We will use a paired-samples t-test instead of an independent-samples t-test because in each comparison, the same people are being remeasured in each group.

To begin, go to Analyze, then Compare Means, then Paired Samples T-test...



In the next window, move the two conditions of our first pairwise comparison (normal vs. bigram) into the field on the right using the arrow.



Click OK. SPSS will produce a series of tables. The “Paired Samples test” table contains the t-statistic and its associated p-value.

T-Test

Paired Samples Statistics				
	Mean	N	Std. Deviation	Std. Error Mean
Pair 1 Bigrams	431.99	38	103.630	16.811
Normal	365.76	38	85.689	13.901

Paired Samples Correlations			
	N	Correlation	Sig.
Pair 1 Bigrams & Normal	38	.963	.000

Paired Samples Test							
	Paired Differences			95% Confidence Interval of the Difference	t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean				
	66.226	31.197	5.061	55.972	76.480	13.086	37 .000
Pair 1 Bigrams - Normal							

Looking at this output, we can see that the Normal and Bigrams conditions are indeed significantly different, $t(37) = 13.09$, $p < .05$.

You can use this same method for the remaining comparisons.

9.6.5 Practice Problems

-
1. Run the same analysis as illustrated in this lab tutorial but with accuracy (correct) as the dependent variable. Use an alpha level of .05. Remember to calculate means per subject and stimulus first. This will generate a table, whose values you can enter into a new SPSS spreadsheet file.
 2. Is there an effect of stimulus on error rate? If so, conduct the appropriate planned comparisons.
 3. Graph the means using a bar graph; include +/-1 SEM Error bars.

9.7 JAMOVI

How to do it in JAMOVI

Chapter 10

Lab 10: Factorial ANOVA

Simplicity is complex. It's never simple to keep things simple. Simple solutions require the most advanced thinking. —Richie Norton

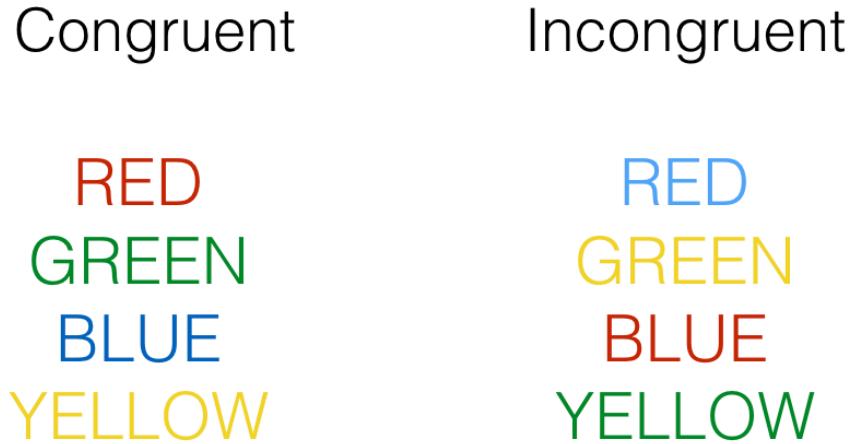
10.1 Does standing up make you focus more?

This lab activity uses the data from “Stand by your Stroop: Standing up enhances selective attention and cognitive control” (Rosenbaum, Mama, Algom, 2017) to teach the analysis of a 2x2 design using ANOVA. Although the research design is a 2x2 repeated measures design, we treat the design both as repeated measures, and as a between-subjects design to illustrate how to conduct either type of ANOVA in software.

10.1.1 STUDY DESCRIPTION

Do you pay more attention when you are sitting or standing? We analyse the data from “Stand by your Stroop: Standing up enhances selective attention and cognitive control” (Rosenbaum, Mama, Algom, 2017). This paper asked whether sitting versus standing would influence a measure of selective attention, the ability to ignore distracting information.

They used a classic test of selective attention, called the Stroop effect. In a typical Stroop experiment, subjects name the color of words as fast as they can. The trick is that sometimes the color of the word is the same as the name of the word, and sometimes it is not. Here are some examples:



Congruent trials occur when the color and word match. So, the correct answers for each of the congruent stimuli shown would be to say, red, green, blue and yellow. Incongruent trials occur when the color and word mismatch. The correct answers for each of the incongruent stimuli would be: blue, yellow, red, green.

The Stroop effect is an example of a well-known phenomena. What happens is that people are faster to name the color of the congruent items compared to the color of the incongruent items. This difference (incongruent reaction time - congruent reaction time) is called the Stroop effect.

Many researchers argue that the Stroop effect measures something about selective attention, the ability to ignore distracting information. In this case, the target information that you need to pay attention to is the color, not the word. For each item, the word is potentially distracting, it is not information that you are supposed to respond to. However, it seems that most people can't help but notice the word, and their performance in the color-naming task is subsequently influenced by the presence of the distracting word.

People who are good at ignoring the distracting words should have small Stroop effects. They will ignore the word, and it won't influence them very much for either congruent or incongruent trials. As a result, the difference in performance (the Stroop effect) should be fairly small (if you have "good" selective attention in this task). People who are bad at ignoring the distracting words should have big Stroop effects. They will not ignore the words, causing them to be relatively fast when the word helps, and relatively slow when the word mismatches. As a result, they will show a difference in performance between the incongruent and congruent conditions.

If we take the size of the Stroop effect as a measure of selective attention, we can then start wondering what sorts of things improve selective attention (e.g.,

that make the Stroop effect smaller), and what kinds of things impair selective attention (e.g., make the Stroop effect bigger).

The research question of this study was to ask whether standing up improves selective attention compared to sitting down. They predicted smaller Stroop effects when people were standing up and doing the task, compared to when they were sitting down and doing the task.

10.1.2 Study Methods

The design of the study was a 2x2 repeated-measures design. The first IV was congruency (congruent vs incongruent). The second IV was posture (sitting vs. standing). The DV was reaction time to name the word. There were 50 participants in the study.

10.2 Lab Skills Learned

- Conducting a 2x2 between-subjects ANOVA
- Conducting a 2x2 repeated-measures ANOVA

10.3 Important Stuff

- citation: Rosenbaum, D., Mama, Y., & Algom, D. (2017). Stand by Your Stroop: Standing Up Enhances Selective Attention and Cognitive Control. *Psychological science*, 28(12), 1864-1867.
- Link to .pdf of article
- Data in .csv format

10.4 R

10.4.1 Load the data

Remember that any line with a # makes a comment and the code does not run. Below is how to load the .csv data from the online repository, or from a local file (you need to change the file path to where the local file is, if you downloaded it). The data contains all of the measures and conditions from Experiment 1 in the paper.

```
library(data.table)
#all_data <- fread("https://github.com/CrumpLab/statisticsLab/raw/master/stroop_stand.csv")
all_data <- fread("data/stroop_stand.csv")
```

10.4.2 Inspect the dataframe

This will give you a big picture of the data frame. Click the button to view it in your browser, then take a look to see what is in it.

```
library(summarytools)
view(dfSummary(all_data))
```

We see there are four columns of numbers. The column names tell us whether the data is for a congruent or incongruent condition, and whether the participant was sitting or standing. Note, this data is in wide-format, not long-format. Each subject has 4 measures per row. We will need to change this to work with the data in R.

10.4.3 Get the data you need

This data file contains all of the data from Experiment 1 in the paper. So, we don't need to get rid of any rows.

10.4.4 Get the data into the format you want

As mentioned before we need to convert the data from wide to long format. What we want at the end of this conversion is:

1. A column for the subject variable
2. A column for the congruency variable
3. A column for the posture (sit vs. stand) variable
4. A column for the DV (mean reaction times)

We look at two ways to do the transformation from wide to long. The first way is to “do it by hand”, which refers to creating every variable individually, and then putting them together in a single data frame. This next bit of code does this, and you can check out `stroop_df` to see the result.

```
RTs <- c(as.numeric(unlist(all_data[,1])),
          as.numeric(unlist(all_data[,2])),
          as.numeric(unlist(all_data[,3])),
          as.numeric(unlist(all_data[,4])))
)

Congruency <- rep(rep(c("Congruent", "Incongruent"), each=50), 2)
Posture <- rep(c("Stand", "Sit"), each=100)
Subject <- rep(1:50, 4)

stroop_df <- data.frame(Subject, Congruency, Posture, RTs)
```

Another way to transform between long and wide is to use R functions that were designed to do this job. For example there are the `spread` and `gather` functions from the `tidyverse` package, and the `melt` and `cast` functions, which also

do some data frame transforming. The transformation from wide to long can be complicated depending on the structure of the data, and you may often find it helpful to google these functions to look for more examples of their use.

Let's use the `tidyR gather` function to change our data from wide to long

```
library(tidyR)

stroop_long <- gather(all_data, key=Condition, value=RTs,
                      congruent_stand, incongruent_stand,
                      congruent_sit, incongruent_sit)
```

Take a moment to look at `stroop_long`. It is almost what we need. It is certainly in long format. There is a column for Subjects, and a column for the RTs, but there is only one column for both IVs, that's no good. There are two IVs, we need two columns. Fortunately, the levels in the new Condition column are coded with a specific and consistent structure:

1. congruent_stand
2. incongruent_stand
3. congruent_sit
4. incongruent_sit

If only we could split these by the “_” (underscore), then we would have two columns for the congruency and the posture variable. We can do this using `tstrsplit` from the `data.table` package

```
new_columns <- tstrsplit(stroop_long$Condition, "_", names=c("Congruency", "Posture"))
```

You can look inside `new_columns` to see that we successfully made the split. Now, we just need to add them on to the `stroop_long` data frame.

```
stroop_long <- cbind(stroop_long, new_columns)
```

Look at the `stroop_long` data frame and you will find that we have added two new columns, one that codes for Congruency, and the other that codes for posture.

Using this method we still haven't added a column for subjects. We can do that like this:

```
stroop_long <- cbind(stroop_long, Subject=rep(1:50, 4))
```

10.4.4.1 The independent variables

After all of this data transformation you should be familiar with the IVs.

1. Congruency: congruent vs. incongruent
2. Posture: stand vs. sit

10.4.4.2 The dependent variables

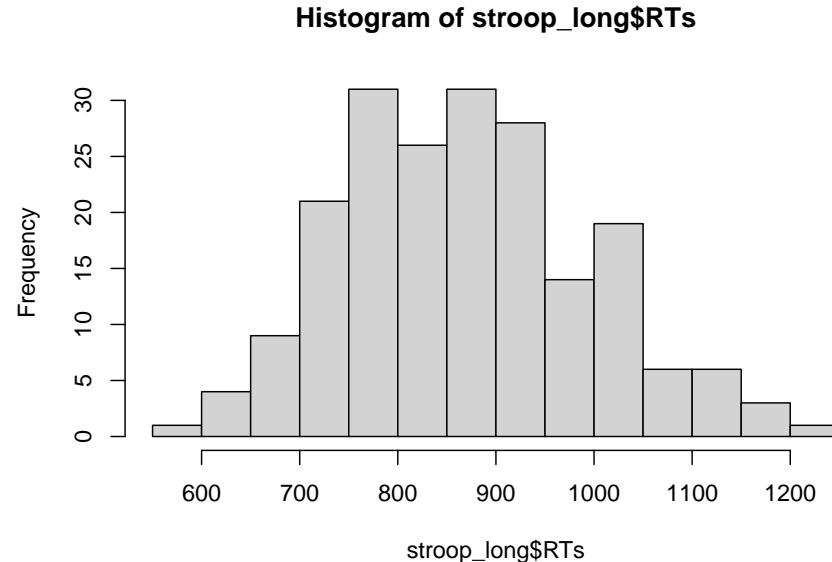
There is only one DV that we look at, that is the mean reaction time to name the color.

10.4.5 Look at the data

Remember before we do any analysis, we always want to “look” at the data. This first pass let’s us know if the data “look right”. For example, the data file could be messed up and maybe there aren’t any numbers there, or maybe the numbers are just too weird.

Let’s make a quick histogram of all of the RT data, like this:

```
hist(stroop_long$RTs)
```



This looks pretty good, there are no super huge numbers here.

10.4.6 Look at the means

As part of looking at the data, we might as well make a figure that shows the mean reaction times in each condition, and some error bars to look at the spread in each condition. The following code takes two important steps:

1. Get the means for each condition, by averaging over the means for each subject. These are put into the data frame called `plot_means`.
2. Make a graph with the `plot_means` data frame using `ggplot2`.

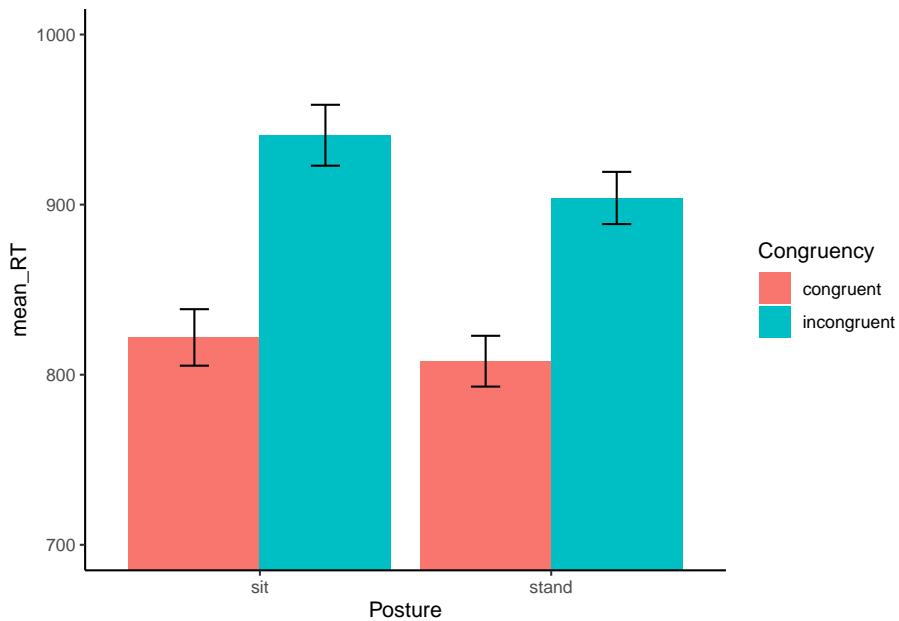
```

library(dplyr)
library(ggplot2)

plot_means <- stroop_long %>%
  group_by(Congruency, Posture) %>%
  summarise(mean_RT = mean(RTs),
            SEM = sd(RTs)/sqrt(length(RTs)))

ggplot(plot_means, aes(x=Posture, y=mean_RT, group=Congruency, fill=Congruency))+
  geom_bar(stat="identity", position="dodge")+
  geom_errorbar(aes(ymin=mean_RT-SEM, ymax=mean_RT+SEM),
                position=position_dodge(width=0.9),
                width=.2)+
  theme_classic()+
  coord_cartesian(ylim=c(700,1000))

```



10.4.7 Conduct the ANOVA

In this lab we will show you how to conduct ANOVAs for factorial designs that are for:

1. fully between-subjects designs (both IVs are between-subjects IVs)
2. fully repeated measures designs (both IVs are repeated measures)

The data we are looking at right now is fully repeated measures.

However, in this lab we are first going to pretend that the experiment was not repeated measures. We are going to pretend it was fully between-subjects. Then we are going to conduct a between-subjects ANOVA. After that, we will conduct a repeated-measures ANOVA, which is what would be most appropriate for this data set. The overall point is to show you how to both of them, and the discuss how to interpret them and write them both up.

10.4.8 Between Subjects ANOVA

We can always conduct a between-subjects version of the ANOVA on repeated-measures data if we wanted to. In this case we wouldn't really want to do this. But, we will do this for educational purposes to show you how to do it in R.

The syntax is very similar to what we do for one-way ANOVAs, remember the syntax was:

```
aov(DV ~ IV, dataframe)
```

If you want to add another IV, all you need to do is insert another one into the formula, like this:

```
aov(DV ~ IV1*IV2, dataframe)
```

Just, switch DV to the name of your dependent measure, and IV1 and IV2 to the names of your independent variables. Finally, put the name of your dataframe. Your dataframe must be in long format with one observation of the DV per row.

Our formula will look like this:

```
aov(RTs ~ Congruency*Posture, stroop_long)
```

In plain language, this formula means, analyze RTs by the Congruency and Posture Variables. R will automatically produce the main effects for Congruency and Posture, as well as the interaction (Congruency X Posture). Also, remember, that in the following code, we use a few other functions so that we can print out the results nicely.

```
library(xtable)

aov_out<-aov(RTs ~ Congruency*Posture, stroop_long)
summary_out<-summary(aov_out)

library(xtable)
knitr::kable(xtable(summary_out))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Congruency	1	576821.635	576821.635	43.7344419	0.0000000
Posture	1	32303.453	32303.453	2.4492381	0.1191950
Congruency:Posture	1	6560.339	6560.339	0.4974029	0.4814808
Residuals	196	2585080.215	13189.185	NA	NA

We can also have R print out the Grand Mean, the means for each level of each main effect, and the means for the interaction term. This is the same print out you would get in the console for R. It is admittedly not very pretty. There's probably a way to make the means provided by `model.tables()` more pretty. If we find a way we will update this section, if you find a way, please let us know.

```
print(model.tables(aov_out, "means"), format="markdown")

## Tables of means
## Grand mean
##
## 868.6454
##
## Congruency
## Congruency
##   congruent incongruent
##     814.9      922.3
##
## Posture
## Posture
##   sit stand
## 881.4 855.9
##
## Congruency:Posture
##           Posture
## Congruency   sit   stand
##   congruent    821.9 808.0
##   incongruent 940.8 903.9
```

10.4.8.1 ANOVA write-up

Here are the steps for writing up the results of an ANOVA:

1. Say what means you analyzed
2. Say what test you performed
3. Say the inferential statistic for each of the effects (main effects and interaction)
4. Say the pattern of results for each effect.

A short example of the whole write-up is below:

Example write-up

We submitted the mean reaction times for each group to a 2 (Congruency: congrueny vs. incongruent) x 2 (Posture: Standing vs. Sitting) between-subjects ANOVA.

There was a main effect of Congruency, $F(1, 196) = 43.73$, $MSE = 13189.185$, $p < 0.001$. Mean reaction times were slower for incongruent (922 ms) than congruent groups (815 ms).

There main effect of Posture was not significant, $F(1, 196) = 2.45$, $MSE = 13189.185$, $p = .119$. Mean reaction times were slower for sitting (881 ms) than standing groups (855 ms).

The two-way interaction between Congruency and Posture was not significant, $F(1, 196) = .497$, $MSE = 13189.185$, $p < 0.481$.

For every F-value, we report $F(df_1, df_2) = F\text{-value}$, $MSE = MSE$ for the error term, and $p = x.xxx$.

In R, the df_1 , for the df in the numerator is always listed beside the name for a particular effect. For example, Congruency has 1 degree of freedom (there are two condition, and $2-1=1$). Similarly, the relevant F and p-value are listed in the same row as the effect of interest.

However, the error term used to calculate the F-value is listed at the bottom, in R this is called “Residuals”. Df_2 , the df for the denominator is listed beside “Residuals”, in our case it was 196. The important bit is the MSE, which was 13189.185. Notice, that in the write up for each main effect and interaction we always reported the same MSE. That’s because in this between-subjects version of the ANOVA, we divide by same very same error term. Also notice that we don’t report the sums of squares of the MSE for the effect.

Why not? The main reason why not is that you can reconstruct those missing numbers just by knowing the dfs, the MSE for the error, and the f-value.

For example, you can get the MSE for the effect by multiplying the F-value by the MSE for the error. Now you have both MSEs. You can get both Sums of Squares by multiplying by their associated dfs. That’s just working backwards from the F-value.

You can always check if you are reporting the correct MSE for the error term. If the MSE for your effect (numerator) divided by the MSE you are using for the error term does not equal the F-value, then you must be using the wrong terms!

10.4.9 Repeated measures ANOVA

Of course, the design for this experiment was not between-subjects, it was fully within-subjects. Every participant completed both congruent and incongruent trials, while they were standing or sitting. For this reason, we should conduct a repeated measures ANOVA. This way we will be able capitolize on the major benefit provided by the repeated measures design. We can remove the variance

due to individual subjects from the error terms we use to calculate F-values for each main effect and interaction.

Remember the formula for the one-factor repeated-measures ANOVA, we'll remind you:

```
aov( DV ~ IV + Error(Subject/IV), dataframe)
```

To do the same for a design with more than one IV we put in another IV to the formula, like this:

```
aov( DV ~ IV1*IV2 + Error( Subject/(IV1*IV2) ), dataframe)
```

- DV = name of dependent variable
- IV1 = name of first independent variable
- IV2 = name of second independent variable
- Subject = name of the subject variable, coding the means for each subject in each condition
- dataframe = name of the long-format data frame

Here is what our formula will look like:

```
aov(RTs ~ Congruency*Posture + Error(Subject/(Congruency*Posture)),  
stroop_long)
```

The main thing you need to watch out for when running this analysis in R, is that all your factors need to be **factors** in R. Often times people will use numbers rather than words or letters to code the levels for specific factors. This can be very often done for the subjects factor, using number 1 for subject one, and number 2 for subject 2. If you want your column variable to be treated as a factor, then you may need to convert it to a factor. We do this below for the Subject variable, which happens to be coded as numbers. If we do not do this, the repeated-measures ANOVA will return incorrect results.

For example, if you look at the **stroop_long** data frame, and click the little circle with an arrow on it in the environment panel, you should see that Subject is an **int**. That stands for integer. You should also see that Congruency and Posture are **Factor**, that's good. We need to turn Subject into **Factor**.

```
stroop_long$Subject <- as.factor(stroop_long$Subject) #convert subject to factor  
  
summary_out<-aov(RTs ~ Congruency*Posture + Error(Subject/(Congruency*Posture)), stroop_long)  
  
library(xtable)  
knitr::kable(xtable(summary_out))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Residuals	49	2250738.636	45933.4416	NA	NA
Congruency	1	576821.635	576821.6349	342.452244	0.0000000
Residuals	49	82534.895	1684.3856	NA	NA
Posture	1	32303.453	32303.4534	7.329876	0.0093104
Residuals1	49	215947.614	4407.0942	NA	NA
Congruency:Posture	1	6560.339	6560.3389	8.964444	0.0043060
Residuals	49	35859.069	731.8177	NA	NA

```
print(model.tables(aov_out,"means"), format="markdown")
```

```
## Tables of means
## Grand mean
##
## 868.6454
##
## Congruency
## Congruency
##   congruent incongruent
##      814.9      922.3
##
## Posture
## Posture
##   sit stand
## 881.4 855.9
##
## Congruency:Posture
##           Posture
## Congruency   sit   stand
##   congruent  821.9 808.0
##   incongruent 940.8 903.9
```

What's different here? Are any of the means different now that we have conducted a repeated-measures version of the ANOVA, instead of the between-subjects ANOVA? NO! The grand mean is still the grand mean. The means for the congruency conditions are still the same, the means for the Posture conditions are still the same, and the means for the interaction effect are still the same. The only thing that has changed is the ANOVA table. Now that we have removed the variance associated with individual subjects, our F-values are different, and so are the p-values. Using an alpha of 0.05, all of the effects are "statistically significant".

Each main effect and the one interaction all have their own error term. In the table below, R lists each effect in one row, and then immediately below lists the error term for that effect.

10.4.9.1 ANOVA write-up

Here is what a write-up would look like.

Example write-up

We submitted the mean reaction times for each subject in each condition to a 2 (Congruency: congrueny vs. incongruent) x 2 (Posture: Standing vs. Sitting) repeated measures ANOVA.

There was a main effect of Congruency, $F(1, 49) = 342.45$, MSE = 1684.39, $p < 0.001$. Mean reaction times were slower for incongruent (922 ms) than congruent groups (815 ms).

The main effect of Posture was significant, $F(1, 49) = 7.33$, MSE = 4407.09, $p = .009$. Mean reaction times were slower for sitting (881 ms) than standing groups (855 ms).

The two-way interaction between Congruency and Posture was significant, $F(1, 49) = 8.96$, MSE = 731.82, $p < 0.004$. The Stroop effect was 23 ms smaller in the standing than sitting conditions.

10.4.10 Follow-up comparisons

In a 2x2 ANOVA there are some follow-up comparisons you may be interested in making that are not done for you with the ANOVA. If an IV only have 2 levels, then you do not have to do any follow-up tests for the main effects of those IVs (that's what the main effect from the ANOVA tells you). So, we don't have to do follow-up comparisons for the main effects of congruency or posture. What about the interaction?

Notice the interaction is composed of four means. The mean RT for congruent and incongruent for both sitting and standing.

Also notice that we only got one F-value and one p-value from the ANOVA for the interaction term. So, what comparison was the ANOVA making? And what comparisons was it not making? It has already made one comparison, so you do not need a follow-up test for that...which one is it?

If you remember back to the textbook, we should you how to analyze a 2x2 design with paired-samples t-tests. We analyzed the interaction term as the comparison of difference scores. Here, we would find the differences scores between incongruent and congruent for each level of the posture variable. In other words, we compute the Stroop effect for each subject when they were sitting and standing, then compare the two Stroop effects. This comparison is looking at the difference between two differences scores, and that is the comparison that the ANOVA does for the interaction. To be more precise the comparison is:

$$(sit : incongruent - sit : congruent) - (stand : incongruent - stand : congruent)$$

What comparisons are not made, what are the other ones we could do? Here are some:

1. sit:congruent vs sit:incongruent
2. stand:congruent vs stand:incongruent
3. sit:congruent vs stand:incongruent
4. stand:congruent vs sit:incongruent

We could add a few more. These kinds of comparisons are often called **simple effects**, apparently referring to the fact they are just comparing means in a straight forward way. There are a few different comparisons we could do. Should we do any of them?

Whether or not you compare means usually depends on the research question you are asking. Some comparisons make sense within context of the research question, and others may not. We will do two follow-up comparisons. Our question will be about the size of the Stroop effect in the Sitting and Standing conditions. We already know that the size of the effect was smaller in the Standing condition. But, we don't know if it got so small that it went away (at least statistically speaking). Now, we can ask:

1. Was the Stroop effect only for the sitting condition statistically significant. In other words, was the difference in mean RT between the incongruent and congruent conditions unlikely under the null (or unlikely to be produced by chance)
2. Was the Stroop effect only for the standing condition statistically significant. In other words, was the difference in mean RT between the incongruent and congruent conditions unlikely under the null (or unlikely to be produced by chance)

We can answer both of the questions using paired sample t-tests comparing the means in question

10.4.10.1 Sitting Stroop

```
means_to_compare <- stroop_long %>%
  filter(Posture=="sit")

t.test(RTs~Congruency, paired=TRUE, var.equal=TRUE, data=means_to_compare)

## 
## Paired t-test
## 
## data: RTs by Congruency
## t = -16.516, df = 49, p-value < 2.2e-16
```

```
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
## -133.3250 -104.3997
## sample estimates:
## mean difference
## -118.8623
```

10.4.10.2 Standing Stroop

```
means_to_compare <- stroop_long %>%
  filter(Posture=="stand")

t.test(RTs~Congruency, paired=TRUE, var.equal=TRUE, data=means_to_compare)

##
## Paired t-test
##
## data: RTs by Congruency
## t = -14.327, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
## -109.41185 -82.49462
## sample estimates:
## mean difference
## -95.95323
```

10.4.11 Generalization Exercise

Here are some means for four conditions in a 2x2 Design:

		IV1	
		Level 1	Level 2
IV 2	Level 1	100	200
	Level 2	200	500

Your task is to:

- A. Compute the mean difference for the main effect of IV1
- B. Compute the mean difference for the main effect of IV2
- C. Compute the mean difference for the interaction

10.4.12 Writing assignment

(2 points - Graded)

Factorial designs have main effects and interactions.

1. Explain the concept of a main effect. (1 point)
2. Explain the concept of an interaction. (1 point)

General grading.

- You will receive 0 points for missing answers
- You must write in complete sentences. Point form sentences will be given 0 points.
- Completely incorrect answers will receive 0 points.
- If your answer is generally correct but very difficult to understand and unclear you may receive half points for the question

10.5 Excel

How to do it in Excel

10.6 SPSS

In this lab, we will use SPSS to:

1. Conduct and graph a Between-Subjects Two-Factor Analysis of Variance (ANOVA)
2. Calculate simple effects
3. Conduct and graph a Repeated Measures Two-Factor Analysis of Variance (ANOVA)
4. Calculate simple effects

10.6.1 Experiment Background

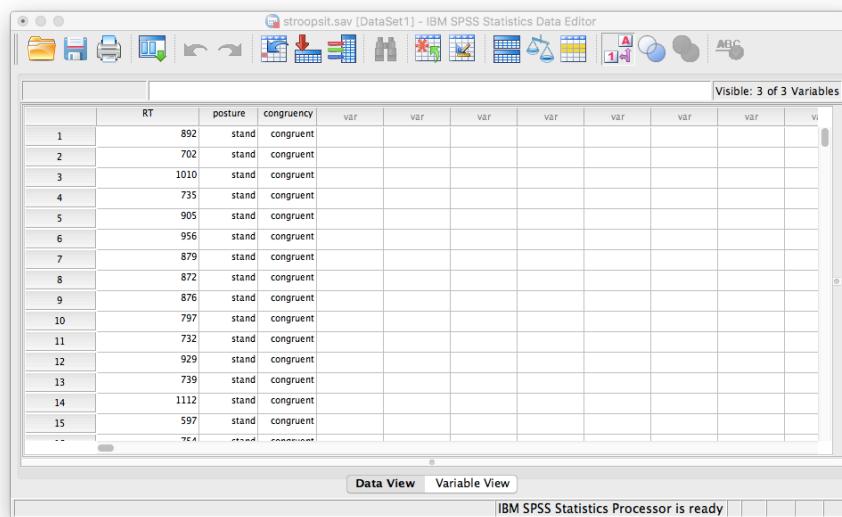
The Rosenbaum, Mama, and Algom (2017) paper asked whether sitting versus standing would influence a measure of selective attention, the ability to ignore distracting information. Selective attention here is measured as performance on the Stroop task.

In a typical Stroop experiment, subjects name the color of words as fast as they can. The trick is that sometimes the color of the word is the same as the name of the word, and sometimes it is not.

The design of the study was a 2x2 design. The first IV was **congruency** (congruent vs incongruent). The second IV was **posture** (sitting vs. standing). The DV was reaction time (RT) to name the word.

10.6.2 Conduct a Between-Subjects Two-Factor Analysis of Variance (ANOVA)

Here is a link to our data file. It is named stroopsit.sav. Your data should look like this:

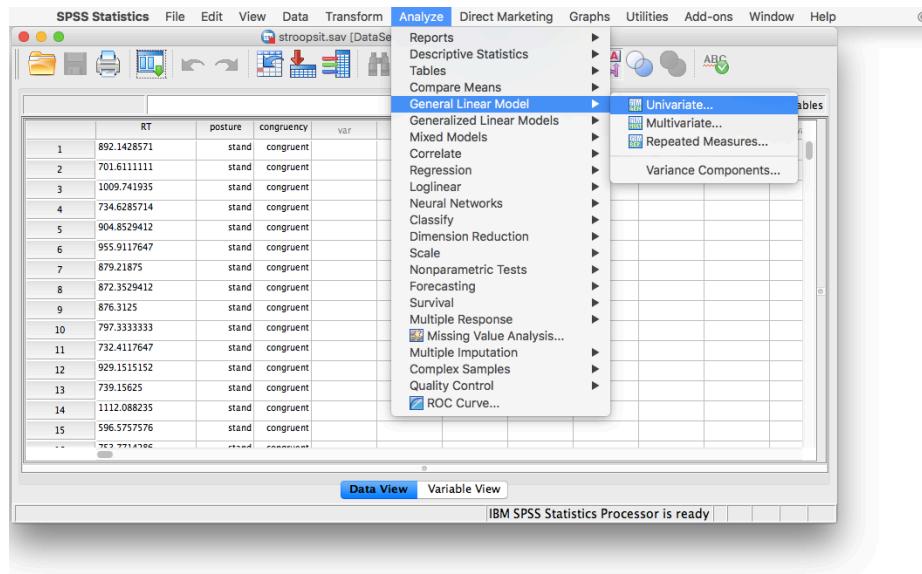


	RT	posture	congruency	var	Vi							
1	892	stand	congruent									
2	702	stand	congruent									
3	1010	stand	congruent									
4	735	stand	congruent									
5	905	stand	congruent									
6	956	stand	congruent									
7	879	stand	congruent									
8	872	stand	congruent									
9	876	stand	congruent									
10	797	stand	congruent									
11	732	stand	congruent									
12	929	stand	congruent									
13	739	stand	congruent									
14	1112	stand	congruent									
15	597	stand	congruent									
--	--	--	--									

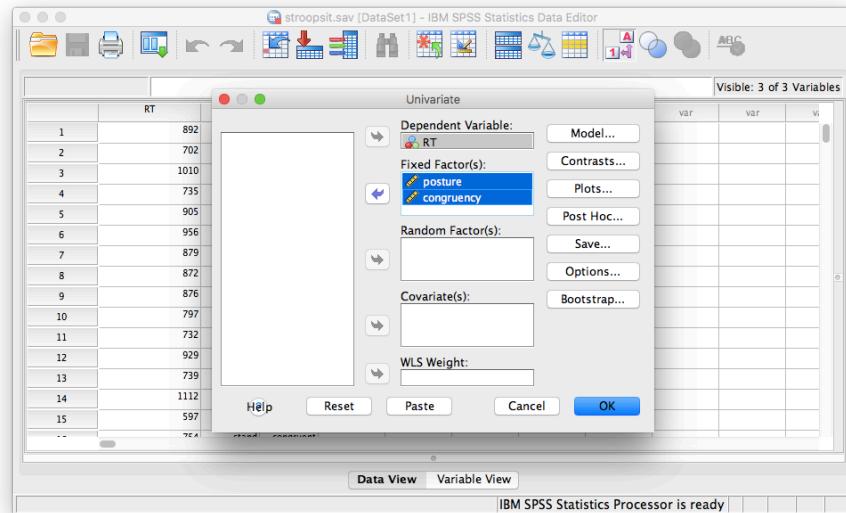
Notice that in this file, there are 200 rows (corresponding to 200 subjects). Each subject is categorized according to 2 independent variables: posture (whether they were in the standing or sitting condition), and congruency (whether the stimuli they received were congruent or incongruent). In this application, we are treating the design as between-subjects. This means each participant only experienced one of the following four conditions:

1. congruent stand
2. incongruent stand
3. congruent sit
4. incongruent sit

Now, let's run our Two-Factor ANOVA. Go to Analyze, then General Linear Model, then Univariate...

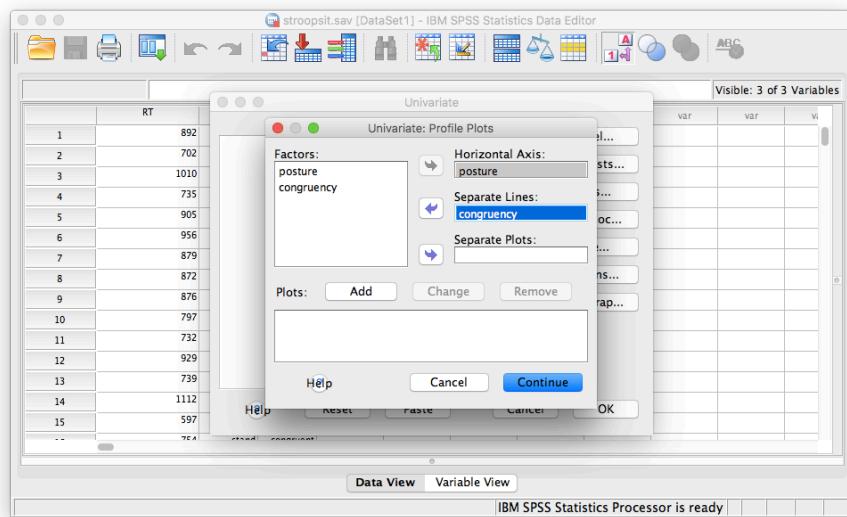


A window will appear asking for you to specify your variables. Place RT into the “Dependent Variable” field, and put both “posture” and “congruency” into the “Fixed Factors” field.

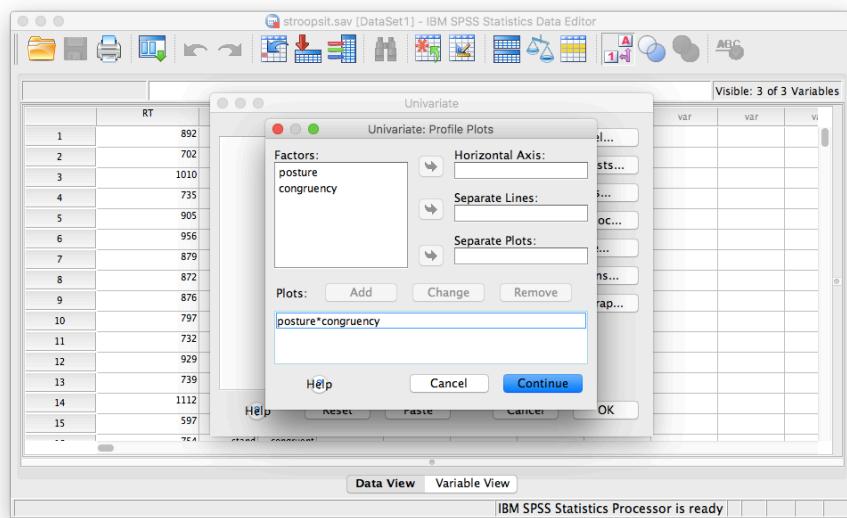


Now, before pressing anything else, click on Plots... (this will allow us to create a plot of means right along with our SPSS output). In the plots window, you must place the **congruency** and **posture** variables into the fields labeled “Horizontal

Axis” and “Separate Lines”. You can place either variable in either field, and the graph will still make sense. However, for this example, I will be putting **congruency** into the “Separate Lines” field, and **posture** in “Horizontal Axis”:



You must click Add for this table to be included in your SPSS output. After you do this, Your window should look like this:



Then click Continue and OK. Your output will include a table labeled “Tests of Between Subjects Effects” and a plot of the means.

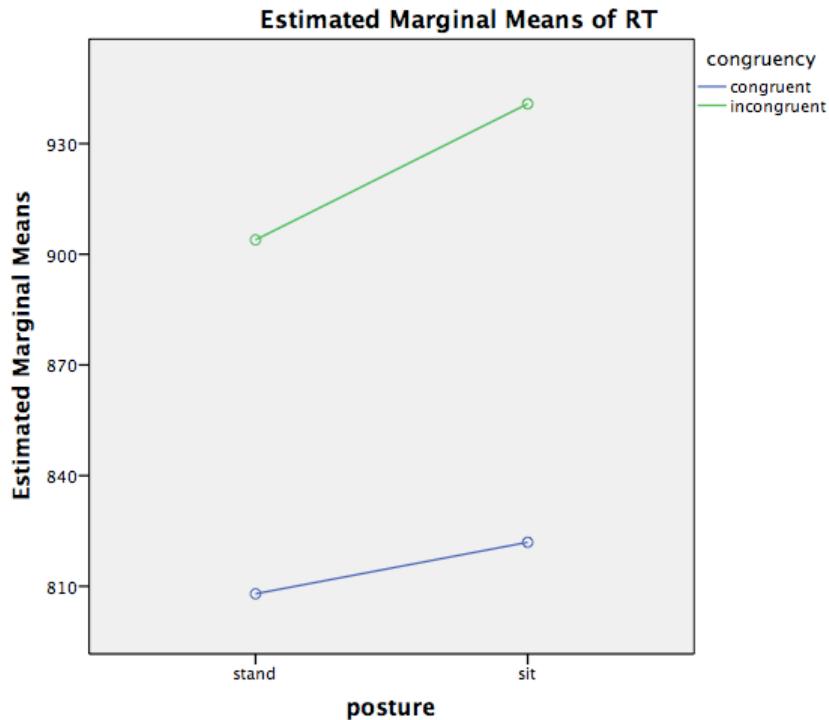
Tests of Between-Subjects Effects

Dependent Variable: RT

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	615685.43 ^a	3	205228.476	15.560	.000
Intercept	150908975	1	150908975	11441.873	.000
posture	32303.453	1	32303.453	2.449	.119
congruency	576821.635	1	576821.635	43.734	.000
posture * congruency	6560.339	1	6560.339	.497	.481
Error	2585080.21	196	13189.185		
Total	154109741	200			
Corrected Total	3200765.64	199			

a. R Squared = .192 (Adjusted R Squared = .180)

Profile Plots



You can see from this output that:

1. There is no main effect of posture, $F(1, 196)=2.449$, $p=NS$
2. There is a main effect of congruency, $F(1, 196)=43.734$, $p<.05$ where (from looking at the plot) incongruent stimuli were processed with a longer re-

action time than congruent stimuli.

3. There is no interaction between posture and congruency, $F(1, 196)=.497$, $p=NS$

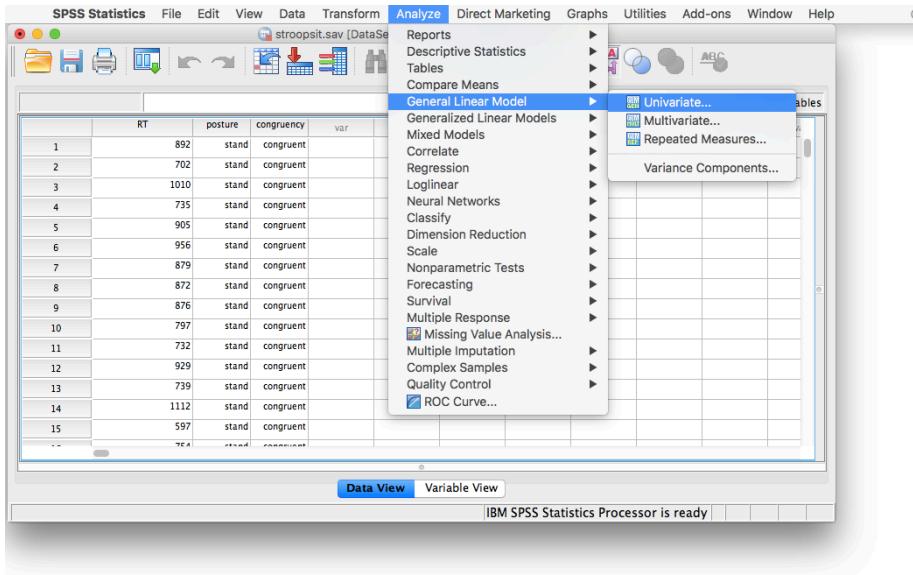
10.6.3 Calculate simple effects

In previous ANOVAs, we have conducted both planned and unplanned comparisons to locate the significant differences. In the case of a two-factor ANOVA, there are many comparisons that can be made. However, the comparisons you choose will depend on the results of your ANOVA. In this case, we found only a significant main effect of congruency. So, we can explore this effect more by asking:

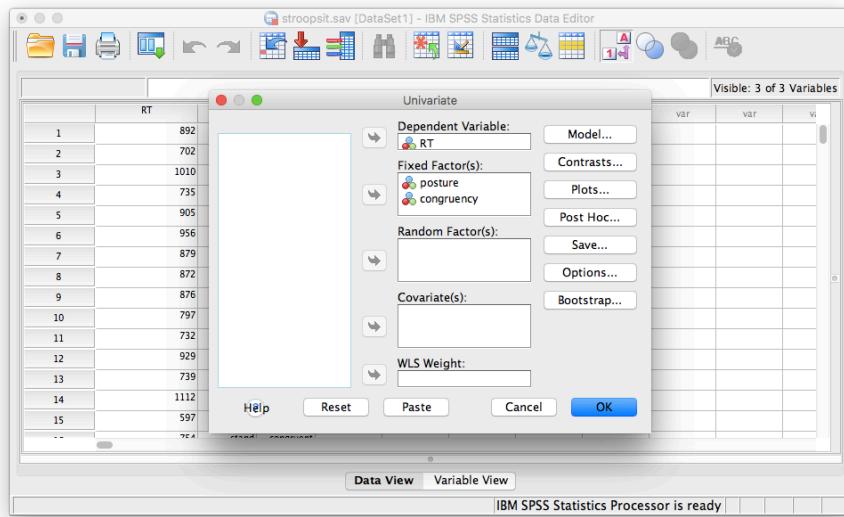
1. For the sitting condition only, are congruent and incongruent means significantly different?
2. For the standing condition only, are congruent and incongruent means significantly different?

To answer these questions, we must calculate something called simple effects. They look at mean differences within levels of a single independent variable. In order to achieve this in SPSS, we have to get into the coding environment that runs the SPSS program; it's called Syntax. Don't freak out here, we're going to make this as simple as possible.

First, let's go to the menu just as we did to run this ANOVA. Analyze, General Linear Model, then Univariate...



The next window looks just as we left it, with the variables in the right places. Here, click on the button that says Paste.



The result will be a new window with code already entered into it. This code corresponds to the ANOVA setup we have specified.

```

UNIANOVA RT BY posture congruency
/METHOD=SSTYPE(3)
/INTERCEPT=INCLUDE
/PLOT=PROFILE(posture*congruency)
/EMMEANS=TABLESOVERALL
/CRITERIA=ALPHA(.05)
/DESIGN=posture congruency posture*congruency.

```

The screenshot shows the Syntax2 - IBM SPSS Statistics Syntax Editor. The syntax window contains the following SPSS command:

```

UNIANOVA RT BY posture congruency
/METHOD=SSTYPE(3)
/INTERCEPT=INCLUDE
/PLOT=PROFILE(posture*congruency)
/EMMEANS=TABLESOVERALL
/CRITERIA=ALPHA(.05)
/DESIGN=posture congruency posture*congruency.

```

The status bar at the bottom indicates 'IBM SPSS Statistics Processor is ready' and 'In 1 Col 0'.

You are going to edit this code so that it looks like this:

```
UNIANOVA RT BY posture congruency /METHOD=SSTYPE(3)
/INTERCEPT=INCLUDE /PLOT=PROFILE(posture * congruency)
/EMMEANS=TABLES(posture * congruency) COM-
PARE(congruency) ADJ(LSD) /CRITERIA=ALPHA(.05) /DE-
SIGN=posture congruency posture * congruency.
```

Once it does, click the big green triangle (“play”) at the top of the window.

SPSS will produce a series of tables. The one that refers to simple effects is called “Pairwise Comparisons.”

Pairwise Comparisons						
Dependent Variable: RT			Mean Difference (I-J)	Std. Error	95% Confidence Interval for Difference ^b	
posture	(I) congruency	(J) congruency			Sig. ^b	Lower Bound
stand	congruent	incongruent	-95.953	22.969	.000	-141.251 -50.655
	incongruent	congruent	95.953	22.969	.000	50.655 141.251
sit	congruent	incongruent	-118.862	22.969	.000	-164.160 -73.565
	incongruent	congruent	118.862	22.969	.000	73.565 164.160

Based on estimated marginal means
 *. The mean difference is significant at the
 b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

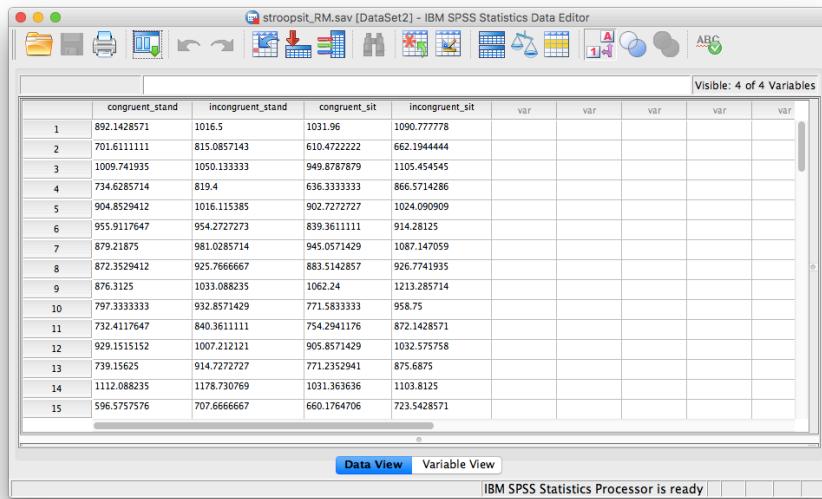
From this table, we can see that,

1. In the sitting condition, the congruent and incongruent conditions are significantly different ($p < .05$).
2. In the standing condition, the congruent and incongruent conditions are significantly different ($p < .05$).

10.6.4 Conduct a Repeated Measures Two-Factor Analysis of Variance (ANOVA)

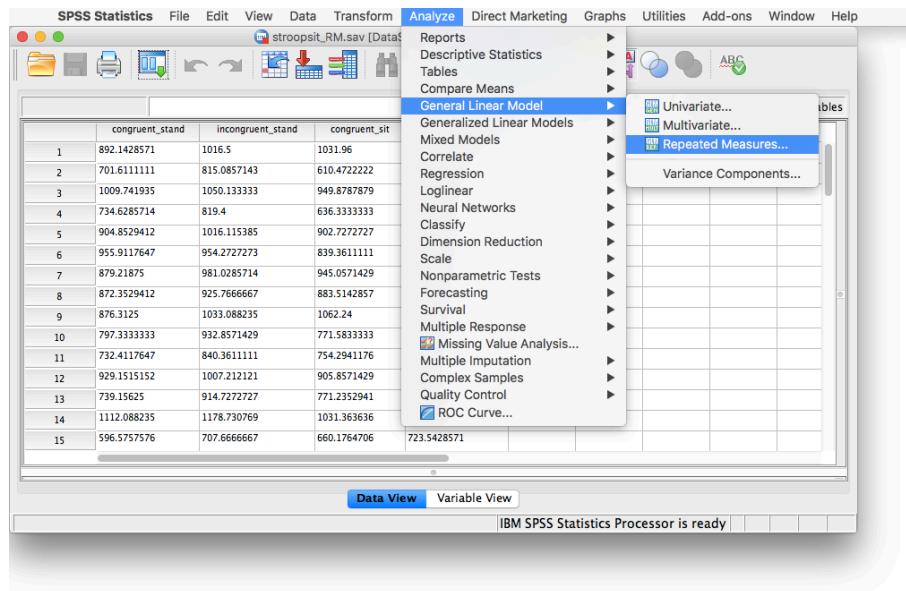
Next, we will use this same data but treat it as a repeated measures (within-subjects) design (just as in the original experiment). This means each person in the study experienced ALL 4 conditions.

To start, we need a new data file. Here is link. This data file is called stroop_RM.sav. I have set it up so that the data is arranged for a repeated-measures design. Notice that each person is represented by a single row, and the columns correspond to the 4 conditions:



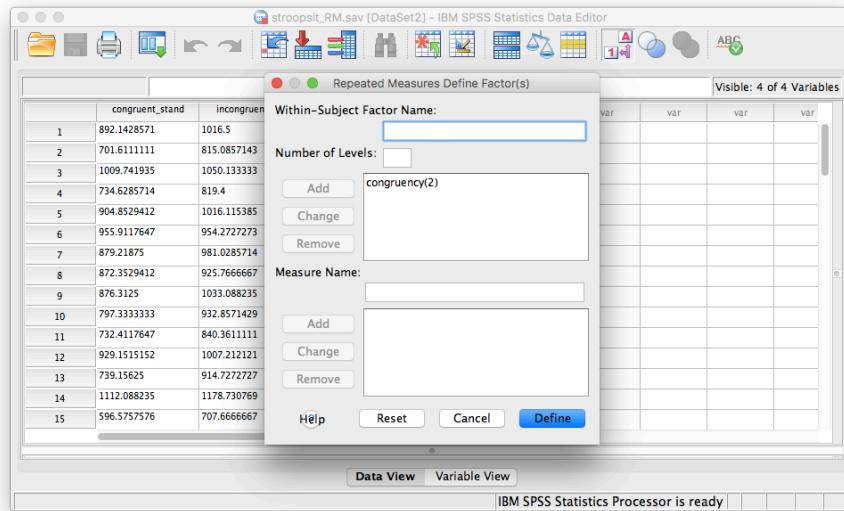
	congruent_stand	incongruent_stand	congruent_sit	incongruent_sit	var	var	var	var
1	892.1428571	1016.5	1031.96	1090.777778				
2	701.6111111	815.0857143	610.4722222	662.1944444				
3	1009.741935	1050.133333	949.8787879	1105.454545				
4	734.6285714	819.4	636.333333	866.5714286				
5	904.8529412	1016.115385	902.7272727	1024.090909				
6	955.9117647	954.2727273	839.3611111	914.28125				
7	879.21875	981.0285714	945.0571429	1087.147059				
8	872.3529412	925.7666667	883.5142857	926.7741935				
9	876.3125	1033.088235	1062.24	1213.285714				
10	797.3333333	932.8571429	771.5833333	958.75				
11	732.4117647	840.3611111	754.2941176	872.1428571				
12	929.1515152	1007.212121	905.8571429	1032.575758				
13	739.15625	914.7272727	771.2352941	875.6875				
14	1112.088235	1178.730769	1031.363636	1103.8125				
15	596.5757576	707.6666667	660.1764706	723.5428571				

Now, to run a repeated measures ANOVA, we go to Analyze, then General Linear Model, then Repeated Measures:

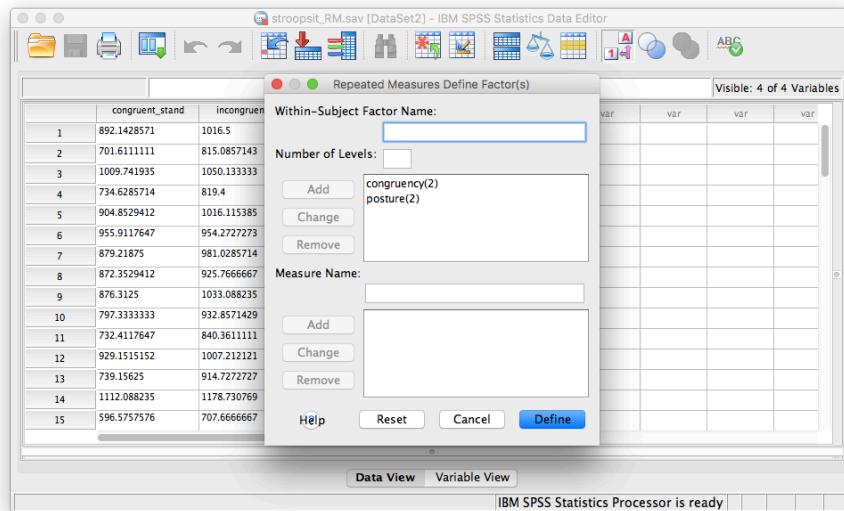


The next window will ask for the name and number of levels of your within-subjects factors (variables); We have two within-subjects factors: **posture** and **congruency**, and each factor has two levels. We must enter them here one at a time. First, change the default **factor1** name to **congruency**, and specify that

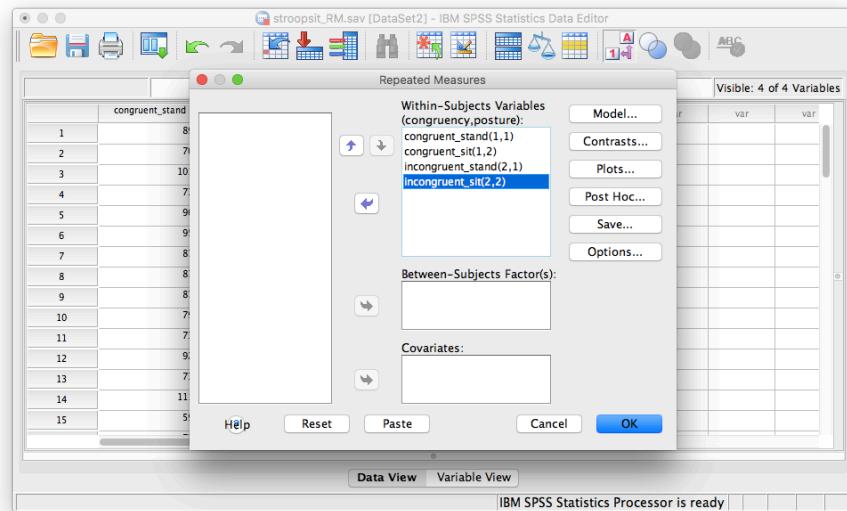
it has 2 levels:



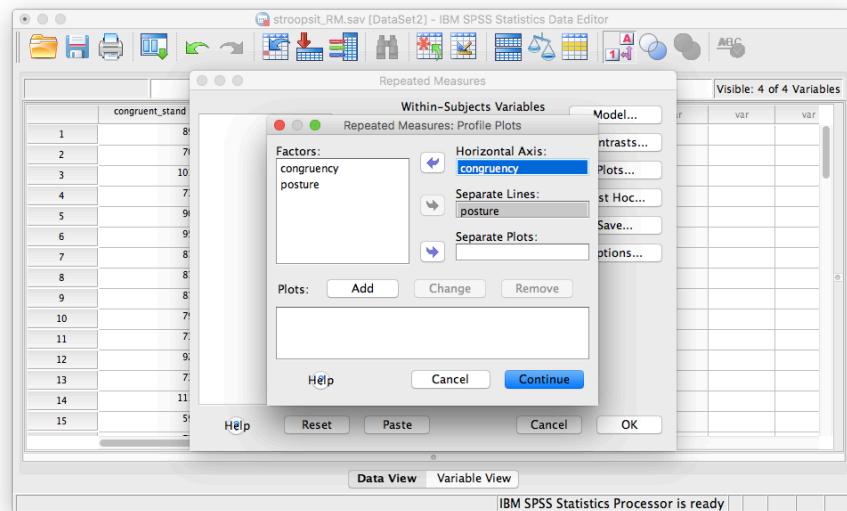
Click Add, and then enter a new within-subjects factor, **posture** with 2 levels as well:



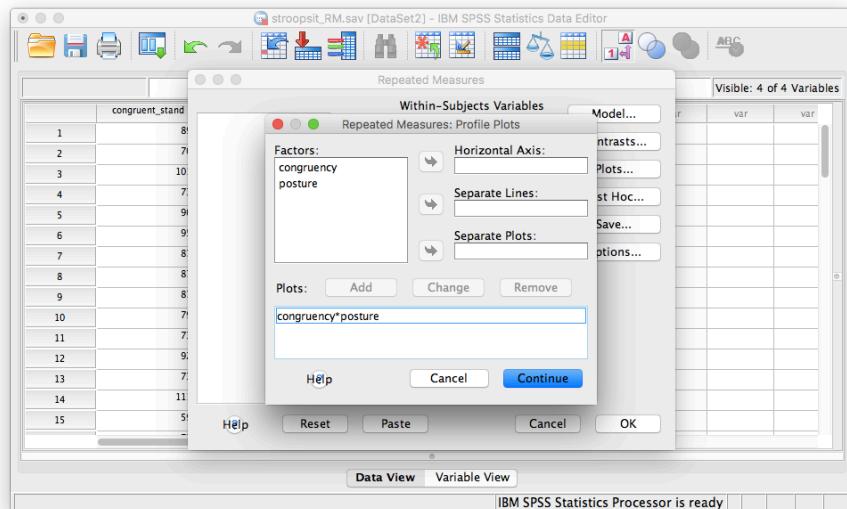
Now, click Add, and then Define. The next window asks for the 4 conditions produced by our two independent variables. Because we entered **posture** first, and **congruency** next, the four conditions, in THIS EXACT order, are entered as follows:



Click Plots and specify a graph exactly as we did in the previous example. Place **posture** in the “Separate Lines” field and **congruency** in the “Horizontal axis” field.



Remember, you must click Add to process this graph!



Now, click Continue, then OK.

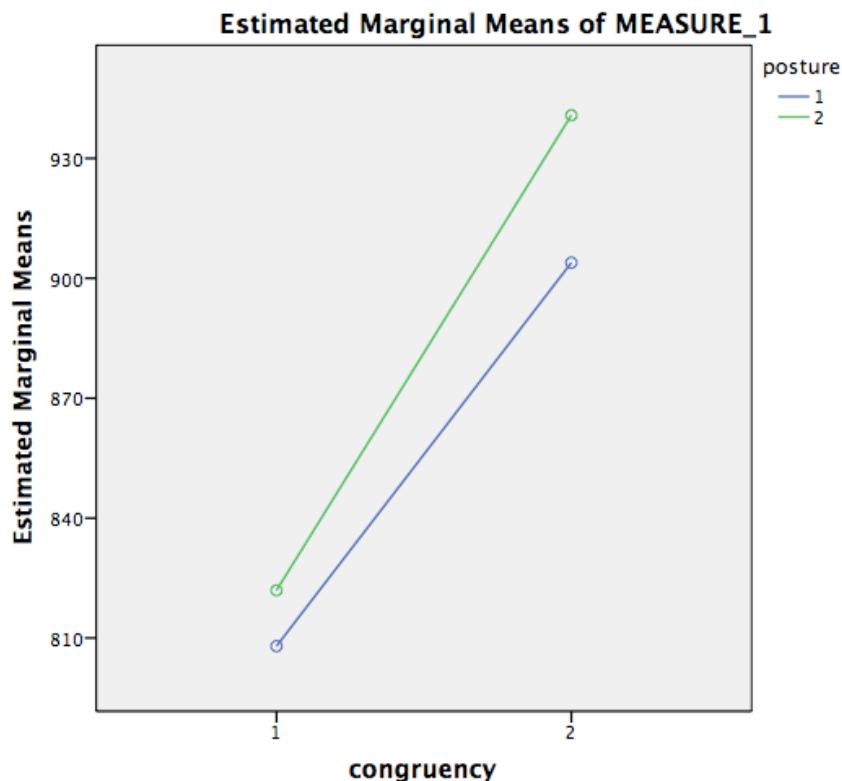
SPSS will produce several tables of output, along with a graph. The Fs for our ANOVA are located in the table labeled “Tests of Within-Subjects Effects”, and the plot we requested is found below:

Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
congruency	Sphericity Assumed	576821.635	1	576821.635	342.452	.000
	Greenhouse-Geisser	576821.635	1.000	576821.635	342.452	.000
	Huynh-Feldt	576821.635	1.000	576821.635	342.452	.000
	Lower-bound	576821.635	1.000	576821.635	342.452	.000
Error(congruency)	Sphericity Assumed	82534.895	49	1684.386		
	Greenhouse-Geisser	82534.895	49.000	1684.386		
	Huynh-Feldt	82534.895	49.000	1684.386		
	Lower-bound	82534.895	49.000	1684.386		
posture	Sphericity Assumed	32303.453	1	32303.453	7.330	.009
	Greenhouse-Geisser	32303.453	1.000	32303.453	7.330	.009
	Huynh-Feldt	32303.453	1.000	32303.453	7.330	.009
	Lower-bound	32303.453	1.000	32303.453	7.330	.009
Error(posture)	Sphericity Assumed	215947.614	49	4407.094		
	Greenhouse-Geisser	215947.614	49.000	4407.094		
	Huynh-Feldt	215947.614	49.000	4407.094		
	Lower-bound	215947.614	49.000	4407.094		
congruency * posture	Sphericity Assumed	6560.339	1	6560.339	8.964	.004
	Greenhouse-Geisser	6560.339	1.000	6560.339	8.964	.004
	Huynh-Feldt	6560.339	1.000	6560.339	8.964	.004
	Lower-bound	6560.339	1.000	6560.339	8.964	.004
Error (congruency*posture)	Sphericity Assumed	35859.069	49	731.818		
	Greenhouse-Geisser	35859.069	49.000	731.818		
	Huynh-Feldt	35859.069	49.000	731.818		
	Lower-bound	35859.069	49.000	731.818		

Profile Plots



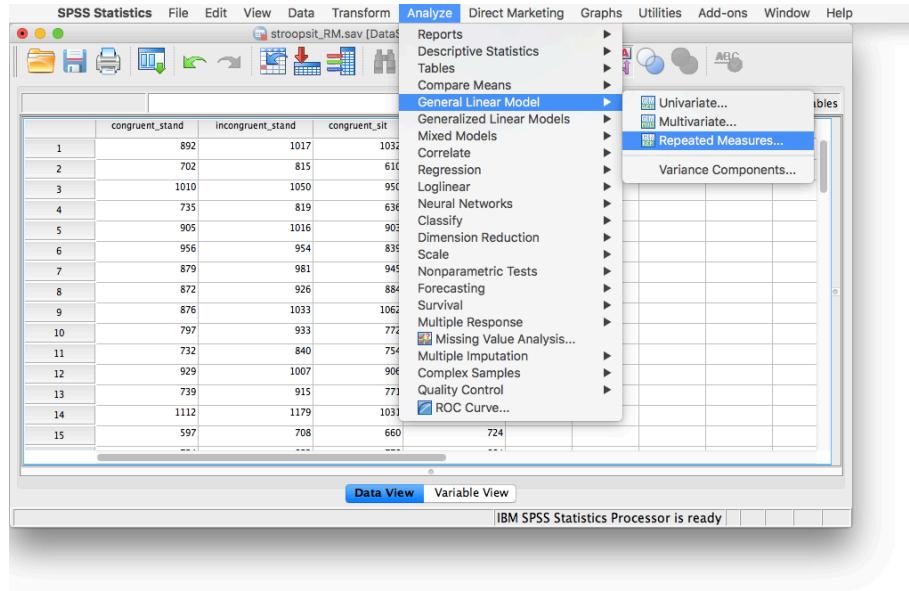
According to this output: 1. There is a main effect of congruency, $F(1, 49)=342.45$, $p<.05$. 2. There is a main effect of posture, $F(1, 49)=7.33$, $p<.05$. 3. There is an interaction effect between congruency and posture, $F(1, 49)=8.96$, $p<.05$.

From the plot we can see that incongruent words produced longer RTs than congruent words. We also see that sitting produced longer RTs than standing. As for the interaction, you can see that the difference between congruent and incongruent is more pronounced (larger) in the sitting condition than in the standing condition.

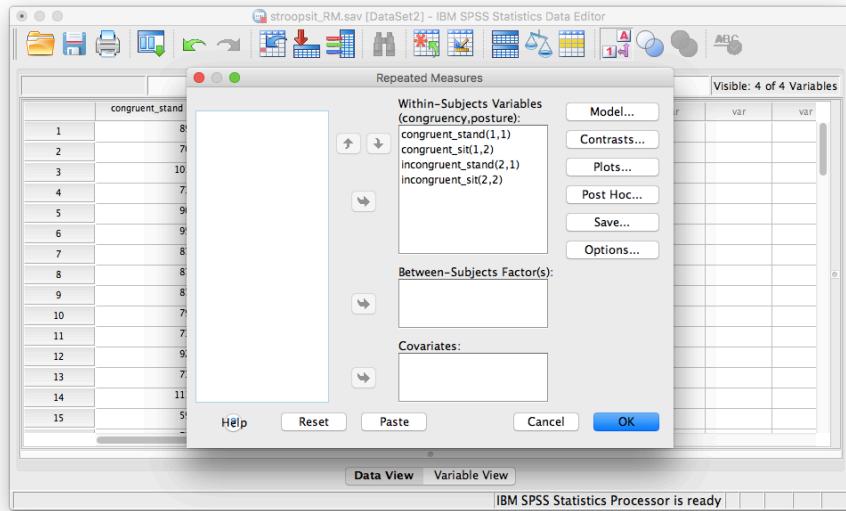
10.6.5 Calculate simple effects

We may want to dissect these effects similarly to the way did in the previous example. This time, because all 3 effects were significant, we will ask SPSS

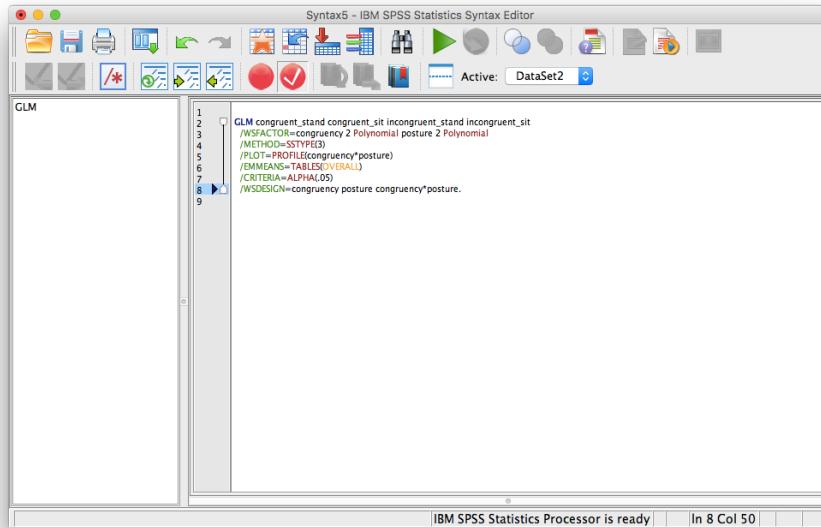
to conduct simple effects for both **congruency** and **posture**. To begin, go to Analyze, General Linear Model, and Repeated Measures...



In the next window, all of your saved settings are still there, so click Define. In the next window, click Paste:



You will be taken to the SPSS Syntax window.



Change the syntax you see to the following:

```
GLM congruent_stand congruent_sit incongruent_stand incongruent_sit
/WSFACTOR=congruency 2 Polynomial posture 2 Polynomial
/METHOD=SSTYPE(3) /PLOT=PROFILE(congruency * posture)
/EMMEANS=TABLES(congruency * posture) compare(congruency) adj(lsd)
/EMMEANS=TABLES(congruency * posture) compare(posture) adj(lsd)
/CRITERIA=ALPHA(.05)
/WSDESIGN=congruency posture congruency * posture.
```

Click the green triangle (“play”) at the top and you will see several output tables appear in the output window. We will focus on the following two:

Pairwise Comparisons

Measure: MEASURE_1

posture	(I) congruency	(J) congruency	Mean Difference (I-J)	Std. Error	Sig. ^b	95% Confidence Interval for Difference ^b	
						Lower Bound	Upper Bound
1	1	2	-95.953*	6.697	.000	-109.412	-82.495
	2	1	95.953*			82.495	109.412
2	1	2	-118.862*	7.197	.000	-133.325	-104.400
	2	1	118.862*			104.400	133.325

Based on estimated marginal means

*. The mean difference is significant at the

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Pairwise Comparisons

Measure: MEASURE_1

congruency	(I) posture	(J) posture	Mean Difference (I-J)	Std. Error	Sig. ^b	95% Confidence Interval for Difference ^b	
						Lower Bound	Upper Bound
1	1	2	-13.963	9.856	.163	-33.770	5.843
	2	1	13.963			-5.843	33.770
2	1	2	-36.872*	10.412	.001	-57.797	-15.948
	2	1	36.872*			15.948	57.797

Based on estimated marginal means

*. The mean difference is significant at the

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

When reading these tables, remember that:

1. For posture, 1=stand and 2=sit
2. For congruency, 1=congruent and 2=incongruent.

In this output, we can see that: 1. For standing only, congruent and incongruent RTs are significantly different ($p < .05$). 2. For sitting only, congruent and incongruent RTs are significantly different ($p < .05$). 3. For congruent words only, standing and sitting RTs are not significantly different ($p = \text{NS}$). 4. For incongruent words only, standing and sitting are significantly different ($p < .05$).

10.6.6 Practice Problems

Below is fictitious data representing the number of millimeters a plant has grown under several water/sunlight combinations:

Water & Sunlight	NoWater & Sunlight	Water & NoSunlight	NoWater & NoSunlight
1.2	2.4	3.1	2.5
3.0	1.1	2.2	3.4
2.5	1.2	2.5	4.2

Water & Sunlight	NoWater & Sunlight	Water & NoSunlight	NoWater & NoSunlight
1.6	2.4	4.3	2.1

1. Enter this data into SPSS as appropriate for a Two-Factor Between-Subjects ANOVA (N=16). Perform the ANOVA and report all results in standard statistical reporting format (use alpha=.05). Include a plot of means.
2. Enter this data into SPSS as appropriate for a Two-Factor Repeated-Measures ANOVA (N=4). Perform the ANOVA and report all results in standard statistical reporting format (use alpha=.05). Include a plot of means.

10.7 JAMOVI

How to do it in JAMOVI

Chapter 11

Lab 11: Mixed Factorial ANOVA

No amount of experimentation can ever prove me right; a single experiment can prove me wrong —Albert Einstein

11.1 Do you remember things better when you take pictures of them?

People take pictures of things all the time on their phones. Barasch et al. (2017) asked whether taking pictures of things had consequences for later memory of those experiences.

11.1.1 Study description

In Experiment 1, participants visited a museum exhibit. Half of the participants were allowed to take photographs (with camera, at least 10 pictures) and the other half were not (no camera). They freely looked at anything in the exhibit, and were allowed to take pictures of anything they wanted (if they were in the camera condition). Additionally, while visiting the exhibit, participants listened to audio guides about the things they were looking at.

After participants were done with the exhibit they returned to the sign-in desk. At this point they were given two memory tests for the things they saw and heard in the exhibit. They were given a visual recognition test containing pictures of objects, and were asked to identify which objects they remembered seeing. They were also given an auditory recognition test containing statements that could have been on the audio guide, and they had to identify which ones they had heard before.

This is a 2x2 mixed design. IV 1 was a between-subjects manipulation involving picture-taking (camera vs. no camera). IV2 was a within-subject manipulation of memory test (visual vs. audio). The dependent measure was performance on the memory test.

An overarching question was whether or not participants would have better visual memory for exhibit objects when they took pictures, compared to when they didn't. Additionally, taking pictures or not, may have no influence on memory for the statements in the audio guide.

11.2 Lab Skills Learned

- Conducting a 2x2 mixed design ANOVA

11.3 Important Stuff

- citation: Barasch, A., Diehl, K., Silverman, J., & Zauberman, G. (2017). Photographic memory: The effects of volitional photo taking on memory for visual and auditory aspects of an experience. *Psychological science*, 28(8), 1056-1066.
- Link to .pdf of article
- Data in .csv format

11.4 R

11.4.1 Load the data

Remember that any line with a # makes a comment and the code does not run. Below is how to load the .csv data from the online repository, or from a local file (you need to change the file path to where the local file is, if you downloaded it). The data contains all of the measures and conditions from Experiment 1 in the paper.

```
library(data.table)
#all_data <- fread("https://github.com/CrumpLab/statisticsLab/raw/master/stroop_stand.
all_data <- fread("data/study1_data_photo.csv")
```

11.4.2 Inspect the dataframe

This will give you a big picture of the data frame. Click the button to view it in your browser, then take a look to see what is in it.

```
library(summarytools)
view(dfSummary(all_data))
```

11.4.3 Get the data you need

This data file contains all of the data from Experiment 1 in the paper.

11.4.4 Get the data into the format you want

The data is sort of in long-format, we will see that we need to do some transformation after looking more closely at the independent and dependent variables

11.4.4.1 The independent variables

1. Photo vs. No photo is coded in the `condition` column
2. Memory task (visual vs. audio). There are no column variables describing which memory task people performed. Instead, the percent correct for the visual memory task is coded in the `vpercent` column, and the percent correct for the audio memory task is coded in the `apercent` column. We will need to transform the data, we do that after describing the DV

11.4.4.2 The dependent variable

In a sense there are two dependent variables, `vpercent` and `apercent`. However, we will treat them as a single memory performance variable.

11.4.4.3 Transforming the data

There were 297 participants. You can check that we have data for each participant by looking at the `labID` column.

```
length(unique(all_data$labID))
```

```
## [1] 297
```

There are 297 rows of data, each corresponding to a single subject. Let's build a data frame we can use for analysis

```
subjects <- as.factor(c(all_data$labID, all_data$labID))
memory_accuracy <- c(all_data$vpercent, all_data$apercent)
memory_task<-as.factor(rep(c("visual", "audio"), each=297))
camera <- as.factor(c(all_data$condition, all_data$condition))

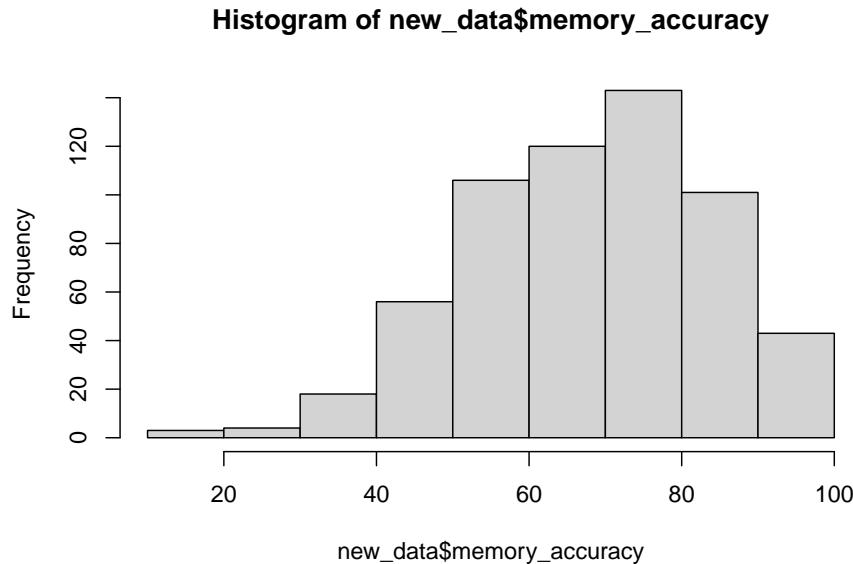
new_data <- data.frame(subjects,
                        memory_accuracy,
                        memory_task,
                        camera)
```

11.4.5 Look at the data

Remember before we do any analysis, we always want to “look” at the data. This first pass let's us know if the data “look right”. For example, the data

file could be messed up and maybe there aren't any numbers there, or maybe the numbers are just too weird. For example, the memory performance DV is a percent, so it should range between 0 and 100, if we find numbers that are negative or greater than 100, then we know something is wrong.

```
hist(new_data$memory_accuracy)
```



```
range(new_data$memory_accuracy)
```

```
## [1] 11.11111 100.00000
```

Great, the numbers check out, they are all inside the range of 0 to 100.

11.4.6 Look at the means

As part of looking at the data, let's graph the means in each condition.

1. Get the means for each condition, by averaging over the means for each subject. These are put into the data frame called `plot_means`.
2. Make a graph with the `plot_means` data frame using `ggplot2`.

```
library(dplyr)
library(ggplot2)

plot_means <- new_data %>%
  group_by(memory_task,camera) %>%
  summarise(mean_accuracy = mean(memory_accuracy),
```

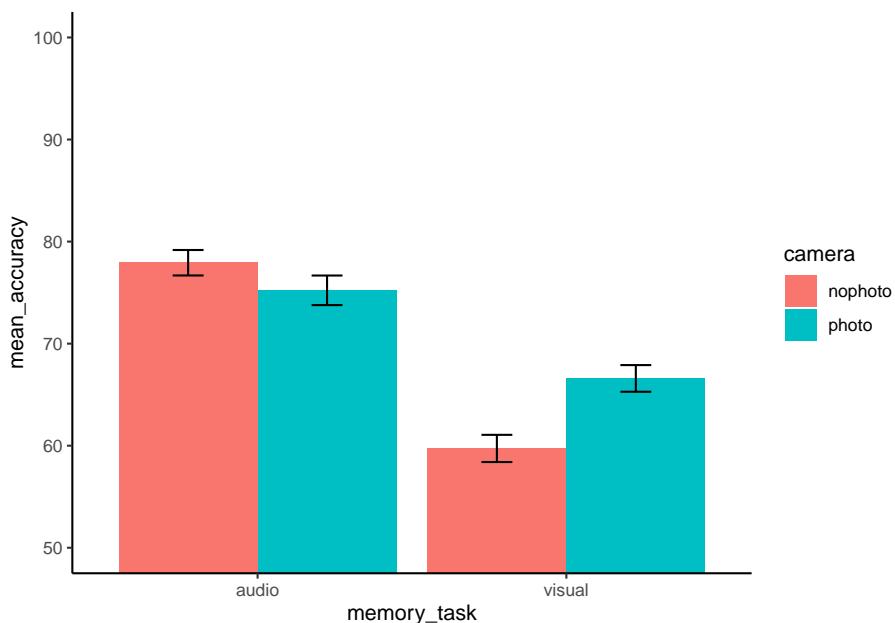
```

SEM = sd(memory_accuracy)/sqrt(length(memory_accuracy))

ggplot(plot_means, aes(x=memory_task, y=mean_accuracy, group=camera, fill=camera))+  

  geom_bar(stat="identity", position="dodge")+
  geom_errorbar(aes(ymin=mean_accuracy-SEM, ymax=mean_accuracy+SEM),
                 position=position_dodge(width=0.9),
                 width=.2)+
  theme_classic()+
  coord_cartesian(ylim=c(50,100))

```



11.4.7 Conduct the ANOVA

This is a 2x2 mixed design. So, we need to run a mixed design ANOVA. The formula for running a mixed design ANOVA is very similar what we have seen before. The only difference is that we specify which IV is within-subjects by placing it in the error term:

```

aov(DV ~ IVB * IVW + Error(Subjects/IVW, data)
library(xtable)

aov_out<-aov(memory_accuracy ~ camera*memory_task + Error(subjects/memory_task), new_data)
summary_out<-summary(aov_out)

knitr::kable(xtable(summary_out))

```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
camera	1	641.9545	641.9545	2.26222	0.1336337
Residuals	295	83712.7032	283.7719	NA	NA
memory_task	1	26786.5902	26786.5902	108.84808	0.0000000
camera:memory_task	1	3394.1862	3394.1862	13.79237	0.0002441
Residuals	295	72597.0014	246.0915	NA	NA

Let's also print out the means for the main effects and interaction:

```
print(model.tables(aov_out,"means"), format="markdown")

## Tables of means
## Grand mean
##
## 69.86532
##
## camera
##   nophoto photo
##   68.83 70.91
## rep 298.00 296.00
##
## memory_task
##   audio visual
##   76.58 63.15
## rep 297.00 297.00
##
## camera:memory_task
##   memory_task
## camera   audio visual
## nophoto 77.93 59.73
## rep     149.00 149.00
## photo    75.23 66.59
## rep     148.00 148.00
```

11.4.8 Generalization Exercise

The generalization exercise is the writing assignment for this lab.

11.4.9 Writing assignment

Your writing assignment is to write up the results of the ANOVA that you just conducted. You can follow the general recipe from the ANOVA write-up from the previous lab on factorial designs. Your write up will be in two parts

(3 points total)

Part 1

1. Say what the numbers were that you are analyzing

2. Say what the statistical test was
3. Say the results of the statistical test for each main effect and interaction
4. For each effect, say what the patterns of the means were.

Part 2

Conduct follow-up t-tests.

- A. Determine whether memory accuracy is better in the camera vs no camera condition for the visual memory task. Report the t-test and the pattern of means.
- B. Determine whether memory accuracy is better in the camera vs. no camera condition for the audio memory task. Report the t-test and the pattern of means.

11.5 Excel

How to do it in Excel

11.6 SPSS

In this lab, we will use SPSS to:

1. Conduct and graph a Mixed-Factorial Analysis of Variance (ANOVA)
2. Calculate simple effects

11.6.1 Experiment Background

People take pictures of things all the time on their phones. Barasch et al. (2017) asked whether taking pictures of things had consequences for later memory of those experiences.

In this experiment participants visited a museum exhibit. Half the participants were able to use a camera to document their visit, while the other half were not, all while listening to an audio tour of the exhibit. At the end of the visit, everyone was given two memory tests (visual and audio) for the content they saw and heard, respectively.

This is a 2x2 Mixed-Factorial design. The individuals in the photo group are different than the individuals in the no photo group (this is our between-subjects variable—it is called `condition`), while the memory `test_type` (audio and visual) is our within-subjects variable since everyone took both types of tests.

11.6.2 Conduct a Mixed-Factorial Analysis of Variance (ANOVA)

Using SPSS, open the data file linked here. It is called “study1_data_photo.sav”. It should look like this:

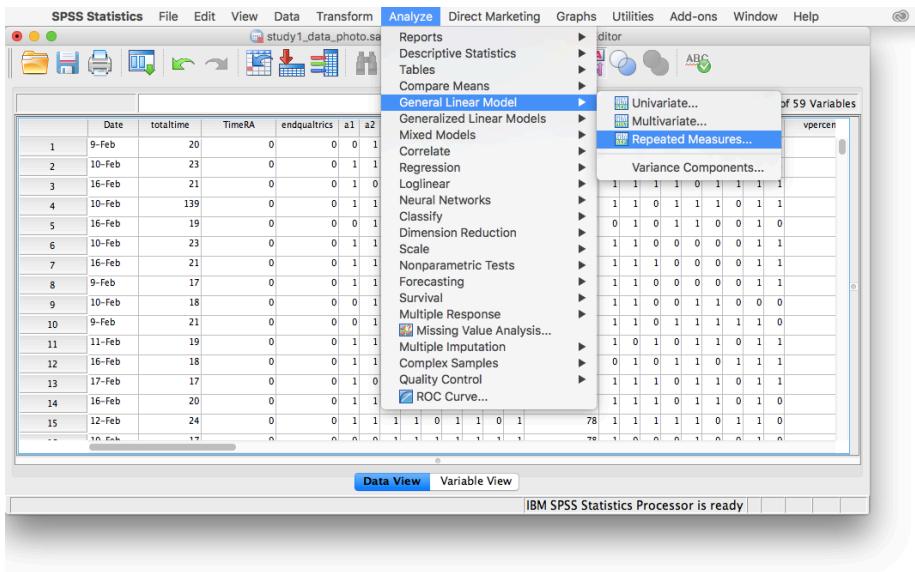
The screenshot shows the SPSS Data Editor window with the following data structure:

	Date	totaltime	TimeRA	endqualtrics	a1	a2	a3	a4	a5	a6	a7	a8	a9	apercent	v1	v2	v3	v4	v5	v6	v7	v8	v9	vpercen
1	9-feb	20	0	0 0 1 1 1 1 0 1 1 1	78	1 0 0 1 0 0 1 1 1																		
2	10-feb	23	0	0 1 1 1 1 1 1 1 1 1	100	1 1 1 1 0 0 1 1 1																		
3	16-feb	21	0	0 0 1 0 0 1 1 1 0 0	56	1 1 1 1 0 1 1 1 1																		
4	10-feb	139	0	0 1 1 1 0 1 0 0 1 0	56	1 1 0 1 1 1 0 1 1																		
5	16-feb	19	0	0 0 1 1 1 1 1 1 1 1	89	0 1 0 1 1 0 0 1 0																		
6	10-feb	23	0	0 1 1 1 1 1 1 1 1 1	100	1 1 0 0 0 0 0 1 1																		
7	16-feb	21	0	0 1 1 1 1 1 1 1 1 1	100	1 1 1 0 0 0 0 0 1 1																		
8	9-feb	17	0	0 1 1 1 0 1 1 1 1 1	89	1 1 0 0 0 0 0 1 1 1																		
9	10-feb	18	0	0 0 1 0 1 1 1 1 1 1	78	1 1 0 0 1 1 0 0 0 0																		
10	9-feb	21	0	0 0 1 0 0 0 1 0 1 1	44	1 1 0 1 1 1 1 1 1 0																		
11	11-feb	19	0	0 1 1 1 0 1 1 1 1 1	89	1 0 1 0 1 1 0 1 1 1																		
12	16-feb	18	0	0 1 1 0 1 1 1 1 1 1	89	0 1 0 1 1 0 1 1 1 1																		
13	17-feb	17	0	0 1 0 1 1 1 1 0 0 0	56	1 1 1 0 1 1 0 1 1 1																		
14	16-feb	20	0	0 1 1 0 1 1 1 0 1 1	78	1 1 1 0 1 1 0 1 1 0																		
15	12-feb	24	0	0 1 1 1 1 0 1 1 0 1	78	1 1 1 1 0 1 1 0 1 0																		
--	16-feb	17	0	0 0 0 0 0 0 0 0 0 0	78	1 0 0 0 0 0 0 0 0 0																		

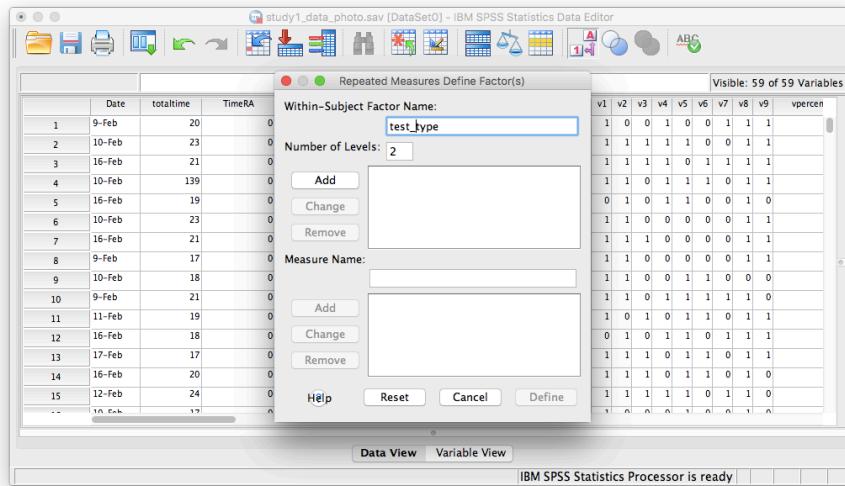
Scroll through the file. Find the variable named **condition**. This variable tells you if the row in question corresponds to a person who was or was not able to use a camera. Now, find the variables **apercent** and **vpercen**. These variables refer to the percentage score received by everyone on the audio and visual memory tests, respectively.

Notice that every row in the data file only has ONE “condition” label. That is, you were either in the camera or no camera condition. BUT, everyone in the file has scores for BOTH tests, audio and visual. This is the way your data must be structured in SPSS in order to perform a mixed-factorial ANOVA. Now, let’s begin.

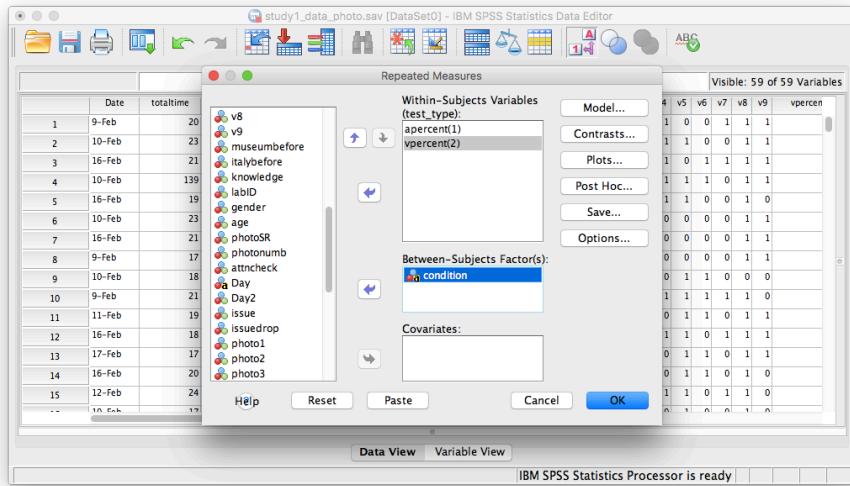
Go to the top menu and choose Analyze, General Linear Model , and Repeated Measures...



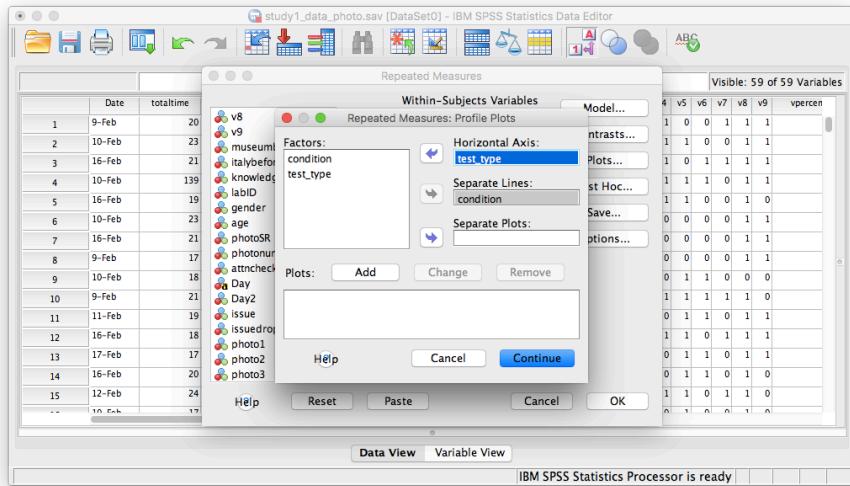
The next window will ask for you to specify the Within-Subjects variable ONLY. Our within-subjects variable (the variable for which participants were measured across all levels) was the memory `test_type`. Enter `test_type` into this field, and specify that there are 2 levels. You must click Add in order to continue:



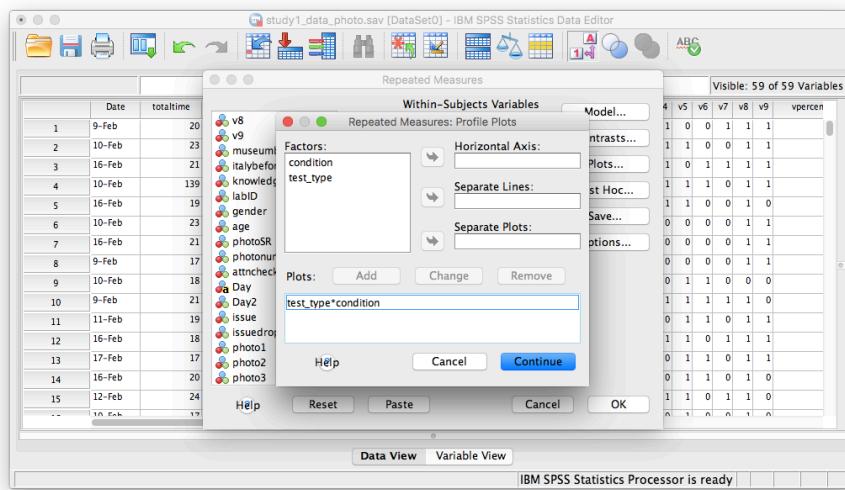
Then click Define. The next window will ask you to specify the levels of your within-subjects variable. Place `apercent` and `vpercent` from the list on the left into the field labeled "Within-Subjects Variables". Then, place `condition` into the field labeled "Between-Subjects variables".



Now, click Plots. Place **condition** in the “Separate Lines” field and **test_type** in the “Horizontal axis” field:



Then click Add. Your window will look like this once you've done that:



Now, click Continue and then OK. SPSS will produce output tables for your ANOVA, as well as a plot:

Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
test_type	Sphericity Assumed	26722.116	1	26722.116	108.586	.000
	Greenhouse-Geisser	26722.116	1.000	26722.116	108.586	.000
	Huynh-Feldt	26722.116	1.000	26722.116	108.586	.000
	Lower-bound	26722.116	1.000	26722.116	108.586	.000
test_type * condition	Sphericity Assumed	3394.186	1	3394.186	13.792	.000
	Greenhouse-Geisser	3394.186	1.000	3394.186	13.792	.000
	Huynh-Feldt	3394.186	1.000	3394.186	13.792	.000
	Lower-bound	3394.186	1.000	3394.186	13.792	.000
Error(test_type)	Sphericity Assumed	72597.001	295	246.092		
	Greenhouse-Geisser	72597.001	295.000	246.092		
	Huynh-Feldt	72597.001	295.000	246.092		
	Lower-bound	72597.001	295.000	246.092		

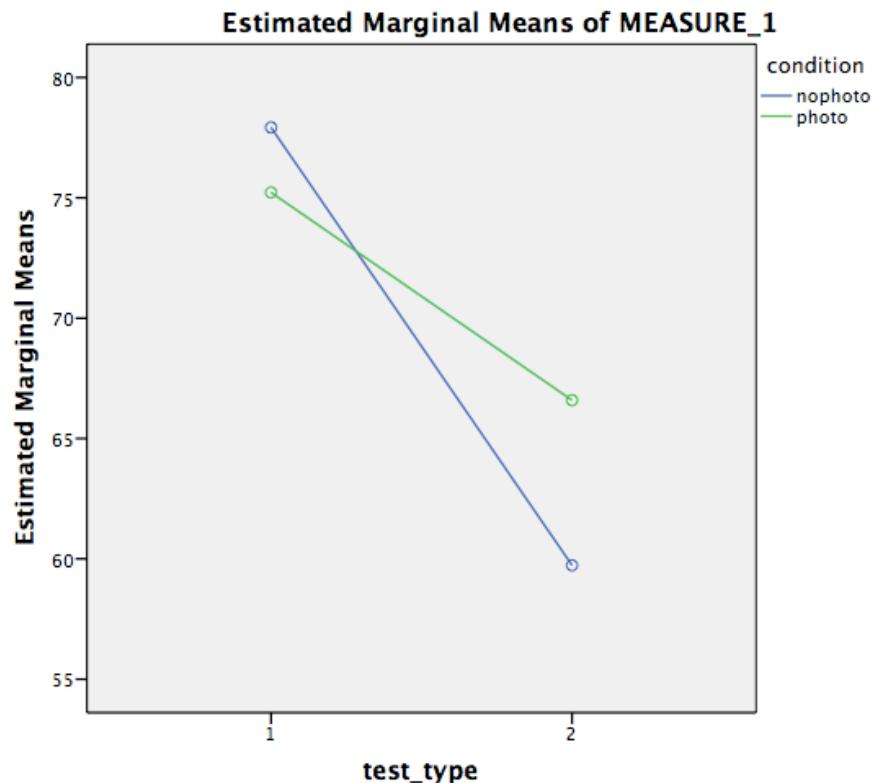
Tests of Between-Subjects Effects

Measure: MEASURE_1

Transformed Variable: Average

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Intercept	2899668.43	1	2899668.43	10218.308	.000
condition	641.954	1	641.954	2.262	.134
Error	83712.703	295	283.772		

Profile Plots



Let's look at our output tables. Based on its values, we can tell that:

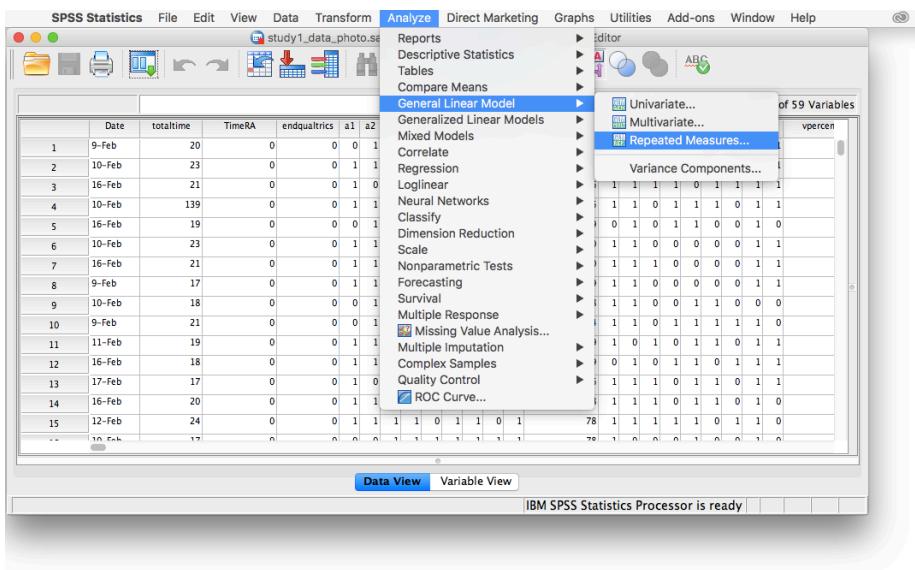
1. There is a significant main effect of test type, $F(1, 295)=108.59$, $p<.05$. Overall, memory test scores were higher for the audio test than the video test.
2. There is a significant interaction effect between test type and condition, $F(1, 295)=13.79$, $p<.05$. People taking photos did worse on the audio test than people without photos, however, on the visual test, people taking photos did better than those who did not take photos.
3. There is no significant main effect of condition, $F(1, 295)=2.26$, $p=NS$.

11.6.3 Calculate simple effects

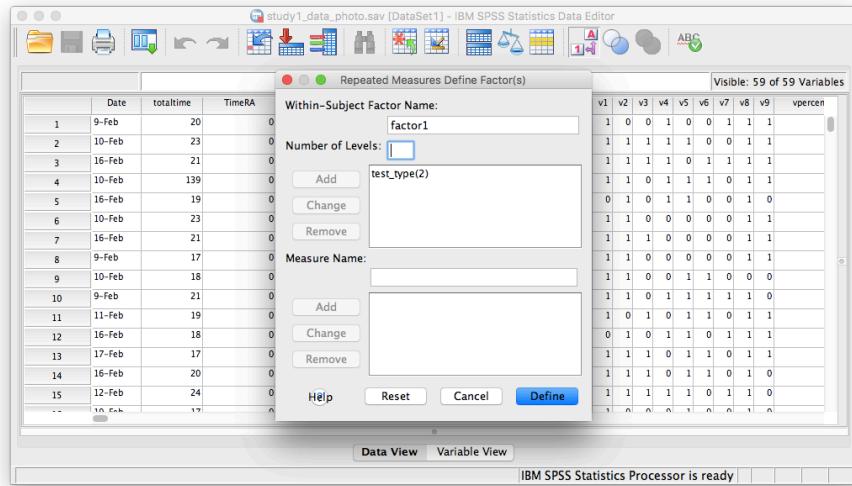
What if we asked the questions: 1. In the no photo condition only, was there

a difference between audio and visual memory score? 2. In the photo condition only, was there a difference between audio and visual memory score? 3. Were audio memory test scores different between photo and no photo conditions? 4. Were visual memory test scores different between photo and no photo conditions?

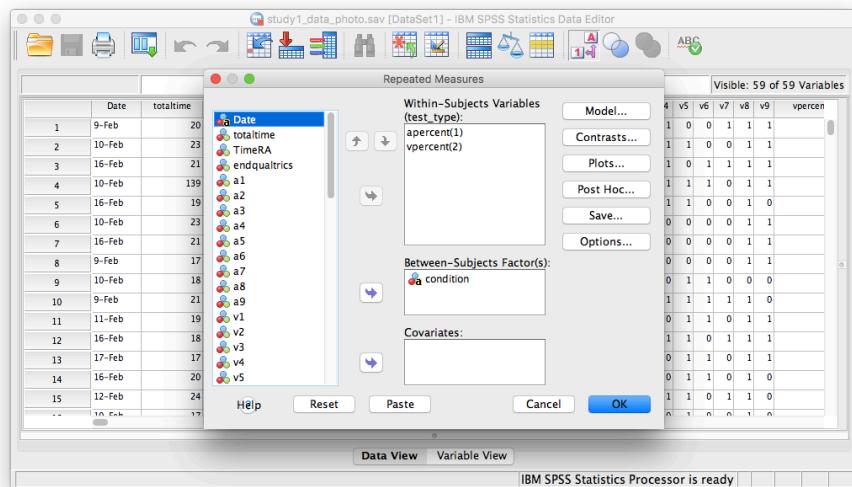
To answer these questions, we must run simple effects. If you remember from earlier labs, we must use SPSS syntax to do this. Go to Analyze, then General Linear Model, then Repeated Measures...



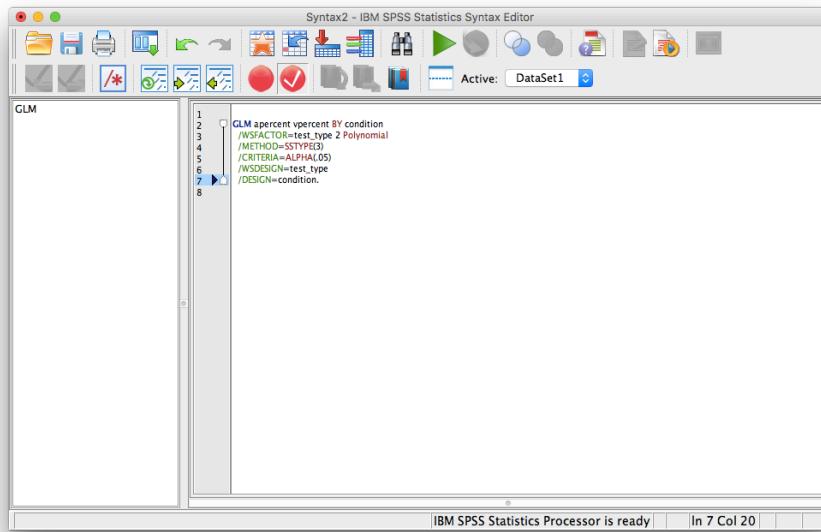
In the next window, you'll find your settings from the previously-run ANOVA have been saved.



Click Define. The following window will appear:



Click Paste and the SPSS Syntax Editor will appear:



Change the code on the screen to the following:

```
GLM apercent vpercent BY condition /WSFACTOR=test_type 2
Polynomial /METHOD=SSTYPE(3) /EMMEANS=TABLES(test_type
* condition) compare(test_type) adj(lsd) /EMMEANS=TABLES(test_type
* condition) compare(condition) adj(lsd) /CRITERIA=ALPHA(.05)
/WSDESIGN=test_type /DESIGN=condition.
```

Click the green triangle ("play") at the top of the window. The following two tables will appear (among others) in your output window:

Pairwise Comparisons

Measure: MEASURE_1

condition	(I) test type	(J) test type	Mean Difference (I-J)	Std. Error	95% Confidence Interval for Difference ^a	
					Sig. ^b	Lower Bound
nophoto	1	2	18.195	1.817	.000	14.619 21.772
	2	1	-18.195*	1.817	.000	-21.772 -14.619
photo	1	2	8.634*	1.824	.000	5.045 12.223
	2	1	-8.634*	1.824	.000	-12.223 -5.045

Based on estimated marginal means

*. The mean difference is significant at the

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Pairwise Comparisons

Measure: MEASURE_1

test_type	(I) condition	(J) condition	Mean Difference (I-J)	Std. Error	95% Confidence Interval for Difference ^a	
					Sig. ^b	Lower Bound
1	nophoto	photo	2.702	1.911	.159	-1.060 6.463
	photo	nophoto	-2.702	1.911	.159	-6.463 1.060
2	nophoto	photo	-6.860*	1.866	.000	-10.533 -3.187
	photo	nophoto	6.860*	1.866	.000	3.187 10.533

Based on estimated marginal means

*. The mean difference is significant at the

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Keep in mind that for `test_type`, 1=audio and 2=visual . According to the table, we can see that:

1. In the no photo condition only, there was a significant difference between audio and visual memory ($p<.05$)?
2. In the photo condition only, there was a significant difference between audio and visual memory ($p<.05$)?
3. Audio memory test scores in the photo condition were not significantly different from those in the no photo condition ($p=NS$).
4. Visual memory test scores in the photo condition were significantly different from those in the no photo condition ($p<.05$).

11.6.4 Practice Problems

1. Using the same data file we used for this lab's tutorial, run a Mixed-Factorial ANOVA using test type (audio and visual) and Gender (Male, Female). Use an alpha level of .05. Report your results in standard statistical reporting format.
2. Produce a plot of means.

11.7 JAMOVI

How to do it in JAMOVI