CPS1102 Fall 2022 SPSS/R Notes & Examples

Michael Carnovale

Contents

Preface **4**
Preface (SPSS-related) . . . 4
Preface (R-related) . . . 4

1 Data Cleaning & Exploration **5**
1.1 SPSS . . . 5
1.1.1 Useful Preliminary Data Cleaning/Prep . . . 5
1.1.2 Variable Descriptives/Plots . . . 6
1.1.3 Missing Data . . . 6
1.2 R . . . 8
1.2.1 Useful Preliminary Data Cleaning/Prep . . . 8
1.2.2 Variable Descriptives/Plots . . . 12
1.2.3 Missing Data . . . 16

2 Correlation **20**
2.1 SPSS . . . 20
2.1.1 Pearson's correlation . . . 20
2.1.2 Spearman's rank correlation . . . 20
2.1.3 Kendall's Tau correlation . . . 20
2.1.4 Point-Biserial correlation . . . 21
2.1.5 Semipartial and partial correlations . . . 21
2.1.6 Bootstrapping correlations . . . 21
2.1.7 Correlation matrices . . . 21
2.1.8 Plotting . . . 22
2.2 R . . . 23
2.2.1 Pearson's correlation . . . 24
2.2.2 Spearman's rank correlation . . . 24
2.2.3 Kendall's Tau correlation . . . 24
2.2.4 Point-Biserial correlation . . . 24
2.2.5 Other robust correlations . . . 25
2.2.6 Semipartial and Partial correlations . . . 26
2.2.7 Bootstrapping correlations . . . 27
2.2.8 Correlation matrices . . . 28
2.2.9 Plotting . . . 28
2.3 Power Analysis Resources . . . 31

3 Regression **33**
3.1 SPSS . . . 33
3.1.1 Simple Linear Regression . . . 33
3.1.2 Multiple Linear Regression . . . 34
3.1.3 Hierarchical regression . . . 34
3.1.4 Robust Regression . . . 34

3.2 R . . . 36
3.2.1 Simple Linear Regression . . . 36
3.2.2 Multiple Linear Regression . . . 46
3.2.3 Hierarchical regression . . . 56
3.2.4 Robust regression . . . 56
3.2.5 Semipartial and Partial correlations . . . 57
3.3 Power Analysis Resources . . . 58

4 t-tests and one-way ANOVA **59**
4.1 SPSS . . . 59
4.1.1 One-sample t-test . . . 59
4.1.2 Independent-samples t-test . . . 59
4.1.3 One-way ANOVA . . . 60
4.1.4 Paired t-test . . . 61
4.2 R . . . 62
4.2.1 One-sample t-test . . . 64
4.2.2 Independent-samples t-test . . . 64
4.2.3 One-way ANOVA . . . 69
4.2.4 Paired t-test . . . 73
4.3 Power Analysis Resources . . . 75

5 ANCOVA **76**
5.1 SPSS . . . 76
5.2 R . . . 76
5.3 Power Analysis Resources . . . 83

6 Factorial Designs and Repeated Measures **84**
6.1 SPSS . . . 84
6.2 R . . . 85
6.2.1 Factorial Designs . . . 85
6.2.2 Repeated-measures ANOVA . . . 91
6.3 Power Analysis Resources . . . 94

7 Mixed Designs **95**
7.1 R . . . 95
7.2 Power Analysis Resources . . . 98

8 Mediation and Moderation **99**
8.1 SPSS . . . 99
8.2 R . . . 99
8.2.1 Moderation . . . 99
8.2.2 Mediation . . . 109
8.3 Power Analysis Resources . . . 110

9 Categorical Outcomes **111**
9.1 SPSS . . . 111
9.2 R . . . 111
9.2.1 Chi-square test of independence . . . 111
9.2.2 Logistic regression . . . 113
9.3 Power Analysis Resources . . . 114

10 Non-Parametric Analyses **115**
10.1 SPSS . . . 115
10.2 R . . . 115
10.2.1 Mann Whitney test . . . 115

10.2.2 Mood's Median test . . . 116
10.2.3 Wilcoxon Signed Rank test . . . 116
10.2.4 McNemar's test . . . 116
10.2.5 Cochran's Q test . . . 117
10.2.6 Kruskal-Wallis test . . . 117
10.2.7 Friedman's test . . . 117
10.3 Power Analysis Resources . . . 118

11 Factor Analysis/Structural Equation Modeling (SEM) 119
11.1 R . . . 119
11.2 Power Analysis Resources . . . 128

12 Multilevel Modeling 129
12.1 R . . . 129
12.2 Power Analysis Resources . . . 137

13 R packages used 138

Preface

This document is meant to serve the following purposes for CPS1102:

- To provide referential and supplemental notes to the material covered in the lecture with respect to SPSS and R,
- To provide the steps in SPSS and R required to run the various statistical methods covered in this class,
- To provide some elaboration regarding the output of results from both SPSS and R, and
- Hopefully probide some insight into data cleaning and exploration in both SPSS and R.

Throughout this document, I make use of *real* datasets from published clinical psychology (or heavily related) papers. These datasets are the same ones that will be used during class demonstrations. I provide an overview of the actual datasets and relevant variables, where needed, along with citations to the papers that the data come from.

Many of the explanations for the statistical methods outlined in this document are informal in nature, and are not meant to be rigorous. Please refer to the appropriate class slides for more information.

As well, any inconsistencies with respect to grammatical tenses are my own doing.

Please let me know if you have any questions regarding any of the material in this document!

Preface (SPSS-related)

In this document, I do not provide the specific output of what is seen in SPSS, but I do provide some explanations of the output in words. In terms of prerequisites:

- Knowledge of how to import datasets into SPSS (if this is not the case, I will demonstrate it in class regardless).

Preface (R-related)

In this document, I provide R code and the output that you would receive if you were to run the code in R with the respective data. The actual R code is highlighted in grey, and the output of the R code usually has two `##` preceding it. In addition to actual written comments outside of the R code, I also make use of comments written *within* the R code. These comments have `#` in front of them and are colour-coded differently from the actual code that you would run. In terms of prerequisites:

- Installation of a program, such as RStudio, to run R itself,
- Knowledge of importing datasets into R (if this is not the case, there is some code throughout the document that shows you how to do this),
- Knowledge of installing packages (this is done using the `install.packages()` function, where the name of the package goes in the parentheses with quotations around its name - e.g., `install.packages("psych")` to install the `psych` package), and
- Knowledge of loading packages once installed (this is done using the `library()` function at the beginning of any R script, where the name of the package goes in the parentheses *without* quotations).

An important thing to note with using R is that external packages sometimes get updates that end up changing the syntax. For example, if you were to try and run any of the code in this document, after freshly installing one of the R packages that are used, it could be the case that the syntax does not work as intended. Therefore, it should be noted that the syntax here is *specific* to the R package versions that I had installed at the time of running this code. See the last page of this document for the package versions that are used in this document.

If you need to install a particular version of an R package, see here: https://search.r-project.org/CRAN/refmans/remotes/html/install_version.html.

1 Data Cleaning & Exploration

In this section, the dataset that will be used comes from an article titled "Reliability, structure, and validity of module I (personality functioning) of the Structured Clinical Interview for the alternative DSM–5 model for personality disorders (SCID-5-AMPD-I)" (link here: https://doi.org/10.1037/per0000576, and data were found here: https://osf.io/bhq94/). The data include an aggregate score from the SCID-5-AMPD-I (representing general personality dysfunction), an aggregate score from the PID-5, total scores from the PHQ-9 and GAD-7, and an aggregate score from the WHODAS 2.0 (representing functional impairment).

1.1 SPSS

1.1.1 Useful Preliminary Data Cleaning/Prep

As mentioned in Lecture 2, it is useful to have a good handle on the following things with respect to data cleaning/prep: subsetting datasets, recoding variables, creating new variables, and making z-scores. Data visualization will be covered in the next section.

Subsetting Datasets In some cases, you might either want to **subset** a dataset depending on some inclusion/exclusion criteria, or you might want to **merge** two datasets if you happened to collect more participants or more variables. Subsetting a dataset can be done on a variable-basis (i.e., removing certain variables) or on a participant-basis (i.e., removing certain participants that meet certain criteria). For most applications, you probably want to subset datasets on a participant-basis.

For example, I would like to filter/subset participants who have a score greater than 2 on the SCID-AMPD - that is, I only want these participants in my dataset:

- `Data` -> `Select Cases`
- Click 'If condition is satisfied', and then 'If' below that
- Drag `scidampd` into the right box, and further type in '>2' using the buttons provided; click `Continue`
- If you would like to maintain the same dataset, keep the `Filter out unselected cases` option endorsed. Otherwise, you can create a new dataset with these selected participants

Creating New Variables and Recoding Here, we are going to create a new *binary* variable in SPSS. In particular, we are going to create a variable representing 'probable depression diagnosis' using scores on the PHQ-9. Scores of 10 or greater indicate probable depression diagnosis. Here, we are going to actually create a *new* variable in SPSS in order to run a point-biserial correlation. As a reminder, the point-biserial correlation is a correlation between a **binary** variable and a continuous variable. In particular, we are going to create a variable representing 'probably depression diagnosis' using scores on the PHQ-9. Research supports good classification accuracy for a depression diagnosis using a score of 10 or greater on the PHQ-9 (see: https://jamanetwork.com/journals/jama/article-abstract/2766865). Below, we create a new variable called 'dep_dx' which takes on a value of 1 if a participant has a score of 10 or greater on phq9, and 0 otherwise.

- Transform -> Recode into Different Variables
- Drag `phq9` into the right box and click `Old and New Values`
- Click `Range, value through HIGHEST` and enter 10 -> under `New Value` type in 1 (meaning phq9 values greater or equal to 10 take on a new response of 1 in the new variable) -> Click `Add`
- Click `Range, LOWEST through value` and enter 9 -> under `New Value` type in 0 -> Click `Add` and `Continue`
- Click the variables in the middle and then type in the new variable name 'dep_dx' -> Click `Change` -> Click `OK`

A similar procedure can be done if you would like to recode individual values of a variable. Instead of using the `Range` options under the `Old and New Values`, you would recode value by value.

Creating Z-scores Creating z-scores is pretty straightforward in SPSS:

- `Analyze` -> `Descriptive Statistics` -> `Descriptives`

- Drag the variables you want to create z-scores for to the right box and also click `Save standardized values as variables`.
- Click `OK` and if you go back to the dataset you should see a new column of z-scores corresponding to your variables of interest.

1.1.2 Variable Descriptives/Plots

When first looking at a dataset, it is helpful to look at various descriptive statistics for the purposes of finding strange values, possible errors in data input, and to generally get a sense of what the different variables look like. Another important thing to keep in mind when looking at a dataset for the first time is to see whether the variable *types* are what they should be.

When loading the SCID-AMPD dataset into SPSS, you will see that the gad7 and phq15 variables are incorrectly labeled as Nominal in the Variable View section of SPSS. To change them to Scale (i.e., a continuous variable), simply click on the variable type for the corresponding variable and switch it to Scale.

Now, let's look at an overview of descriptive statistics for the variables in the dataset. As alluded to above, there are a couple of things that might be worth paying attention to. First, in the case of quantitative variables, you want to see whether the minimum and maximum values of the quantitative variables match with expectations, as this is one way to determine whether there are coding errors or strange variable values. In the case of qualitative variables, it helps to look at every possible category that a variable can take on. Second, with quantitative variables, skew and kurtosis values can give you some insight into how much the variables are close to a normal distribution.

To look at descriptive statistics for continuous variables in SPSS:

- `Analyze` -> `Descriptive Statistics` -> `Descriptives`
- Drag the variables you want to describe to the right box
- Click `Options` and select Kurtosis and Skewness
- Click `Continue` and then `OK`

A similar procedure can be done for categorical variables: instead of clicking `Descriptives`, click `Frequencies`; select the statistics you would like to estimate under the `Options` button; click `Continue` then `OK`.

Histograms are easy to do in SPSS as well and they can be used to check distributions and outliers:

- `Graphs` -> `Chart Builder`
- Click Histogram and drag it into the Chart Preview
- Drag the variable you want to plot onto the x-axis; click `Display normal curve`; click `OK`

1.1.3 Missing Data

To examine missing data (i.e., MCAR tests, t-tests between those with missing/not missing on a given variable) and do imputation at the same time (i.e., EM imputation):

- Analyze -> Missing Value Analysis
- Drag the variables of interest into the correct boxes to the right
- Click `Descriptives`, select the t-tests option, and click `Continue`
- Under Estimation click the EM checkbox
- Then click the EM button, and select `Save completed data` so the imputed dataset is made
- Click `OK`

When this is done, a new dataset should pop up along with the output of the missing data analyses. In the missing data analyses, you will find Little's MCAR test (i.e., if it is significant, this may mean that the data are not MCAR), along with the missing data t-tests. The purpose of the missing data t-tests is to see whether missingness on one variable is systematically (i.e., significantly) related to values on another variable, which is a characteristic of MAR. In the t-test table, the rows represent missingness on a given

variable, and the columns represent values on another variable depending on whether there is missingness on that 'row variable'. If the test statistic (t) is greater than 1.98 for any test, then this may suggest MAR.

1.2 R

Here, we will load the dataset into R explicitly. I copied and pasted the directory from where the actual dataset is located, and in your case you'll have to put in your own directory. On a Mac, you can get some of this info by right-clicking on the file and clicking 'Get Info'.

```
# Here I am loading the dataset into an object called 'df'.
# The 'header = T' tells R that the variable names are
# in the first row of the dataset.
df <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/pdstudy_ampd_validity.csv",
    header = T
)
```

1.2.1 Useful Preliminary Data Cleaning/Prep

As mentioned in Lecture 2, it is useful to have a good handle on the following things with respect to data cleaning/prep: subsetting and merging datasets, recoding variables, creating new variables, dummy coding, and making z-scores. Data visualization will be covered in the next section.

Subsetting and Merging Datasets In some cases, you might either want to **subset** a dataset depending on some inclusion/exclusion criteria, or you might want to **merge** two datasets if you happened to collect more participants or more variables.

Subsetting a dataset can be done on a variable-basis (i.e., removing certain variables) or on a participant-basis (i.e., removing certain participants that meet certain criteria). First, I will subset the SCID AMPD data on a variable-basis such that I will select the scidampd, pid5, and phq9 variables *only*, and I will put this 'new' smaller dataset into an R object called `df_newvariables`. This is all done with the `select()` function, where you first put in the name of the original dataset, and then the names of the variables that you want. The function `str()` allows you to see a preview of the dataset to make sure everything appears as it should.

```
library(tidymodels) # Install if needed
str(df) # Before subsetting
```

```
## 'data.frame':    121 obs. of  10 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
```

```
df_newvariables <- select(df, -scidampd)
str(df_newvariables) # After subsetting
```

```
## 'data.frame':    121 obs. of  9 variables:
##  $ scid5pd: num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr : num  245 226 346 412 NA ...
##  $ ipo    : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd    : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5   : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9   : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7   : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15  : int  14 9 14 7 NA 14 17 9 14 17 ...
```

```
##  $ who    : num  1.42 1.17 2.42 2.42 1.58 ...
```

If you want to remove a single variable from a dataset, you can do `select(df, -phq9)` if you want to, for example, remove the phq9 variable only.

Next, I will subset the SCID AMPD data on a participant-basis such that, for example, I only want participants who have a score of more than 2 on the scidampd. Same idea as before, I save this 'new' dataset into an R object called `df_scid2`, and I use the `filter()` function to select certain participants. The filter function can take various different settings, such as selecting participants that match a certain value or category (using e.g., `dx == "BPD"`), and can take more than one setting at a time (e.g., `dx == "BPD" & scidampd > 2` for both conditions, or instead of `&` can use `|` to denote *or*, like *either* BPD *or* scidampd > 2).

```
df_scid2 <- filter(df, scidampd > 2)
str(df_scid2)
```

```
## 'data.frame':    38 obs. of  10 variables:
##  $ scidampd: num  3.42 2.58 3.08 2.42 2.17 ...
##  $ scid5pd : num  0.4574 0.8723 0.5426 0.0638 0.4894 ...
##  $ lpfssr  : num  346 412 375 283 340 ...
##  $ ipo     : num  2.33 2.3 3.33 1.8 2.33 ...
##  $ opd     : num  2.82 2.39 2.89 1.79 2.39 ...
##  $ pid5    : num  1.39 1.44 1.97 1.01 1.45 1.05 1.47 0.75 1.61 1.33 ...
##  $ phq9    : int  19 13 19 13 19 21 16 12 7 18 ...
##  $ gad7    : int  15 10 14 11 11 18 13 13 7 18 ...
##  $ phq15   : int  14 7 14 9 14 9 15 6 7 16 ...
##  $ who     : num  2.42 2.42 2.58 1.83 3.08 ...
```

```
min(df_scid2$scidampd) # Confirming the lowest scidampd value
```

```
## [1] 2.083333
```

Similar to subsetting a dataset, one can merge a dataset on a variable-basis (i.e., you assessed the same participants on a few extra variables and want to add it to a larger dataset) or on a participant-basis (i.e., you ran a few more participants and want to add them to a larger dataset). For the sake of demonstration, I will split the original SCID AMPD dataset up.

First, though, I will create a participant ID variable, which is usually the case in most datasets.

```
df$case <- seq(1, nrow(df))
```

Now, I will split the dataset up on a variable-basis, and then combine the two new datasets into one again. When merging datasets on a variable-basis, it is required that there is a participant ID variable so R knows which row from the first dataset should be matched with the rows in the second dataset. Here, I will pretend that WHODAS 2.0 scores were collected in a separate dataset (called `df_1`), and the rest of the variables will be in `df_2`. I will then merge the two datasets into `df_merged` using the `merge()` function, which requires you to specify the two datasets you want to merge, and what the identifying variable is ("case").

```
df_1 <- select(df, case, who)
df_2 <- select(df, -who)
str(df_1) # Only case and who variables
```

```
## 'data.frame':    121 obs. of  2 variables:
##  $ case: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ who : num  1.42 1.17 2.42 2.42 1.58 ...
```

```
str(df_2) # Every variable except who
```

```
## 'data.frame':    121 obs. of  10 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
```

```
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ case    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
# Merging the two datasets
df_merged <- merge(df_1, df_2, by = "case")
str(df_merged) # All good
```

```
## 'data.frame':    121 obs. of  11 variables:
##  $ case    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
```

Lastly, I will split the dataset up on a participant-basis, and then combine the two new datasets into one again. When merging datasets on a participant-basis, it is required that the same variables are in both datasets so that R can merge the two datasets seamlessly. To merge two datasets on a participant-basis, you can use the `rbind()` function. Similar to the above example, I'm creating two separate datasets `df_1` and `df_2` and then I will combine in them into a full dataset called `df_merged`.

```
df_1 <- filter(df, case < 100)
df_2 <- filter(df, case >= 100)
str(df_1) # 99 participants
```

```
## 'data.frame':    99 obs. of  11 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
##  $ case    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
str(df_2) # 22 participants
```

```
## 'data.frame':    22 obs. of  11 variables:
##  $ scidampd: num  0.5 0.75 0.917 0.25 1.417 ...
##  $ scid5pd : num  0 0.2128 0.3085 0.0532 0.4574 ...
##  $ lpfssr  : num  158 258 254 178 244 ...
##  $ ipo     : num  1.67 1.7 2 1.67 2 ...
##  $ opd     : num  1.442 1.495 1.832 0.916 1.716 ...
```

```
##  $ pid5    : num  0.99 0.96 0.84 0.38 0.98 0.84 1.13 1.36 0.64 1.06 ...
##  $ phq9    : int  9 5 7 6 9 4 11 13 9 11 ...
##  $ gad7    : int  7 2 5 6 8 2 5 13 4 10 ...
##  $ phq15   : int  6 2 2 11 12 3 12 7 1 15 ...
##  $ who     : num  1 1.42 1.42 1.25 2.08 ...
##  $ case    : int  100 101 102 103 104 105 106 107 108 109 ...
```

```
df_merged <- rbind(df_1, df_2)
str(df_merged) # Now 121 participants
```

```
## 'data.frame':    121 obs. of  11 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
##  $ case    : int  1 2 3 4 5 6 7 8 9 10 ...
```

Creating new variables Here, I will create a new binary variable representing probable depression diagnosis based on PHQ-9 scores. The PHQ-9 can be scored such that scores of 10 or more represent a probable depression diagnosis.

Below, I create a new variable called 'dep_dx' which takes on a response of *1* or *0* depending on a participant's score on the 'phq9' variable. That is, if a participant has a score of 10 or greater on 'phq9', then they will have a response of *1* on 'dep_dx'. If this is not the case, a participant will have a response of *0* on 'dep_dx'. The `mutate` function is something you use to create new variables in R. The function involves specifying the new variable name, specifying the condition for having a *1* or *0* response, and actually typing in the response that you want the variable to have depending on this condition.

```
df <- df %>%
    mutate(dep_dx = ifelse(
        phq9 >= 10, # The condition
        1, # Response if condition is met
        0 # Response otherwise
    ))

table(df$dep_dx) # Good
```

```
##
##  0  1
## 49 59
```

Recoding variables Now let's say I want to recode the dep_dx variable responses to categories rather than simply 0/1. That is, I'd like to change 1 to "Yes" and 0 to "No". This can be done with the `recode()` function as such.

```
df$dep_dx <- recode(df$dep_dx, "1 = 'Yes'; else = 'No'")
```

```
## Warning: Unreplaced values treated as NA as `.x` is not compatible.
## Please specify replacements exhaustively or supply `.default`.
```

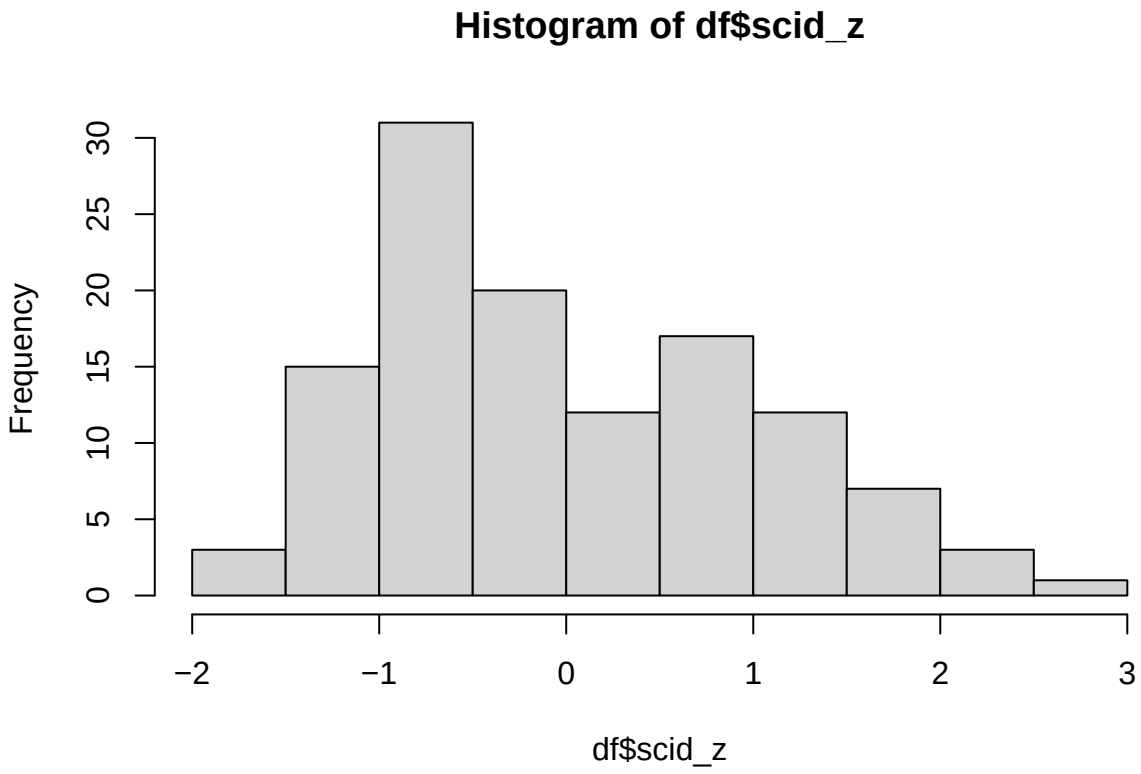
```
table(df$dep_dx)
```

```
##
## 1 = 'Yes'; else = 'No'
##                      59
```

Another common recoding task in research is reverse coding items. This can also be done with the `recode()` function, see here: https://psycnotes.wordpress.com/how-to-recode-in-r/.

Making z-scores Creating z-scores of variables is very straightforward in R. Like with creating a new variable above, you could also use the `mutate` function along with the `scale()` function inside of it, which does the actual z-score calculations. For example, here I want to create a variable representing z-scores on the SCID AMPD, and I will call this new variable `scid_z`.

```
df <- df %>%
    mutate(scid_z = scale(scidampd))
hist(df$scid_z) # Good
```



1.2.2 Variable Descriptives/Plots

When first looking at a dataset, it is helpful to look at various descriptive statistics for the purposes of finding strange values, possible errors in data input, and to generally get a sense of what the different variables look like. Another important thing to keep in mind when looking at a dataset for the first time is to see whether the variable *types* are what they should be. In R, there are a few variable types (see: https://swcarpentry.github.io/r-novice-inflammation/13-supp-data-structures/), where some common ones are `integer` (a variable whose values take on natural numbers like 1, 2, 3, etc.), `numeric` (a variable whose values take on real numbers like 1.23, .56, etc.), and `factor` (a variable whose values take on categories or words like "Yes", "No", etc.). An easy way to look at the variables (and its types) in a loaded dataset can be done with the `str()` function. In the example below, all of the variable types match up with expectations (i.e., they are either numeric or integer because the variables represent mean or total scores). The reason why it is important to see whether the variable types are correct is because certain statistical tests and methods require the variables to be in a certain format.

Below, I'm just going to reload the dataset as if it was from scratch, and then run the `str()` function.

```
df <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/pdstudy_ampd_validity.csv",
    header = T
)
str(df)
```

```
## 'data.frame':    121 obs. of  10 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : int  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
```

Converting between variable types is straightforward in R. Let's say we wanted to change phq9 scores to numeric type, we can use the `as.numeric()` function to overwrite the original variable type. After this is done, you can confirm that the variable type changed by using `str()` again.

```
# Note: df$phq9 means you are referring to that particular variable in the data
df$phq9 <- as.numeric(df$phq9)

str(df) # Indeed changed to numeric
```

```
## 'data.frame':    121 obs. of  10 variables:
##  $ scidampd: num  0.8333 0.0833 3.4167 2.5833 0.0833 ...
##  $ scid5pd : num  0.191 0.298 0.457 0.872 0 ...
##  $ lpfssr  : num  245 226 346 412 NA ...
##  $ ipo     : num  1.77 1.47 2.33 2.3 1.47 ...
##  $ opd     : num  1.48 1.42 2.82 2.39 1.22 ...
##  $ pid5    : num  0.68 0.95 1.39 1.44 0.63 1.97 1.17 1.01 1.45 1.25 ...
##  $ phq9    : num  8 9 19 13 NA 19 8 13 19 11 ...
##  $ gad7    : int  10 9 15 10 NA 14 8 11 11 10 ...
##  $ phq15   : int  14 9 14 7 NA 14 17 9 14 17 ...
##  $ who     : num  1.42 1.17 2.42 2.42 1.58 ...
```

Now, let's look at an overview of descriptive statistics for the variables in the dataset. As alluded to above, there are a couple of things that might be worth paying attention to. First, in the case of quantitative variables, you want to see whether the minimum and maximum values of the quantitative variables match with expectations, as this is one way to determine whether there are coding errors or strange variable values. In the case of qualitative variables, it helps to look at every possible category that a variable can take on. Second, with quantitative variables, skew and kurtosis values can give you some insight into how much the variables are close to a normal distribution.

```
library(tidymodels) # Install if needed
library(psych) # Install if needed
library(datawizard)
describe(df)
```

```
##          vars   n   mean    sd median trimmed   mad    min    max  range  skew
## scidampd    1 121   1.48  0.93   1.25    1.42  0.99   0.00   4.00   4.00  0.55
## scid5pd     2 116   0.30  0.21   0.28    0.28  0.25   0.00   0.87   0.87  0.60
## lpfssr      3 107 255.87 67.65 241.50  250.91 60.05 143.00 470.50 327.50  0.72
## ipo         4 113   2.00  0.43   1.93    1.97  0.49   1.00   3.33   2.33  0.50
```

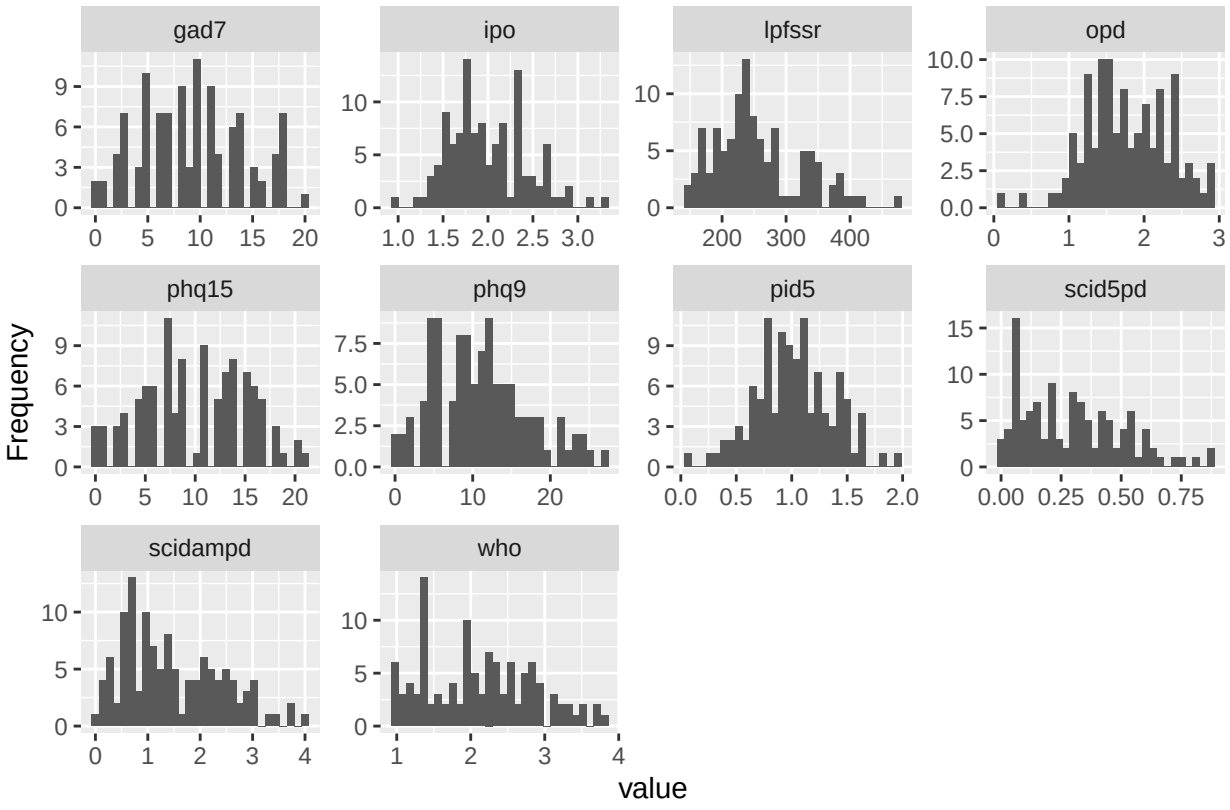
```
## opd         5 111   1.77  0.55   1.76    1.77  0.58   0.14   2.93   2.79 -0.09
## pid5        6 113   1.02  0.35   1.01    1.02  0.36   0.08   1.97   1.89  0.06
## phq9        7 108  11.05  6.01  10.50   10.70  6.67   0.00  27.00  27.00  0.49
## gad7        8 108   9.32  4.90   9.50    9.22  5.19   0.00  20.00  20.00  0.17
## phq15       9 108  10.00  5.23  10.50   10.03  6.67   0.00  21.00  21.00 -0.03
## who        10 113   2.12  0.73   2.08    2.09  0.99   1.00   3.83   2.83  0.28
##          kurtosis   se
## scidampd    -0.57 0.08
## scid5pd     -0.39 0.02
## lpfssr      -0.01 6.54
## ipo         -0.08 0.04
## opd         -0.23 0.05
## pid5        -0.15 0.03
## phq9        -0.32 0.58
## gad7        -0.86 0.47
## phq15       -0.97 0.50
## who         -0.82 0.07
```

```
describe_distribution(df)
```

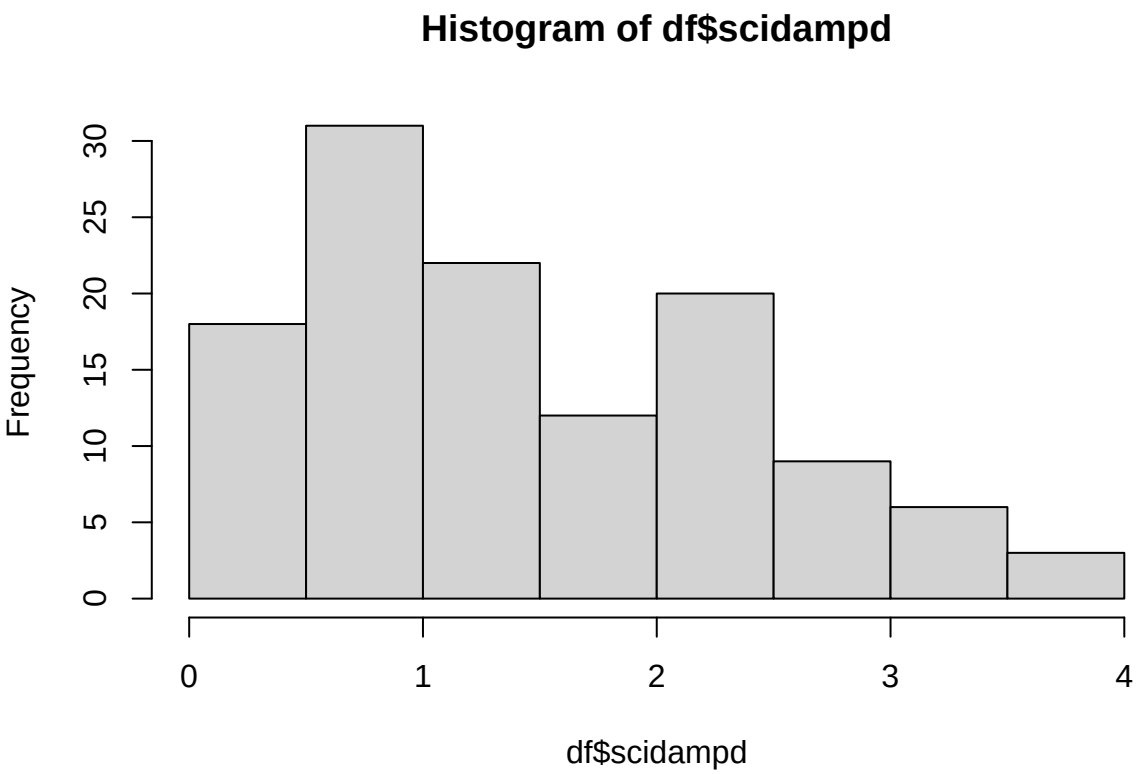
```
## Variable |   Mean |    SD |   IQR |            Range | Skewness | Kurtosis |   n | n_Missing
## ------------------------------------------------------------------------------------------
## scidampd |   1.48 |  0.93 |  1.46 |     [0.00, 4.00] |     0.57 |    -0.50 | 121 |         0
## scid5pd  |   0.30 |  0.21 |  0.34 |     [0.00, 0.87] |     0.61 |    -0.31 | 116 |         5
## lpfssr   | 255.87 | 67.65 | 79.50 | [143.00, 470.50] |     0.74 |     0.11 | 107 |        14
## ipo      |   2.00 |  0.43 |  0.63 |     [1.00, 3.33] |     0.51 |     0.02 | 113 |         8
## opd      |   1.77 |  0.55 |  0.82 |     [0.14, 2.93] |    -0.10 |    -0.13 | 111 |        10
## pid5     |   1.02 |  0.35 |  0.48 |     [0.08, 1.97] |     0.06 |    -0.05 | 113 |         8
## phq9     |  11.05 |  6.01 |  9.00 |    [0.00, 27.00] |     0.50 |    -0.23 | 108 |        13
## gad7     |   9.32 |  4.90 |  8.00 |    [0.00, 20.00] |     0.17 |    -0.80 | 108 |        13
## phq15    |  10.00 |  5.23 |  8.00 |    [0.00, 21.00] |    -0.03 |    -0.92 | 108 |        13
## who      |   2.12 |  0.73 |  1.29 |     [1.00, 3.83] |     0.28 |    -0.76 | 113 |         8
```

If you would like to visualize the distributions using histograms, in order to assess normality and to see whether there are any outliers, you can use the `plot_histogram()` function from the `DataExplorer` package on the *whole dataset*, or you can use the `hist()` function on *individual variables*. You can also visualize the distributions using density plots, with the `densityPlot()` function from the `car` package, which is a smoother-looking visualization of a distribution.

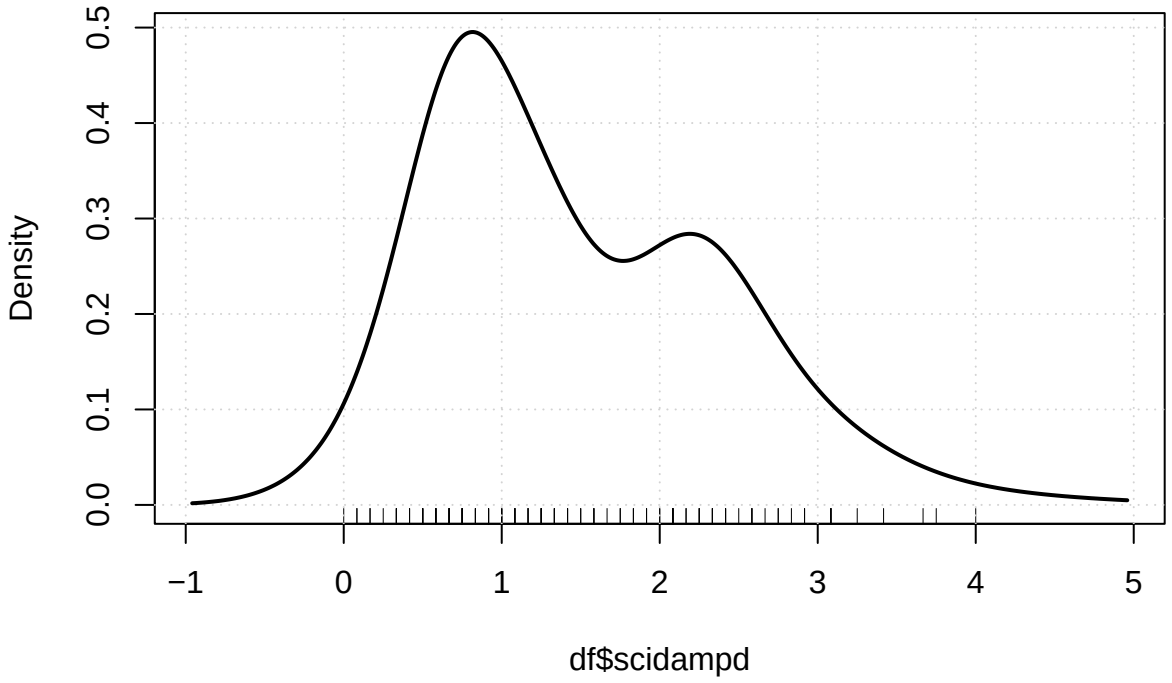
```
library(DataExplorer)
plot_histogram(df) # Every variable
```




```
hist(df$scidampd) # Single variable
```




```
library(car)
densityPlot(df$scidampd)
```



1.2.3 Missing Data

Examining missing data can be done with the `naniar` R package. This package provides descriptives of missingness (using the `miss_var_summary()` function),
provides Little's MCAR test (using the `mcar_test()` function), and provides nice graphs of the missing data (using the `vis_miss()` function).

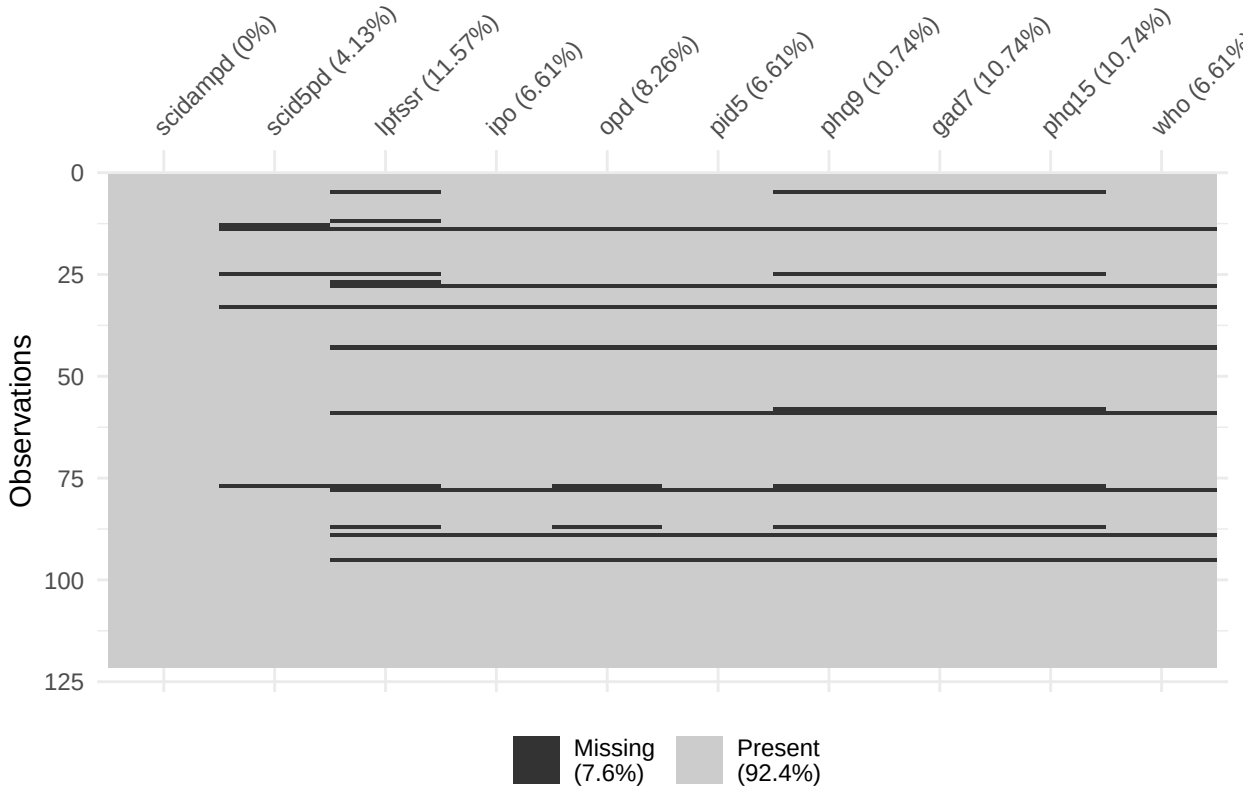
```
library(naniar)
miss_var_summary(df)
```

```
## # A tibble: 10 x 3
##    variable n_miss pct_miss
##    <chr>     <int>    <dbl>
##  1 lpfssr       14    11.6
##  2 phq9         13    10.7
##  3 gad7         13    10.7
##  4 phq15        13    10.7
##  5 opd          10     8.26
##  6 ipo           8     6.61
##  7 pid5          8     6.61
##  8 who           8     6.61
##  9 scid5pd       5     4.13
## 10 scidampd      0     0
```

```
mcar_test(df)
```

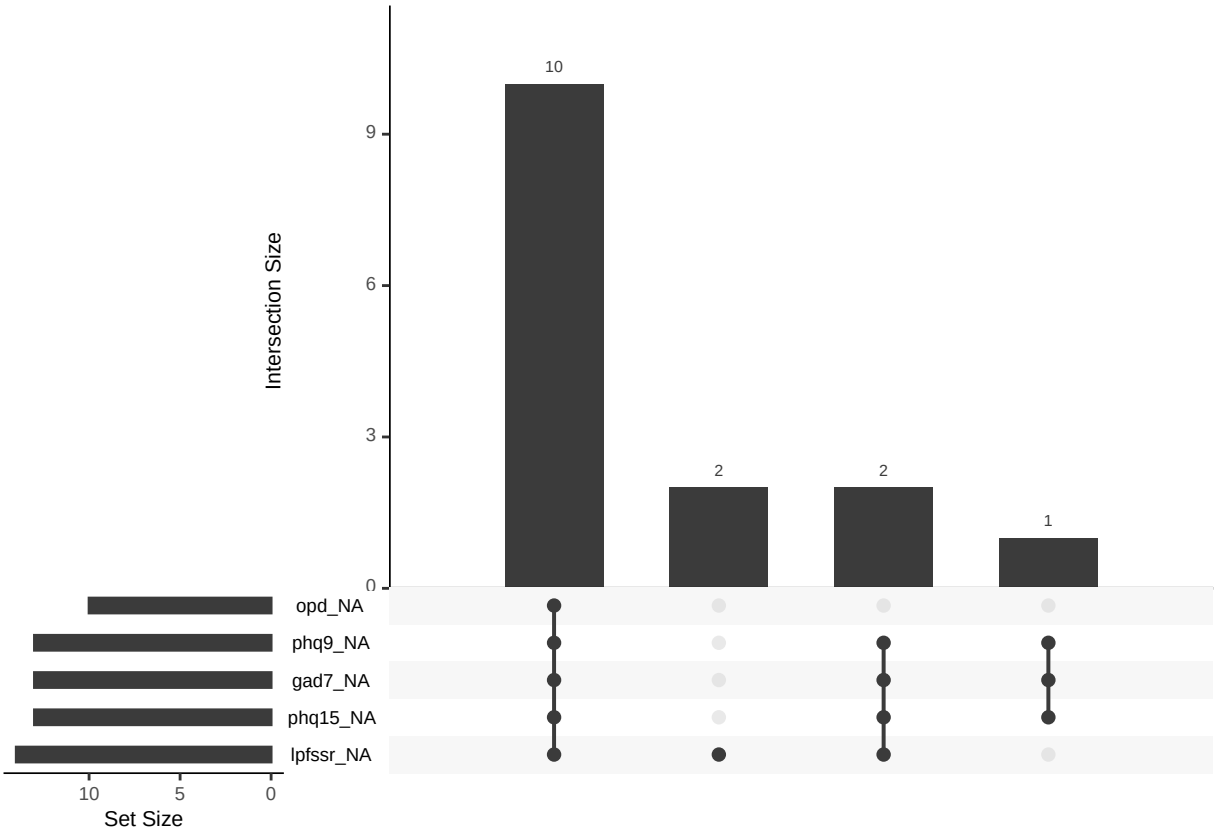
```
## # A tibble: 1 x 4
##   statistic    df p.value missing.patterns
##       <dbl> <dbl>   <dbl>            <int>
## 1      38.2    48   0.845               10
```

```
vis_miss(df)
```



Another useful visualization of missing data patterns is through an Upset Plot. This plot shows you how many participants have missing values on more than one variable, and *which* variables they have missing data on.

```
gg_miss_upset(df)
```



Lastly, single imputation can be done with the `simputation` R package. This package can estimate missing values on any given variable. For example, if you'd like to use EM imputation to fill in missing values for the PHQ-9 variable, the following syntax would be used.

```
library(simputation) # Install if needed
df_new <- impute_em(df, phq9 ~ .)
miss_var_summary(df_new) # Now there are no missing values on phq9
```

```
## # A tibble: 10 x 3
##    variable n_miss pct_miss
##    <chr>     <int>    <dbl>
##  1 lpfssr       14    11.6
##  2 gad7         13    10.7
##  3 phq15        13    10.7
##  4 opd          10     8.26
##  5 ipo           8     6.61
##  6 pid5          8     6.61
##  7 who           8     6.61
##  8 scid5pd       5     4.13
##  9 scidampd      0     0
## 10 phq9          0     0
```

If you'd like to do a full imputation of every variable, use the following syntax.

```
df_new <- impute_em(df, . ~ .)
miss_var_summary(df_new) # Now there are no missing values anywhere
```

```
## # A tibble: 10 x 3
##    variable n_miss pct_miss
```

```
##    <chr>     <int>    <dbl>
##  1 scidampd      0        0
##  2 scid5pd       0        0
##  3 lpfssr        0        0
##  4 ipo           0        0
##  5 opd           0        0
##  6 pid5          0        0
##  7 phq9          0        0
##  8 gad7          0        0
##  9 phq15         0        0
## 10 who           0        0
```

More advanced examples of imputation, such as multiple imputation, can be found in the following book (which was also linked on the Quercus course page): https://stefvanbuuren.name/fimd/

2 Correlation

There are four types of correlation coefficients that we cover in class that can be run in both SPSS and R: **Pearson's correlation**, **Spearman's rank correlation**, **Kendall's Tau correlation**, and **point-biserial correlation**. All of these correlations commonly aim to quantify the degree of an association between two variables, but differ in their underlying calculations, assumptions, and ultimately the research questions they can answer. Semipartial and partial correlations are also covered below. Other robust correlation coefficients can only be run in R to my knowledge, and I will demonstrate them below as well.

In this section, the dataset that will be used comes from an article titled "Reliability, structure, and validity of module I (personality functioning) of the Structured Clinical Interview for the alternative DSM–5 model for personality disorders (SCID-5-AMPD-I)" (link here: https://doi.org/10.1037/per0000576, and data were found here: https://osf.io/bhq94/). The data include an aggregate score from the SCID-5-AMPD-I (representing general personality dysfunction), an aggregate score from the PID-5, total scores from the PHQ-9 and GAD-7, and an aggregate score from the WHODAS 2.0 (representing functional impairment).

2.1 SPSS

2.1.1 Pearson's correlation

Let's take a look at the correlation between general personality dysfunction (variable: scidampd) and functional impairment (variable: who).

- Analyze -> Correlate -> Bivariate
- Select the two variables that you want to correlate
- Click `Confidence interval` -> check off the first box -> click `continue`
- For a cleaner output, on the bottom of the dialogue box click `Show only the lower triangle` and deselect `Show diagonal`
- Click `OK`

In the output, you should get $r = .509$, $p < .001$, 95% CI [.358, .634].

2.1.2 Spearman's rank correlation

Here, we will also take a look at the correlation between the same variables from the Pearson correlation.

- Analyze -> Correlate -> Bivariate
- Select the two variables that you want to correlate
- Select `Spearman` under `Correlation Coefficients`
- Click `Confidence interval` -> check off the first box -> click `continue`
- For a cleaner output, on the bottom of the dialogue box click `Show only the lower triangle` and deselect `Show diagonal`
- Click `OK`

In the output, you should get $r_s = .501$, $p < .001$, 95% CI [.343, .631].

2.1.3 Kendall's Tau correlation

Here, we will also take a look at the correlation between the same variables from the Pearson correlation.

- Analyze -> Correlate -> Bivariate
- Select the two variables that you want to correlate
- Select `Kendall's tau-b` under `Correlation Coefficients`
- Click `Confidence interval` -> check off the first box -> click `continue`
- For a cleaner output, on the bottom of the dialogue box click `Show only the lower triangle` and deselect `Show diagonal`
- Click `OK`

In the output, you should get $r_k = .366$, $p < .001$, 95% CI [.254, .468].

2.1.4 Point-Biserial correlation

Here, we are going to actually create a *new* variable in SPSS in order to run a point-biserial correlation. As a reminder, the point-biserial correlation is a correlation between a **binary** variable and a continuous variable. In particular, we are going to create a variable representing 'probably depression diagnosis' using scores on the PHQ-9. Research supports good classification accuracy for a depression diagnosis using a score of 10 or greater on the PHQ-9 (see: https://jamanetwork.com/journals/jama/article-abstract/2766865). Below, we create a new variable called 'dep_dx' which takes on a value of 1 if a participant has a score of 10 or greater on phq9, and 0 otherwise.

- Transform -> Recode into Different Variables
- Drag `phq9` into the right box
- Click `Range, value through HIGHEST` and enter 10 -> under `New Value` type in 1 (meaning phq9 values greater or equal to 10 take on a new response of 1 in the new variable) -> Click `Add`
- Click `Range, LOWEST through value` and enter 9 -> under `New Value` type in 0 -> Click `Add` and `Continue`
- Click the variables in the middle and then type in the new variable name 'dep_dx' -> Click `Change` -> Click `OK`

Now, to run the actual point-biserial correlation, you would do the same steps as in the Pearson correlation section.

In the output, you should get $r = .581$, $p < .001$, 95% CI [.440, .694].

2.1.5 Semipartial and partial correlations

- Analyze -> Regression -> Linear
- Put in the main dependent variable and independent variable, along with the other independent variables you'd like to control for
- Click `Statistics` and select `Part and partial correlations`
- Click `Continue` and then `OK`

In the output, you should see the semipartial (labeled as part) and partial correlations.

2.1.6 Bootstrapping correlations

To derive bootstrapped confidence intervals for any of the above correlation coefficients:

- Analyze -> Correlate -> Bivariate
- Select the two variables you want to correlate along with the type of correlation
- Click `Bootstrap`, select `Perform bootstrapping`, set the number of bootstrapped samples you want
- Select `Set seed` and put in a random number to ensure reproducibility
- Under CIs, select BCa
- Click `Continue` and then `OK`

Note that bootstrapping can also be done with semipartial and partial correlations.

2.1.7 Correlation matrices

If you would like to estimate pairwise correlations between more than two variables, this information can be presented in a correlation matrix. For example, let's say we want to see the correlations among functional impairment (variable: who), depression (variable: phq9), anxiety (variable gad7), and general personality pathology (variable: pid5).

- Analyze -> Correlate -> Bivariate
- Select all of the variables that you want to correlate
- Select your preferred correlation type under `Correlation Coefficients`
- Click `Confidence interval` -> check off the first box -> click `continue`

- For a cleaner output, on the bottom of the dialogue box click `Show only the lower triangle` and deselect `Show diagonal`
- Click `OK`

2.1.8 Plotting

To plot a scatterplot for the correlation between two variables:

- Graphs -> Chart Builder
- Go to Scatter/Dot, drag the first graph into the chart preview
- Drag the two variables onto the x and y axes
- Near the bottom right of the dialogue box, click `Total` under `Linear Fit Lines`
- Click `OK`

This is almost always a good idea to do *before* you estimate a correlation in order to see whether there are any nonlinear patterns or extreme outliers.

Another interesting graph you can do is called a LOESS plot, which applies a more flexible curve to the scatterplot depending on what the data look like, rather than a strict straight line. Seeing the output of a LOESS plot can help determine whether there are substantial nonlinear relations between two variables. To create a LOESS plot in SPSS:

- Graphs -> Chart Builder
- Go to Scatter/Dot, drag the first graph into the chart preview
- Drag the two variables onto the x and y axes -> Click `OK`
- Double click the resulting graph -> At the top right click `Add Fit Line at Total` -> Select `LOESS`

2.2 R

First, I will show how to load R packages and how to import the dataset into R.

```
library(correlation) # Install if needed
library(performance) # Install if needed
library(skimr) # Install if needed

# Here I am loading the dataset into an object called 'df'.
# The 'header = T' tells R that the variable names are in the first row
df <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/pdstudy_ampd_validity.csv",
    header = T
)

# Basic descriptives and overview of dataset
skim_without_charts(df)
```

Table 1: Data summary

Name	df
Number of rows	121
Number of columns	10
Column type frequency:	
numeric	10
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
scidampd	0	1.00	1.48	0.93	0.00	0.75	1.25	2.17	4.00
scid5pd	5	0.96	0.30	0.21	0.00	0.11	0.28	0.45	0.87
lpfssr	14	0.88	255.87	67.65	143.00	208.50	241.50	286.00	470.50
ipo	8	0.93	2.00	0.43	1.00	1.67	1.93	2.30	3.33
opd	10	0.92	1.77	0.55	0.14	1.42	1.76	2.24	2.93
pid5	8	0.93	1.02	0.35	0.08	0.77	1.01	1.24	1.97
phq9	13	0.89	11.05	6.01	0.00	6.00	10.50	15.00	27.00
gad7	13	0.89	9.32	4.90	0.00	5.00	9.50	13.00	20.00
phq15	13	0.89	10.00	5.23	0.00	6.00	10.50	14.00	21.00
who	8	0.93	2.12	0.73	1.00	1.42	2.08	2.67	3.83

```
# Data preview
head(df)
```

```
##     scidampd   scid5pd lpfssr      ipo      opd pid5 phq9 gad7 phq15      who
## 1 0.83333333 0.1914894  245.0 1.766667 1.484211 0.68    8   10    14 1.416667
## 2 0.08333333 0.2978723  226.5 1.466667 1.421053 0.95    9    9     9 1.166667
## 3 3.41666667 0.4574468  346.5 2.333333 2.821053 1.39   19   15    14 2.416667
## 4 2.58333333 0.8723404  412.5 2.300000 2.389474 1.44   13   10     7 2.416667
## 5 0.08333333 0.0000000      NA 1.466667 1.221053 0.63   NA   NA    NA 1.583333
## 6 3.08333333 0.5425532  375.0 3.333333 2.894737 1.97   19   14    14 2.583333
```

2.2.1 Pearson's correlation

The `cor_test` function takes in the following things and spits out the result: the dataset object ('df'), the two variable names in quotes, and the correlation method in quotes. In this example, we will correlate general personality disorder scores on the SCID-AMPD (variable: scidampd) with functional impairment scores on the WHODAS 2.0 (variable: who). For comparison, the following results are the same as what you get in SPSS.

```
cor_test(df, "scidampd", "who", method = "pearson")
```

```
## Parameter1 | Parameter2 |    r |       95% CI | t(111) |         p
## ------------------------------------------------------------------
## scidampd   |        who | 0.51 | [0.36, 0.63] |   6.24 | < .001***
##
## Observations: 113
```

2.2.2 Spearman's rank correlation

This will also give us the same results like SPSS.

```
cor_test(df, "scidampd", "who", method = "spearman")
```

```
## Parameter1 | Parameter2 |  rho |       95% CI |        S |         p
## ------------------------------------------------------------------------
## scidampd   |        who | 0.50 | [0.34, 0.63] | 1.20e+05 | < .001***
##
## Observations: 113
```

2.2.3 Kendall's Tau correlation

```
cor_test(df, "scidampd", "who", method = "kendall")
```

```
## Parameter1 | Parameter2 |  tau |       95% CI |    z |         p
## ------------------------------------------------------------------
## scidampd   |        who | 0.37 | [0.25, 0.47] | 5.59 | < .001***
##
## Observations: 113
```

2.2.4 Point-Biserial correlation

Here, we are actually going to create a *new* variable within R in order to run a point-biserial correlation. As a reminder, the point-biserial correlation is a correlation between a **binary** variable and a continuous variable. In particular, we are going to create a variable representing 'probable depression diagnosis' using scores on the PHQ-9. Scores greater than or equal to 10 have good classification accuracy for a depression diagnosis (see: https://jamanetwork.com/journals/jama/article-abstract/2766865).

This was demonstrated in the data cleaning section of these notes, but I will repeat the procedure here. Below, I create a new variable called 'dep_dx' which takes on a response of *Yes* or *No* depending on a participant's score on the 'phq9' variable. That is, if a participant has a score of 10 or greater on 'phq9', then they will have a response of *Yes* on 'dep_dx'. If this is not the case, a participant will have a response of *No* on 'dep_dx'. The `mutate` function is something you use to create new variables in R. The function involves specifying the new variable name, specifying the condition for having a *Yes* or *No* response, and actually typing in the response that you want the variable to have depending on this condition. After this function, I change the variable type of 'dep_dx' to be a factor type, which tells R that the variable is categorical rather than a quantitative variable.

```r
# Creating new variable
df <- df %>%
    mutate(dep_dx = ifelse(
        phq9 >= 10, # The condition
        "Yes", # Response if condition is met
        "No" # Response otherwise
    ))

# Changing to categorical variable
df$dep_dx <- as.factor(df$dep_dx)

# Data preview
head(df)
```

```
##     scidampd   scid5pd lpfssr      ipo      opd pid5 phq9 gad7 phq15      who
## 1 0.83333333 0.1914894  245.0 1.766667 1.484211 0.68    8   10    14 1.416667
## 2 0.08333333 0.2978723  226.5 1.466667 1.421053 0.95    9    9     9 1.166667
## 3 3.41666667 0.4574468  346.5 2.333333 2.821053 1.39   19   15    14 2.416667
## 4 2.58333333 0.8723404  412.5 2.300000 2.389474 1.44   13   10     7 2.416667
## 5 0.08333333 0.0000000     NA 1.466667 1.221053 0.63   NA   NA    NA 1.583333
## 6 3.08333333 0.5425532  375.0 3.333333 2.894737 1.97   19   14    14 2.583333
##   dep_dx
## 1     No
## 2     No
## 3    Yes
## 4    Yes
## 5   <NA>
## 6    Yes
```

Now let's actually run the point-biserial correlation. Here, we will correlate depression diagnosis (variable:dep_dx) with functional impairment scores (variable: who). In this R package, setting the method to `pearson` when one of the variables is binary invokes the point-biserial correlation under the hood.

```r
cor_test(df, "dep_dx", "who", method = "pearson")
```

```
## Parameter1 | Parameter2 |    r |       95% CI | t(106) |         p
## ------------------------------------------------------------------
## dep_dx     |        who | 0.58 | [0.44, 0.69] |   7.34 | < .001***
##
## Observations: 108
```

2.2.5 Other robust correlations

Some of the other correlations that I mention in Lecture 3 can be estimated in R as follows.

```r
cor_test(df, "scidampd", "who", method = "biweight") # Biweight midcorrelation
```

```
## Parameter1 | Parameter2 |    r |       95% CI | t(111) |         p
## ------------------------------------------------------------------
## scidampd   |        who | 0.50 | [0.35, 0.62] |   6.25 | < .001***
##
## Observations: 113
```

```r
cor_test(df, "scidampd", "who", method = "percentage") # Percentage bend
```

```
## Parameter1 | Parameter2 |    r |       95% CI | t(111) |         p
## ------------------------------------------------------------------
```

```
## scidampd   |        who | 0.49 | [0.34, 0.62] |   5.95 | < .001***
##
## Observations: 113
```

```
cor_test(df, "scidampd", "who", method = "shepherd") # Shepherd's pi
```

```
## Parameter1 | Parameter2 |  rho |       95% CI |        S |         p
## ------------------------------------------------------------------------
## scidampd   |        who | 0.51 | [0.36, 0.64] | 1.05e+05 | < .001***
##
## Observations: 113
```

```
cor_test(df, "scidampd", "who", method = "distance") # Distance
```

```
## Parameter1 | Parameter2 |    r |       95% CI | t(6214) |         p
## ------------------------------------------------------------------------
## scidampd   |        who | 0.23 | [0.04, 0.39] |   18.22 | < .001***
##
## Observations: 113
```

2.2.6 Semipartial and Partial correlations

With larger regression models, you get a R^2 value that represents the amount of variance explained in the outcome *attributable to the whole set of independent variables*. Although the R^2 value is useful, it does not tell you *which* independent variable explained *what* amount of variance. That is, which independent variable seems to be contributing the most to predicting the outcome?

Semipartial correlations can answer the question of: how correlated is an independent variable with a dependent variable, above and beyond other independent variables?

Partial correlations, on the other hand, are a bit trickier to interpret in my opinion. They can answer the question of: how correlated is an independent variable with a dependent variable, after accounting for the relation between the same dependent variable and other independent variables?

All of this being said, I find that it's useful to square semipartial correlations because that can tell you how much **variance** in a dependent variable is attributable to a given independent variable, above and beyond other independent variables.

The below syntax produces both semipartial correlations (called `part.r` in the output) and partial correlations. In this case, we are interested in the unique relations between functional impairment and three psychopathology variables (depression, anxiety, personality pathology).

Using these results, if we square the semipartial for pid5 (.13^2 = .017), this means that PID-5 scores account for an additional 1.7% of variance in functional impairment scores, above and beyond depression and anxiety scores.

Note that in R it is easier to estimate a regression model first and then compute partial and semipartial correlations from this model. As will be explained in the regression section, the `lm()` function is used to run regression models, where the dependent variable is on the left side of the `~` and the independent variables are on the right side. It is also good practice to 'export' the results of the model into an R object, and here I call that object `model1`.

```
library(jtools) # Install if needed
model1 <- lm(who ~ phq9 + gad7 + pid5, data = df)
summ(model1, part.corr = T)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

	Est.	S.E.	t val.	p	partial.r	part.r
(Intercept)	0.89	0.15	5.89	0.00	NA	NA
phq9	0.07	0.02	3.77	0.00	0.35	0.25
gad7	0.02	0.02	0.73	0.47	0.07	0.05
pid5	0.33	0.16	2.00	0.05	0.19	0.13

Standard errors: OLS

2.2.7 Bootstrapping correlations

Admittedly, bootstrapping is a bit of a process in R. The following code shows you how to bootstrap correlations. Note that this bootstrapping procedure is for Pearson's correlation, and if you'd like to change the type of correlation, you can change the method in the `function` part of the code, under `method`.

```
library(boot)

correlation <- function(data, x, y, indices) {
    df <- df[indices, ]
    out <- df %>%
        cor_test(x, y, method = "pearson") # Change method here
    return(out$r)
}

# Running 1000 bootstraps
results <- boot(
    data = df, x = "who", y = "scidampd", R = 1000,
    statistic = correlation
)

# Calculating the bootstrapped CIs
boot.ci(results)
```

```
## Warning in boot.ci(results): bootstrap variances needed for studentized
## intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.3747,  0.6553 )   ( 0.3858,  0.6637 )
##
## Level     Percentile            BCa
```

```
## 95%   ( 0.3550,  0.6329 )   ( 0.3606,  0.6395 )
## Calculations and Intervals on Original Scale
```

2.2.8 Correlation matrices

If you would like to estimate pairwise correlations between more than two variables, this information can be presented in a correlation matrix. For example, let's say we want to see the correlations among functional impairment (variable: who), depression (variable: phq9), anxiety (variable gad7), and general personality pathology (variable: pid5). Below, I am creating a 'new' dataset (called df_new) that only contains these variables, in order to more efficiently run a correlation matrix on them.

```
# Selecting variables of interest from 'df'
# and making new dataset called 'df_new'
df_new <- select(df, who, phq9, gad7, pid5)
```

The `correlation` function in the `correlation` package (I know, redundant labels) lets you estimate correlation matrices. In comparison to `cor_test` above, you need to put the result into its own object, and here I will call the object `results`. Finally, I use the `summary` function on the `results` object to obtain a correlation matrix.

You can add futher parameters to the `correlation` function, such as the correlation type, and the type of p-value adjustment you want.

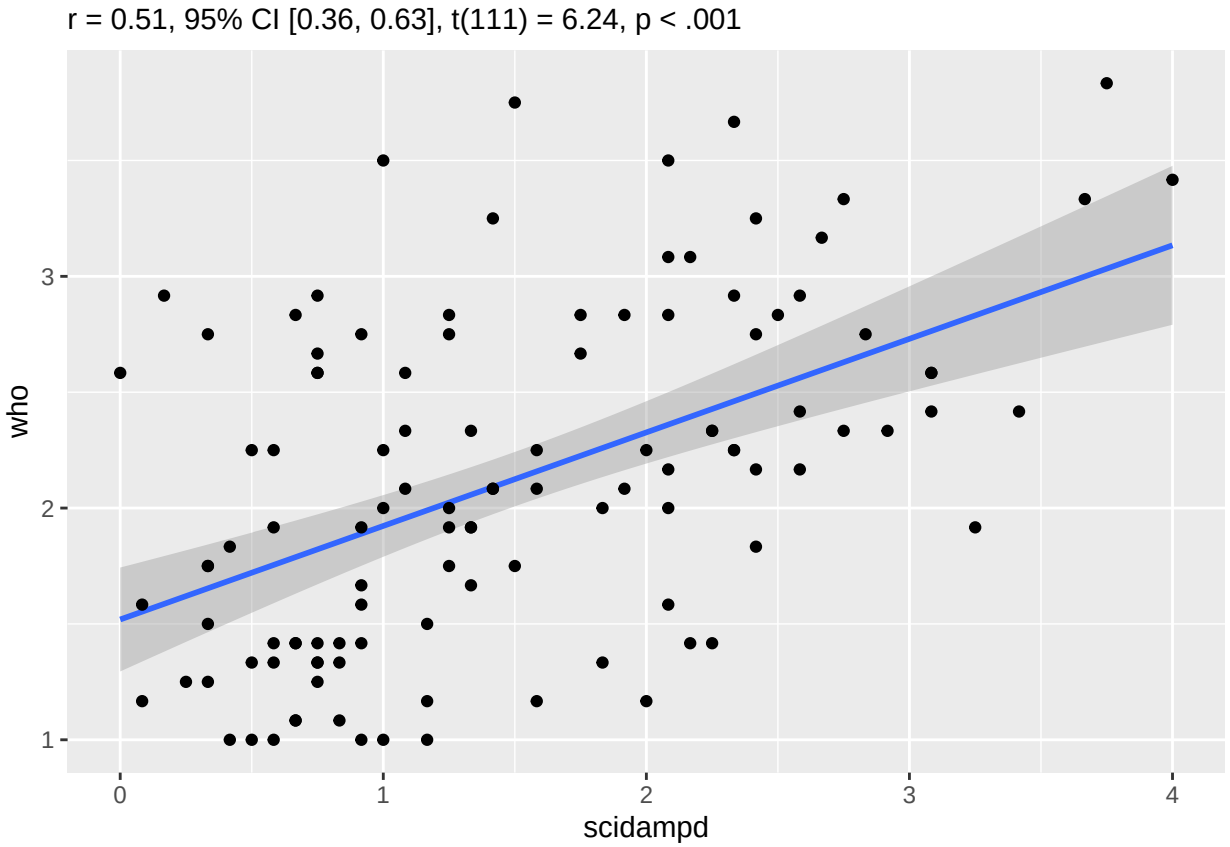
```
results <- correlation::correlation(df_new, method = "pearson")

summary(results)
```

```
## # Correlation Matrix (pearson-method)
##
## Parameter |    pid5 |    gad7 |    phq9
## ----------------------------------------
## who       | 0.51*** | 0.68*** | 0.74***
## phq9      | 0.54*** | 0.89*** |
## gad7      | 0.48*** |         |
##
## p-value adjustment method: Holm (1979)
```

2.2.9 Plotting

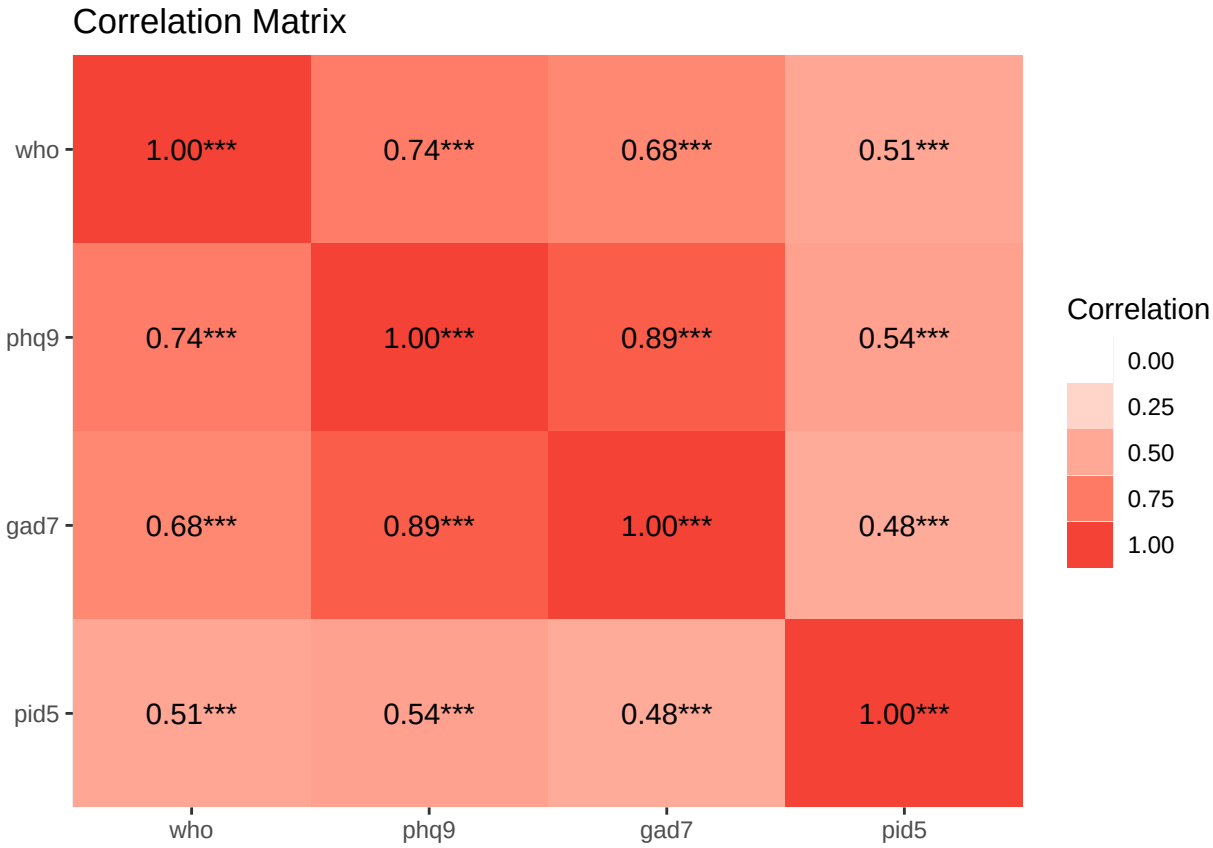
Scatterplots can be done by adding `%>% plot()` after the original `cor_test` function

```
cor_test(df, "scidampd", "who", method = "pearson") %>% plot()
```



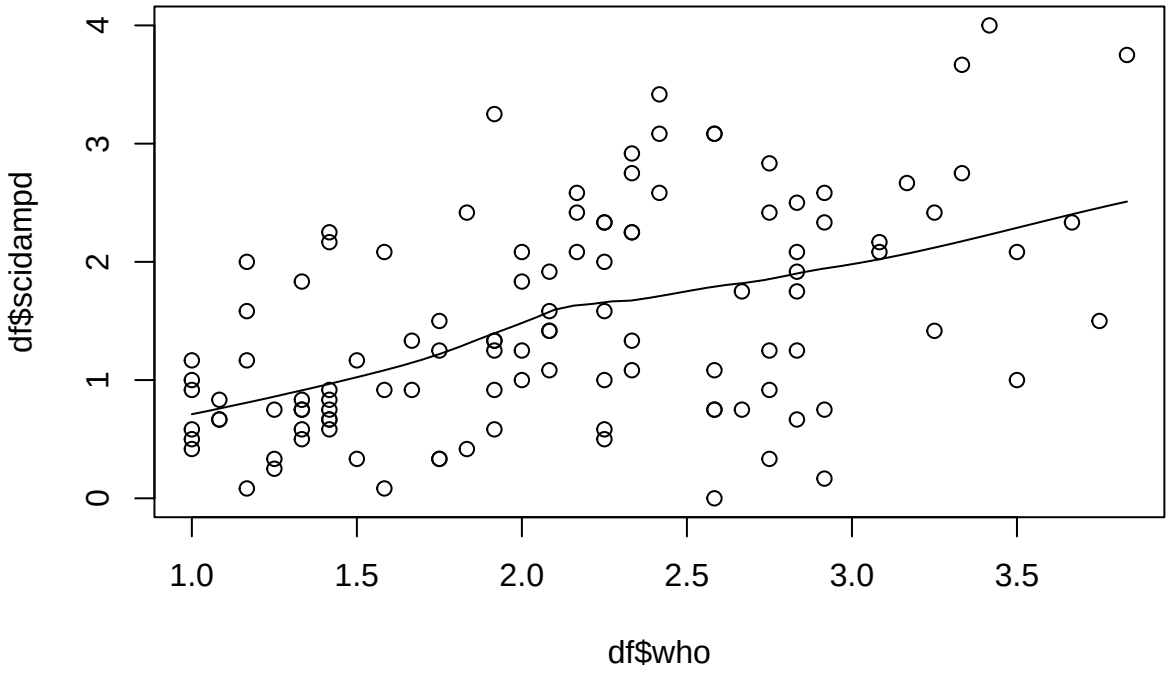
In R, you can also create a visualization of a correlation matrix using the following code and the external R package `see`. This example uses the `results` object created above in the correlation matrix section.

```
library(see)
summary(results, redundant = T) %>% plot()
```



Another interesting graph you can do is called a LOESS plot, which applies a more flexible curve to the scatterplot depending on what the data look like, rather than a strict straight line. Seeing the output of a LOESS plot can help determine whether there are substantial nonlinear relations between two variables.

```
scatter.smooth(df$who, df$scidampd)
```



2.3 Power Analysis Resources

Power analyses for Pearson's correlations can be conducted using the G*Power software, see here (https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower) and here as well (https://link.springer.com/article/10.3758/BRM.41.4.1149). This website provides a manual for conducting power analyses for correlations in this software (among many other statistical tests).* GPower can also do power analyses for *differences* in correlation values.

Another useful website that can be used for correlation power analyses can be found here: https://sample-size.net/correlation-sample-size/. This website lets you input a given alpha value, your Type II error rate (which is 1 - statistical power; e.g., for 80% statistical power, the Type II error rate would be 1 - .80 = .20), and the size of the correlation you would like to detect.

In R, estimating the sample size needed to detect a particular correlation value, with a given amount of statistical power, is straightforward with the help of the `pwr` R package. The `pwr.r.test` function allows you to input a correlation value of interest, a significance (alpha) level of interest, and a degree of statistical power. In the first example below, I would like to determine how many participants I would need in my study if I wanted to detect a true correlation value of .20 with 80% probability (i.e., 80% statistical power) at an alpha level of .05. The output tells me that I would need to collect 194 participants (rounded up). In the second example below, I would like to raise the statistical power to 90%, but also lower the correlation value to .30. The output now tells me that I would need to collect ~112 participants (rounded up).

Another interesting type of power analysis that can be conducted is called a 'sensitivity power analysis'. A sensitivity power analysis tells you the **minimum detectable effect size** given your sample size, a desired amount of statistical power, and an alpha level. A sensitivity power analysis is useful to conduct when you have already collected your data and you would like to see how sensitive your collected sample size is with respect to detecting particular effect sizes. In the third example below, I would like to determine what effect size I can detect with 80% statistical power, given that I collected 100 participants. Rather than putting in the r value that I am interested in detecting, I instead put in my sample size under the `n` option of the function. The output of this result tells me that I could detect correlation values of .28 or above with my sample size (at 80% statistical power). A similar procedure can be conducted using the G*Power software described above.

Note that a sensitivity power analysis is not the same as conducing a post-hoc statistical power analysis, which tells you how much statistical power you had given your sample size and the effect size that you got. This type of power analysis is generally discouraged by methodologists because it is redundant with what you already know from your results - if you receive a non-significant finding, it is already apparent that you did not have enough power to detect effect sizes around and below the one you got, and determining whether you had enough power would be useless.

```
library(pwr)

# First example
pwr.r.test(r = .20, sig.level = .05, power = .80)
```

```
##
##      approximate correlation power calculation (arctangh transformation)
##
##               n = 193.0867
##               r = 0.2
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

```
# Second example
pwr.r.test(r = .30, sig.level = .05, power= .90)
```

```
##
```

```
##      approximate correlation power calculation (arctangh transformation)
##
##               n = 111.8068
##               r = 0.3
##       sig.level = 0.05
##           power = 0.9
##     alternative = two.sided
```

```
# Third example (sensitivity power analysis)
pwr.r.test(n = 100, sig.level = .05, power = .80)
```

```
##
##      approximate correlation power calculation (arctangh transformation)
##
##               n = 100
##               r = 0.275866
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

For Spearman's and Kendall's correlations, to my knowledge there is no readily available software for power analyses, but this paper may be useful as it gives benchmarks for sample size planning given different values of Spearman's and Kendall's correlations: https://www.hilarispublisher.com/open-access/sample-size-charts-of-spearman-and-kendall-coefficients.pdf.

3 Regression

In the following examples, we will use the same dataset that was used for the correlations document - the SCID-AMPD data.

3.1 SPSS

3.1.1 Simple Linear Regression

Here, we will run a simple linear regression where we would like to predict functional impairment from depression scores:

- Analyze -> Regression -> Linear
- Enter the appropriate independent and dependent variables (in this case, IV is phq9 and DV is who)
- Click `Statistics` and select `Confidence intervals`, and `Durbin Watson` under `Residuals`; click `Continue`
- Click `Plots` and select `Histogram` and `Normal probability plot` under Standardized Residual Plots; drag `*ZRESID` to Y and `*ZPRED` to X; click `Continue`
- Click `Save` and select Unstandardized (under Predicted Values), each of the distances, Standardized, Studentized, and Studentized deleted residuals (under Residuals), and Standardized DfBetas and DfFits (under Influence Statistics); click `Continue`
- Click `OK`

In the output, there are a few tables and charts of interest. First, the `Model Summary` table provides multiple R, R^2, adjusted R^2, and the Durbin-Watson test statistic. Second, the `ANOVA` table provides the model F-test. Third, the `Coefficients` table provides the beta coefficients (standardized and unstandardized) along with respective standard errors and confidence intervals. The charts that are produced show: (a) a histogram of the residuals, where this histogram should appear approximately normal; (b) a P-P plot of the residuals, where the dots should be approximately along the solid line; and (c) a scatterplot of predicted values against residuals to assess constant variance and linearity. If you would like to add a LOESS curve to this third chart to better assess these assumptions, you can double-click on the chart and click the `Add Fit Line at Total` button on the top right.

Deviations regarding the **assumptions of constant variance and linearity** can *also* be assessed with a variation of the third chart produced above. Creating a scatterplot with unstandardized predicted values on the x-axis and studentized residuals on the y-axis, along with a LOESS curve, can be useful. The LOESS curve should be relatively straight and there should be no noticeable patterns in the residuals going from left to right. Constant variance can also be tested using the Breusch-Pagan test, which can be found if the regression model was alternately done through:

- Analyze -> General Linear Model -> Univariate
- Put the dependent variable of interest under `Dependent Variable` and the independent variable under `Covariate`
- Click `Options` and select Breusch-Pagan test; click `Continue`
- Click `OK`

Deviations regarding the **assumption of normality** can be assessed with the histogram of residuals, the P-P plot, and looking at descriptive statistics for the residuals. Because the residuals were indeed saved, you can run descriptives and Shapiro-Wilk's test. To get both, Analyze -> Descriptive Statistics -> Explore, then drag the Standardized Residual variable into `Dependent List`, click `Plots` and select Histogram and Normality plot with tests, then click `Continue` and `OK`.

To check for outliers, you can take a look at descriptives of the model statistics you saved above (i.e., distances, residuals, inflence statistics). Similar to what was done with the Standardized Residuals, to look at outliers:

- Analyze -> Descriptive Statistics -> Explore
- Drag Studentized Deleted Residual, Mahalanobis Distance, Cook's Distance, Leverage Value, DfFit, and DfBeta into `Dependent List`

- Click `Statistics` and select `Outliers`; click `Continue`
- Click `OK`

In the output, you will see descriptive statistics for each of these values, the top 5 lowest and highest values on each of these variables, and boxplots that can indicate outliers on these variables. As a reminder: high leverage and Mahalanobis distance has to do with outliers in the IVs, studentized residuals have to do with outliers in terms of model residuals, and influence (Cook's D, DfFits, DfBeta) has to do with outliers that affect parameter estimates.

Another interesting plot that can be created is one that has Leverage values on the x-axis and Standardized Residuals on the y-axis. If a particular participant has high values on both of these variables (i.e., their data point is in the top right corner of this graph), then it there is evidence that this participant may be an outlier. To create this chart:

- Go to Chart Builder, create a scatterplot, and drag Leverage to the x-axis and Standardized Residuals to the y-axis; click `OK`
- To determine which participant is which data point, you can double click on the chart, click `Data Label Mode` at the top right, and click on the data points to determine the participant

3.1.2 Multiple Linear Regression

Here, we will run a multiple regression model where we would like to predict functional impairment from depression scores, anxiety scores, and personality pathology scores. The procedure to estimate a multiple regression model is very similar to that of simple linear regression. The steps will basically be the same with the addition of:

- Selecting `Part and partial correlations` and `Collinearity diagnostics` under `Statistics`

Checking the assumptions will also largely be the same procedure as in simple linear regression, with the addition of focusing on collinearity diagnostics (e.g., VIF) and multivariate outliers (e.g., with Mahalanobis distance, Leverage values).

3.1.3 Hierarchical regression

Hierarchical regression is a methodology where you are interested in comparing the predictive performance of a larger model with a smaller model. In some psychological research, this is often called 'incremental validity', and basically tries to answer whether an additional independent variable explains a significant amount of variance in an outcome *above and beyond* another independent variable. For example, we can see whether the addition of GAD-7 and PID-5 scores provide incremental validity (with respect to statistical significance) for predicting functional impairment, above and beyond PHQ-9 scores. Note, hierarchical regression is not to be confused with hierarchical linear models, which is basically another name for multilevel models. To do this in SPSS:

- Analyze -> Regression -> Linear
- Put the phq9 and who variables in the appropriate spots
- Click `Next` to create a second block, add pid5 and gad7 variables
- Under `Statistics`, select `R-squared change`; click `Continue` and click `OK`

Under the `Model Summary` table, the p-value associated with the R^2 change (or in other words, the F change) can be found.

3.1.4 Robust Regression

If it is the case that the assumption of constant variance is severely violated, you can use robust standard errors which correct for this violation and produce more accurate confidence intervals and p-values. To do this in SPSS:

- Analyze -> General Linear Model -> Univariate

- Put the dependent variable of interest under `Dependent Variable` and the independent variable under `Covariate`
- Click `Options` and select `Parameter estimates with robust standard errors`; click `Continue`
- Click `OK`

If it is the case that there are a few outliers, you can use Quantile Regression which is calculated using medians, rather than means as in typical regression.

3.2 R

3.2.1 Simple Linear Regression

Here, we will run a simple linear regression model where we would like to predict functional impairment from depression scores. Note that we are using the same dataset that was imported in the Correlation section of this document. See that section for the code on how to import the relevant dataset into R.

The following code examples show how to estimate unstandardized and standardized coefficients (with either standard errors or confidence intervals), along with indices of global model performance (e.g., R^2).

The `lm()` function is used to estimate regression models, and the results of such models can be put into an object for further analyses. The `lm()` function uses a particular syntax style where the dependent variable is written to the left of `~` and the independent variable is written to the right of `~`. The `jtools` package is a package that provides clean output for regression models, using its `summ()` function.

```
library(jtools) # Good regression package

model1 <- lm(who ~ phq9, data = df)
summ(model1, digits = 3) # Basic output (unstandardized)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(1,106)	124.606
R^2	0.540
Adj. R^2	0.536

	Est.	S.E.	t val.	p
(Intercept)	1.135	0.101	11.209	0.000
phq9	0.090	0.008	11.163	0.000

Standard errors: OLS

```
summ(model1, confint = T, digits = 3) # Basic output + CIs (unstandardized)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(1,106)	124.606
R^2	0.540
Adj. R^2	0.536

	Est.	2.5%	97.5%	t val.	p
(Intercept)	1.135	0.934	1.336	11.209	0.000
phq9	0.090	0.074	0.106	11.163	0.000

Standard errors: OLS

```
summ(model1, scale = T, transform.response = T, digits = 3) # Standardized
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(1,106)	124.606
R^2	0.540
Adj. R^2	0.536

	Est.	S.E.	t val.	p
(Intercept)	0.000	0.066	0.000	1.000
phq9	0.735	0.066	11.163	0.000

Standard errors: OLS; Continuous variables are mean-centered and scaled by 1 s.d.

```
summ(model1, scale = T, transform.response = T, confint = T, digits = 3) # Standardized with CIs
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(1,106)	124.606
R^2	0.540
Adj. R^2	0.536

	Est.	2.5%	97.5%	t val.	p
(Intercept)	0.000	-0.130	0.130	0.000	1.000
phq9	0.735	0.605	0.866	11.163	0.000

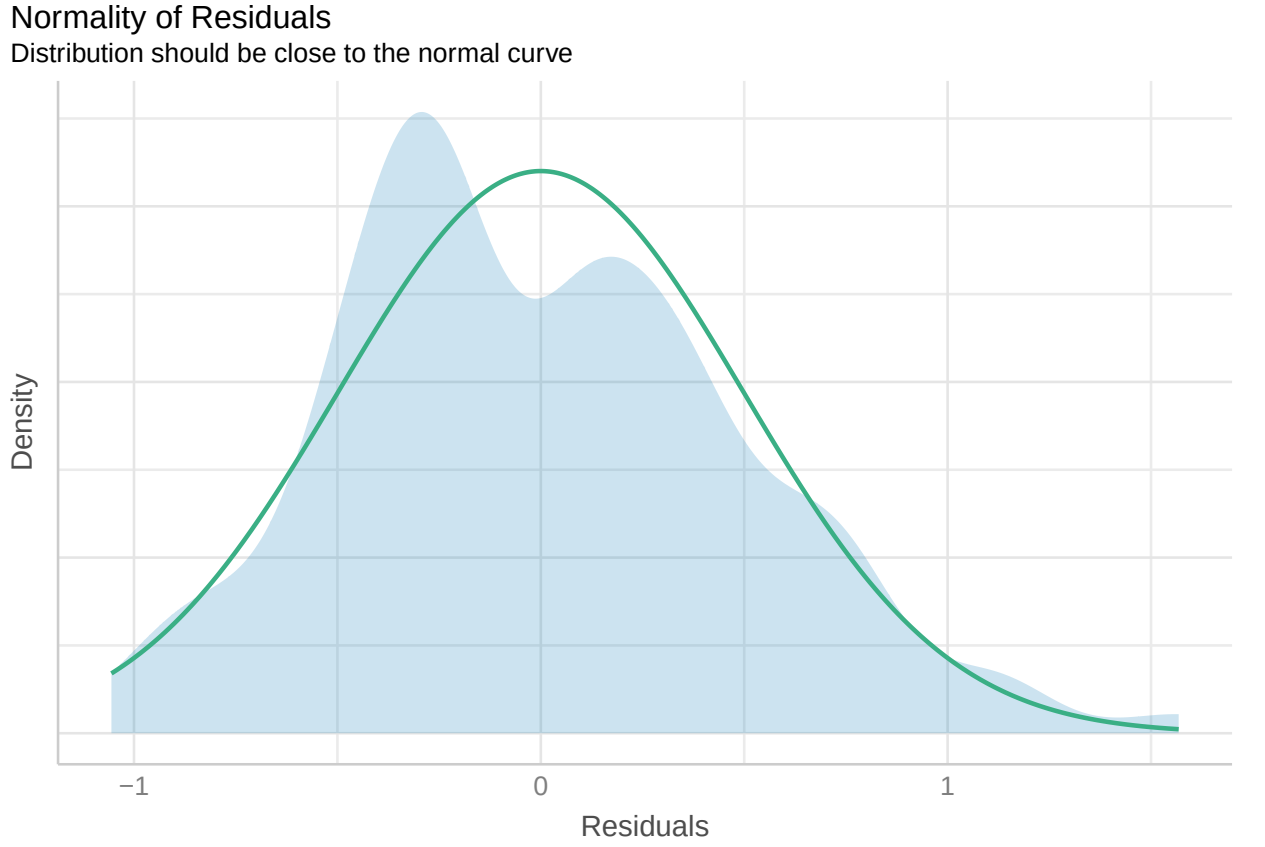
Standard errors: OLS; Continuous variables are mean-centered and scaled by 1 s.d.

We can easily look at whether there is evidence that assumptions are violated with the `performance` package, along with other diagnostics (e.g., outliers).

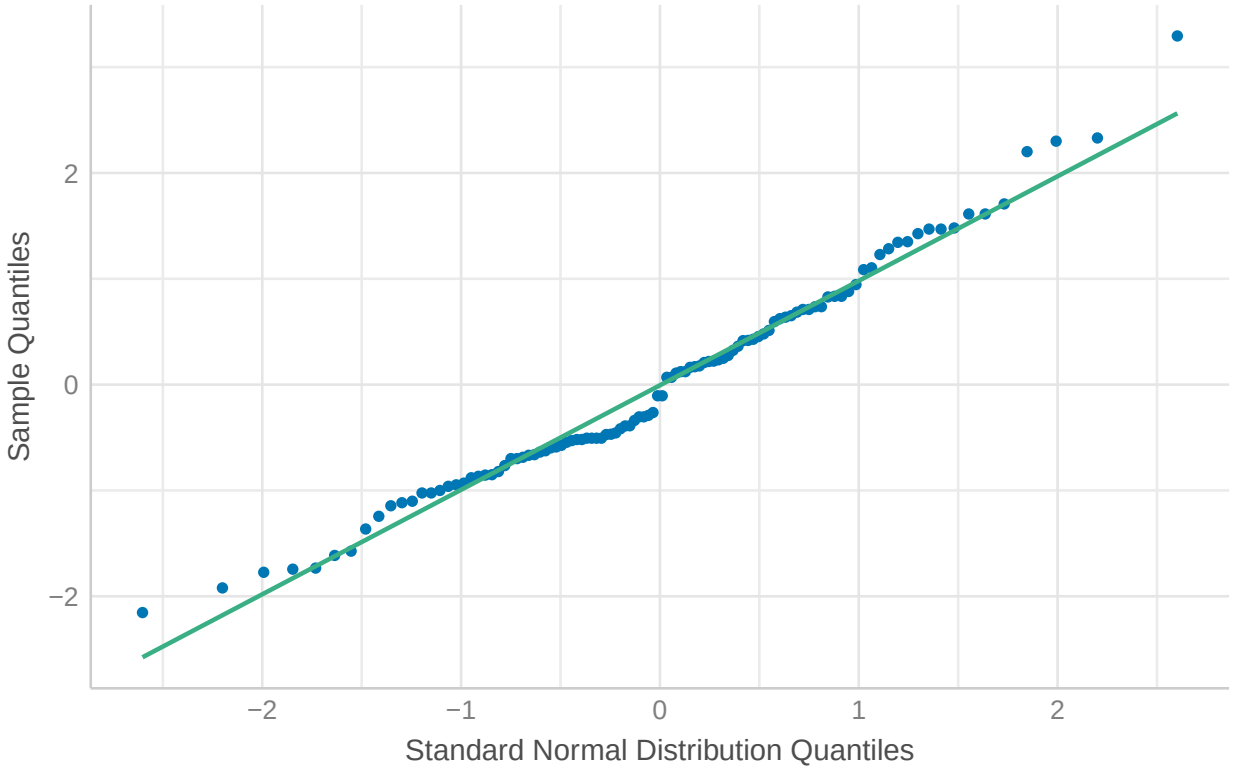
```
library(performance)
library(car)

# Normality of residuals
check_normality(model1) # Uses Shapiro-Wilks test
```

```
## OK: residuals appear as normally distributed (p = 0.129).
```

```
check_normality(model1) %>% plot()
```



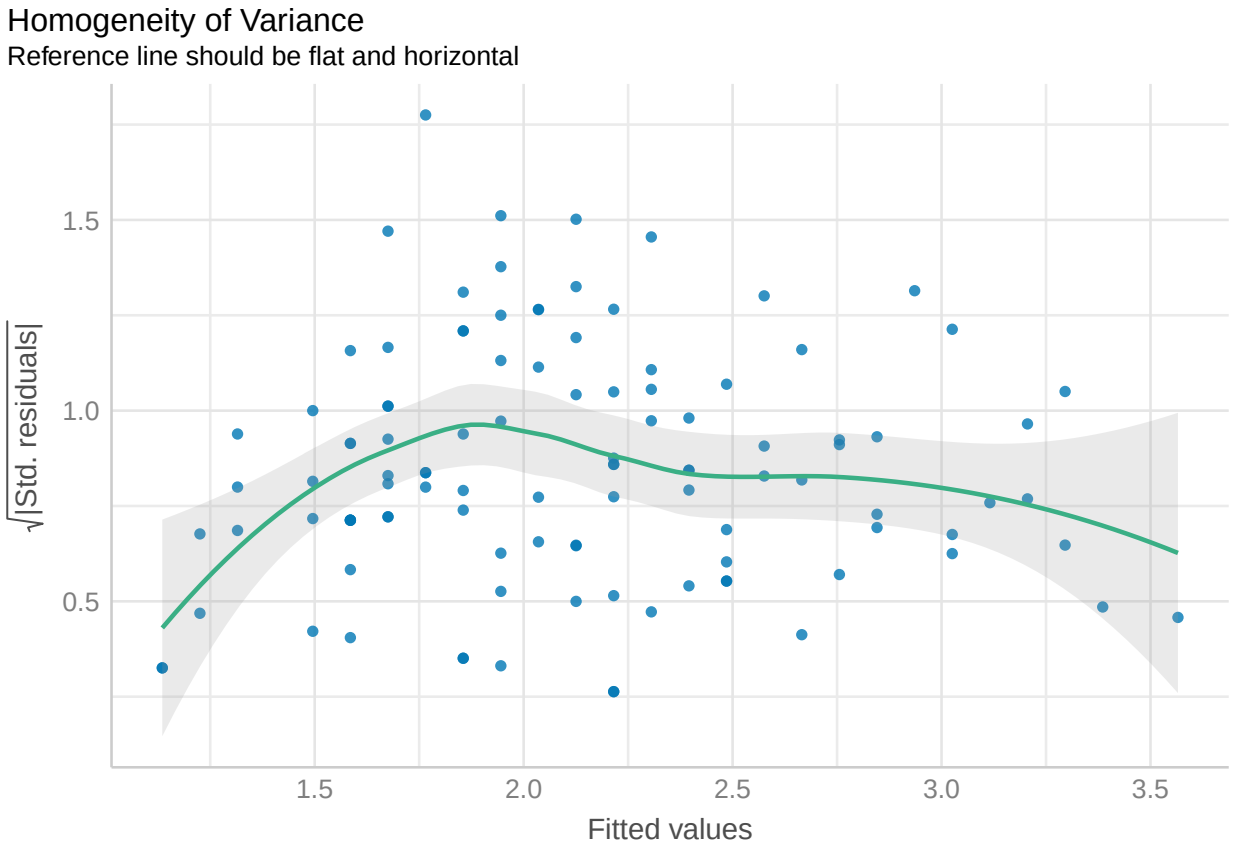

```
check_normality(model1) %>% plot(type = "qq")
```



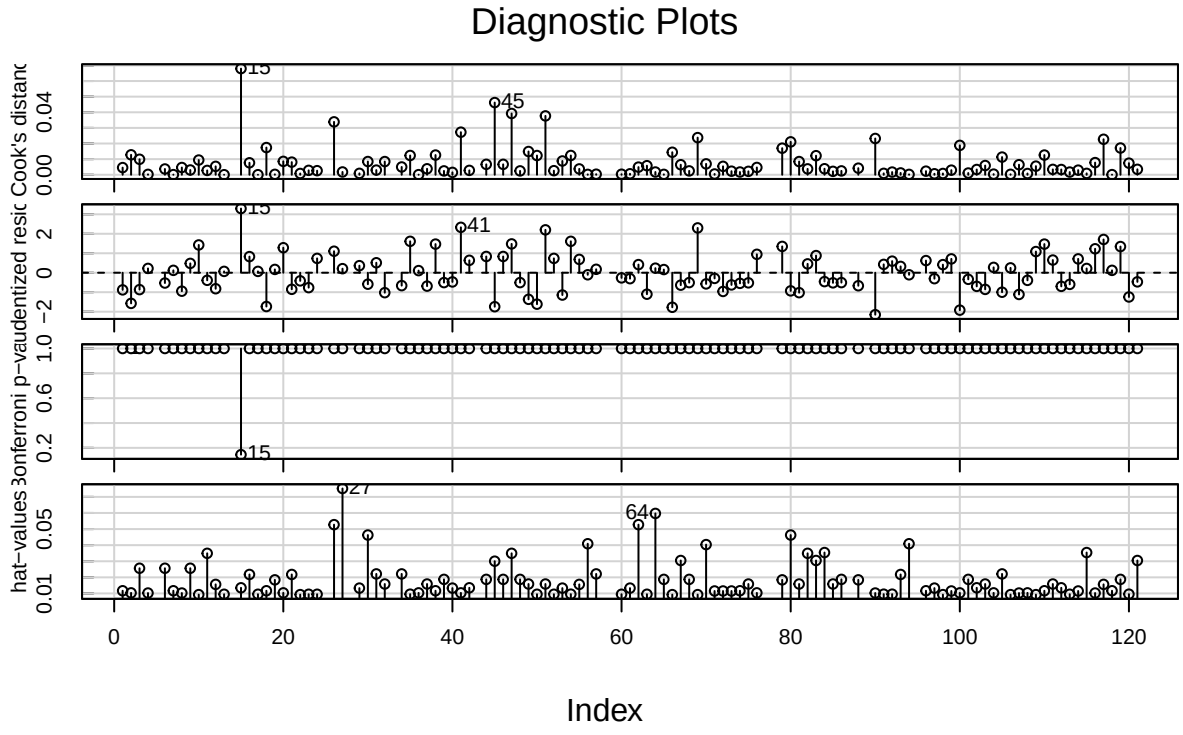

```
# Constant variance
check_heteroscedasticity(model1) # Uses Breusch-Pagan test
```

```
## OK: Error variance appears to be homoscedastic (p = 0.645).
```

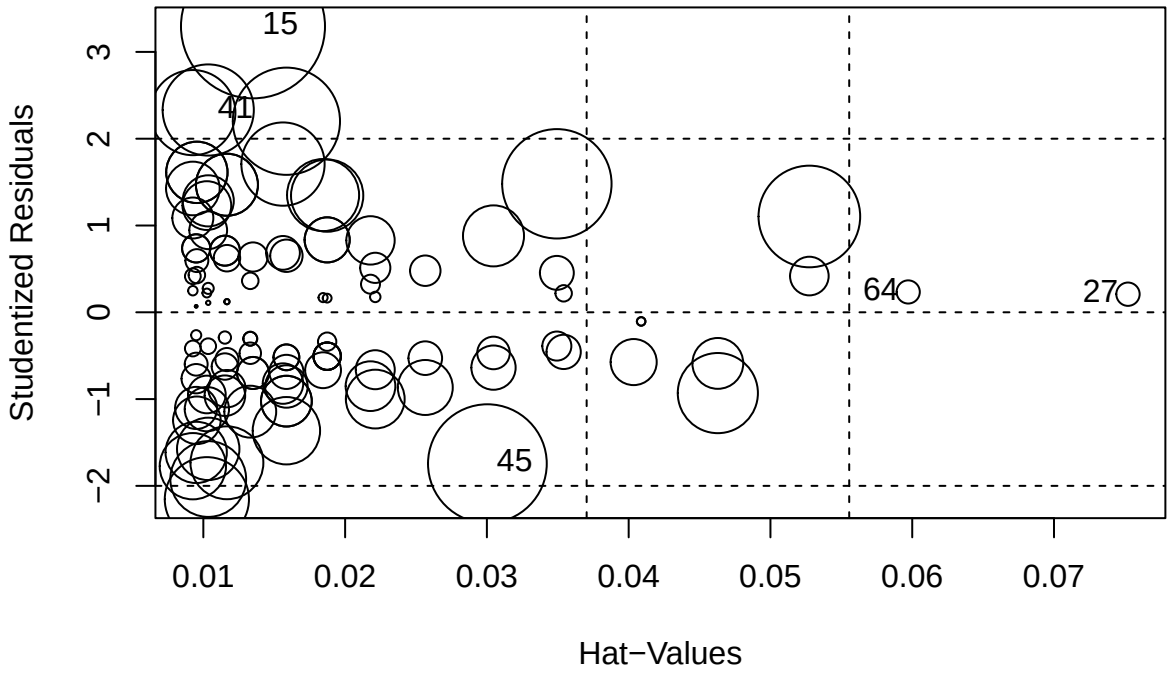
```
check_heteroscedasticity(model1) %>% plot()
```




```
# Outliers
influenceIndexPlot(model1) # Shows high Cook's D, residual, and leverage (hat)
```

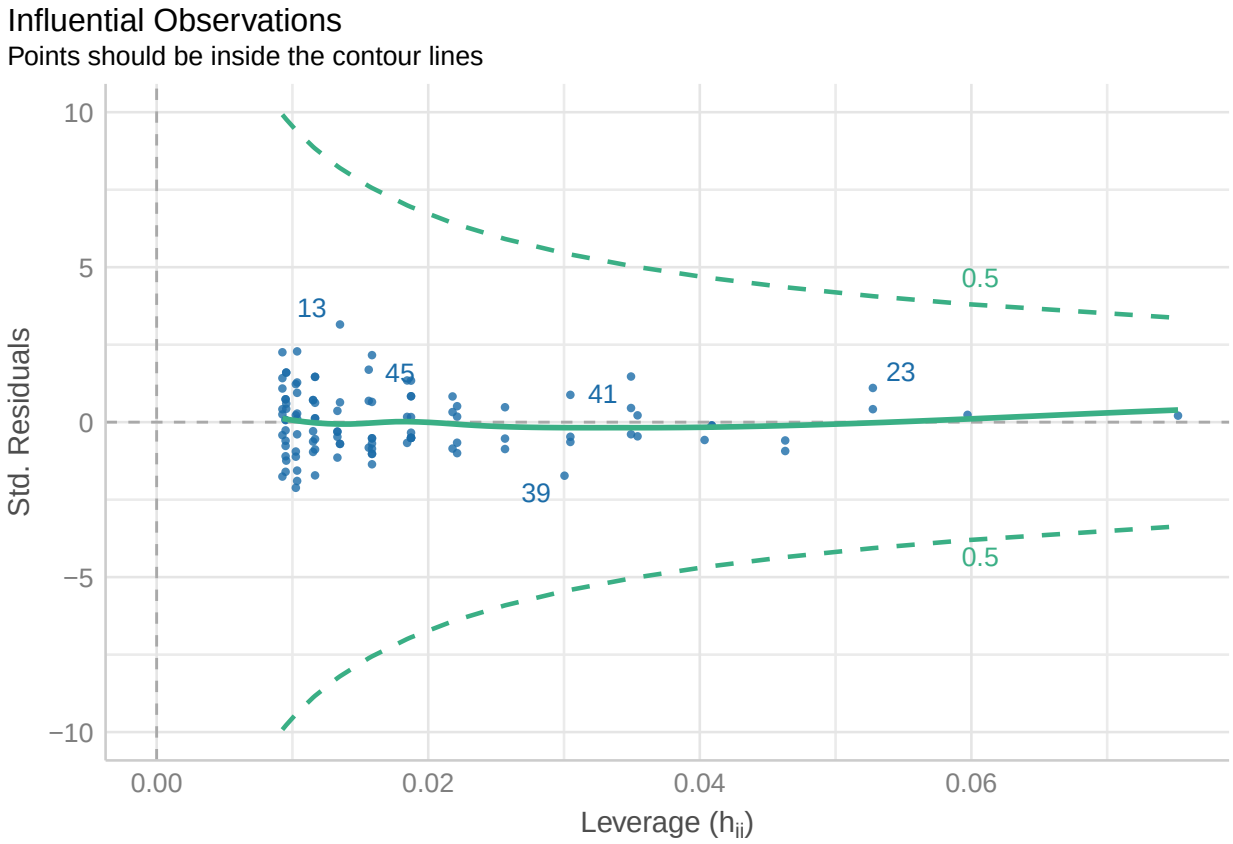


```
influencePlot(model1) # Plot of leverage (x-axis) and residuals (y-axis)
```




```
##       StudRes        Hat       CookD
## 15  3.2946489 0.01350220 0.067965537
## 27  0.2086027 0.07521830 0.001785791
## 41  2.3306821 0.01034441 0.027250104
## 45 -1.7440936 0.03003501 0.046205649
## 64  0.2342807 0.05971729 0.001758629
```

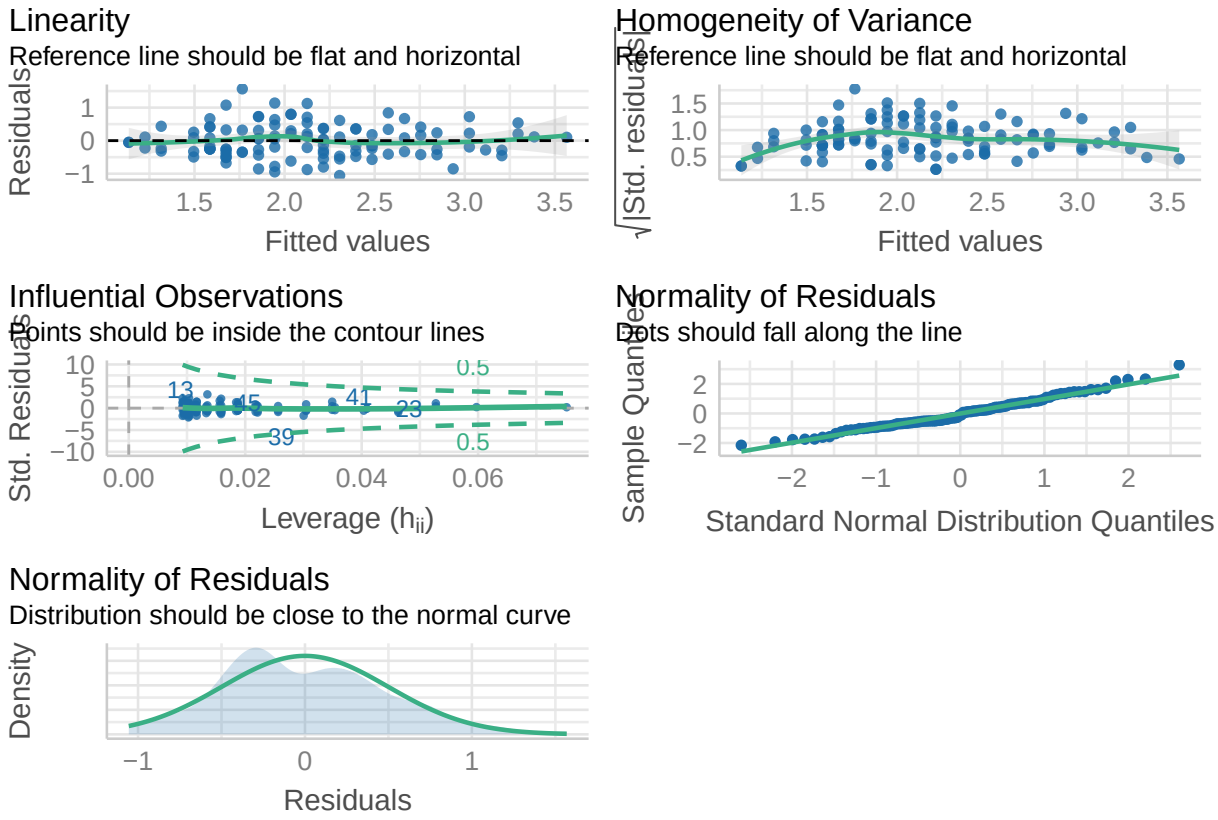
```
check_outliers(model1) %>% plot() # Similar to above
```



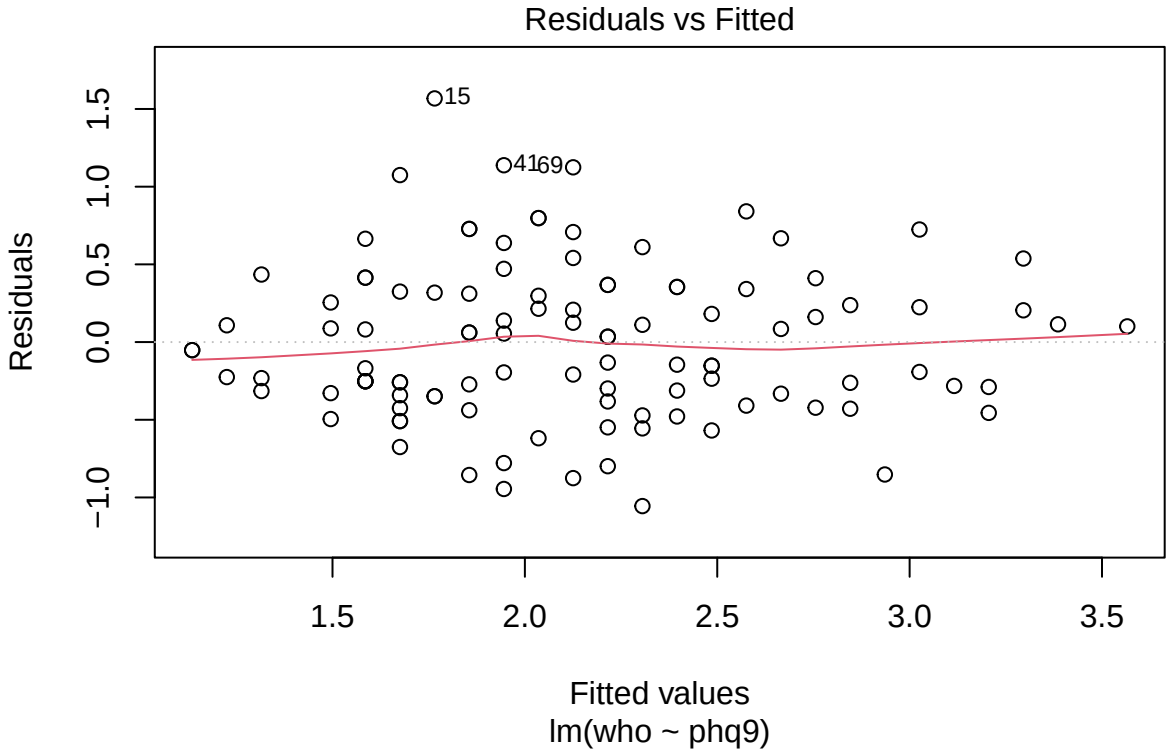

```
influence.measures(model1) %>% summary() # Includes DfBeta and DfFit
```

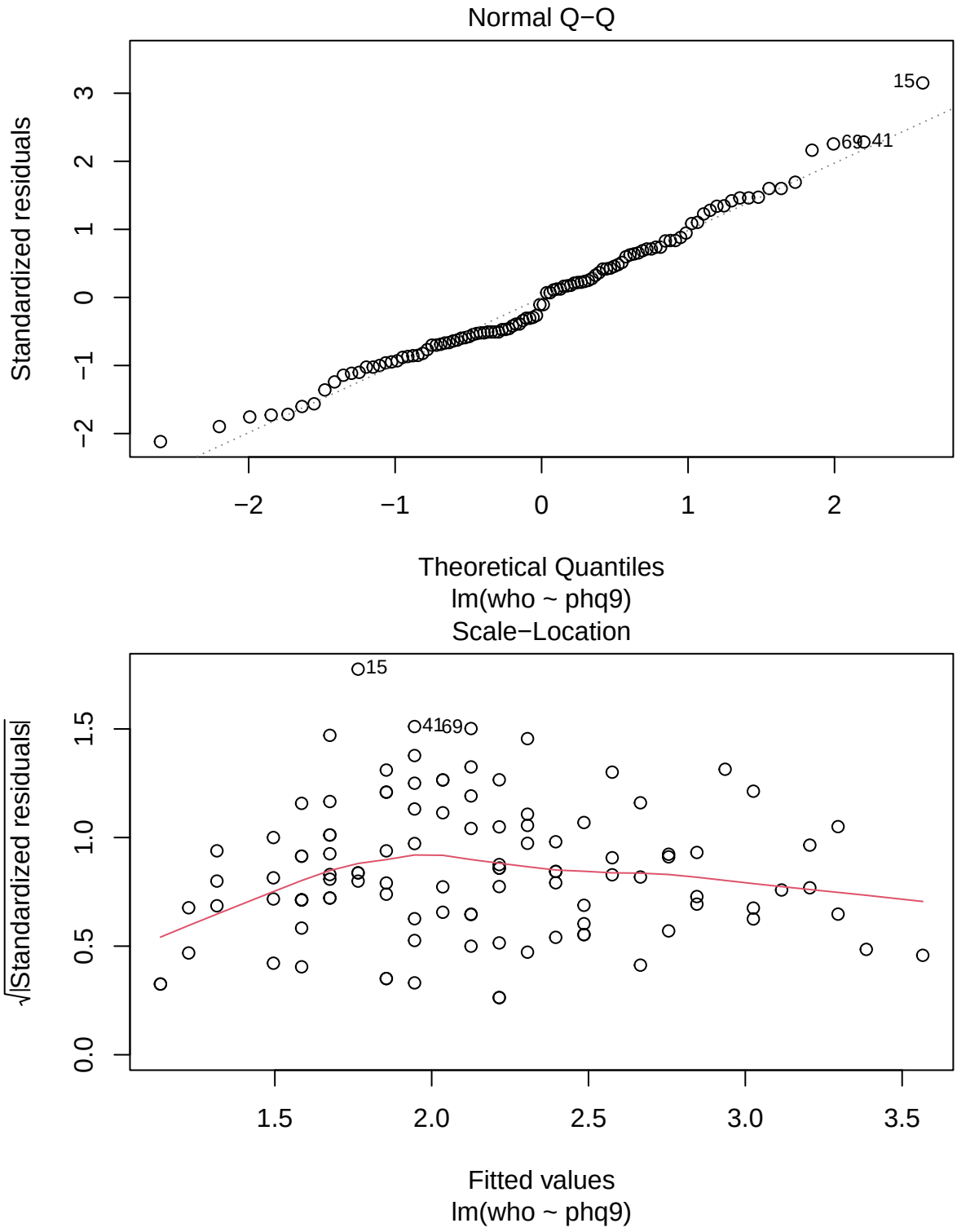
```
## Potentially influential observations of
##   lm(formula = who ~ phq9, data = df) :
##
##    dfb.1_ dfb.phq9 dffit cov.r   cook.d hat
## 15  0.34  -0.22     0.39  0.85_*  0.07   0.01
## 27 -0.04   0.06     0.06  1.10_*  0.00   0.08_*
## 30  0.07  -0.12    -0.13  1.06_*  0.01   0.05
## 41  0.18  -0.08     0.24  0.93_*  0.03   0.01
## 56 -0.02   0.02    -0.02  1.06_*  0.00   0.04
## 62 -0.06   0.09     0.10  1.07_*  0.00   0.05
## 64 -0.04   0.05     0.06  1.08_*  0.00   0.06_*
## 69  0.11   0.00     0.22  0.93_*  0.02   0.01
## 94 -0.02   0.02    -0.02  1.06_*  0.00   0.04
```

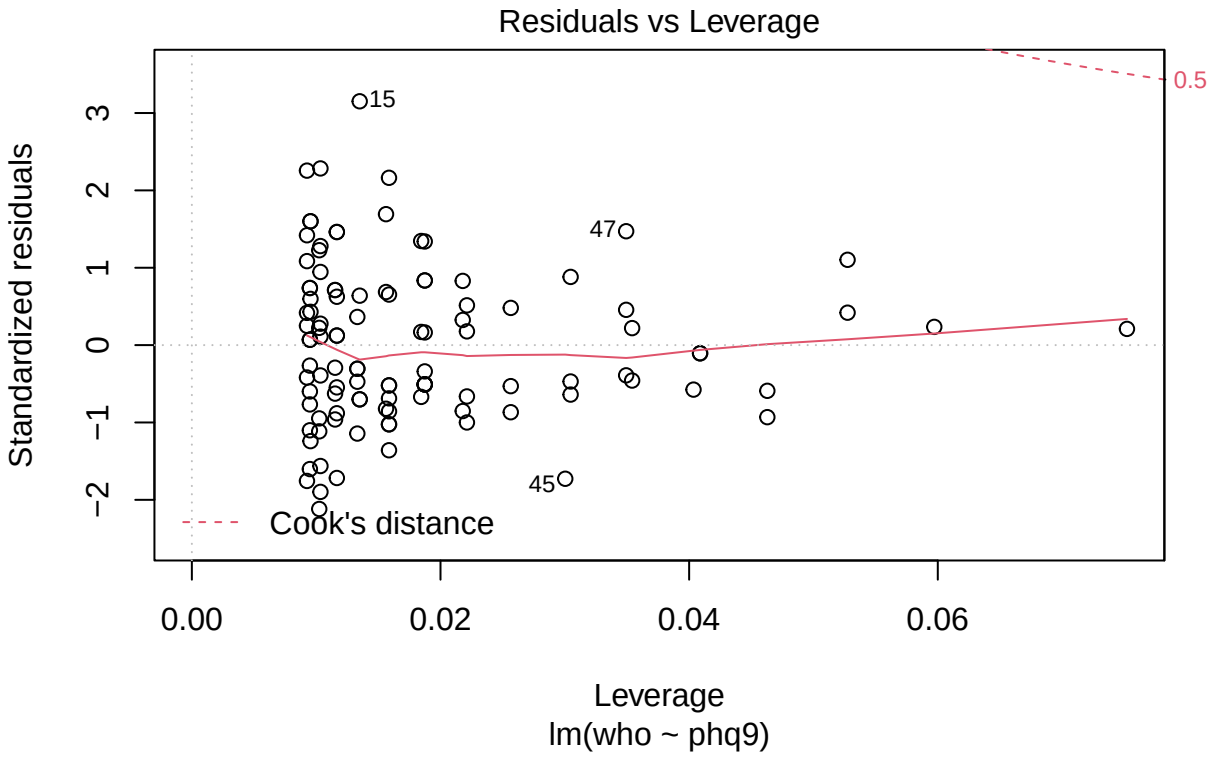
```
# A global function to run all of the above plots + linearity
check_model(model1)
```




```
# R's built-in assumption plots
plot(model1)
```

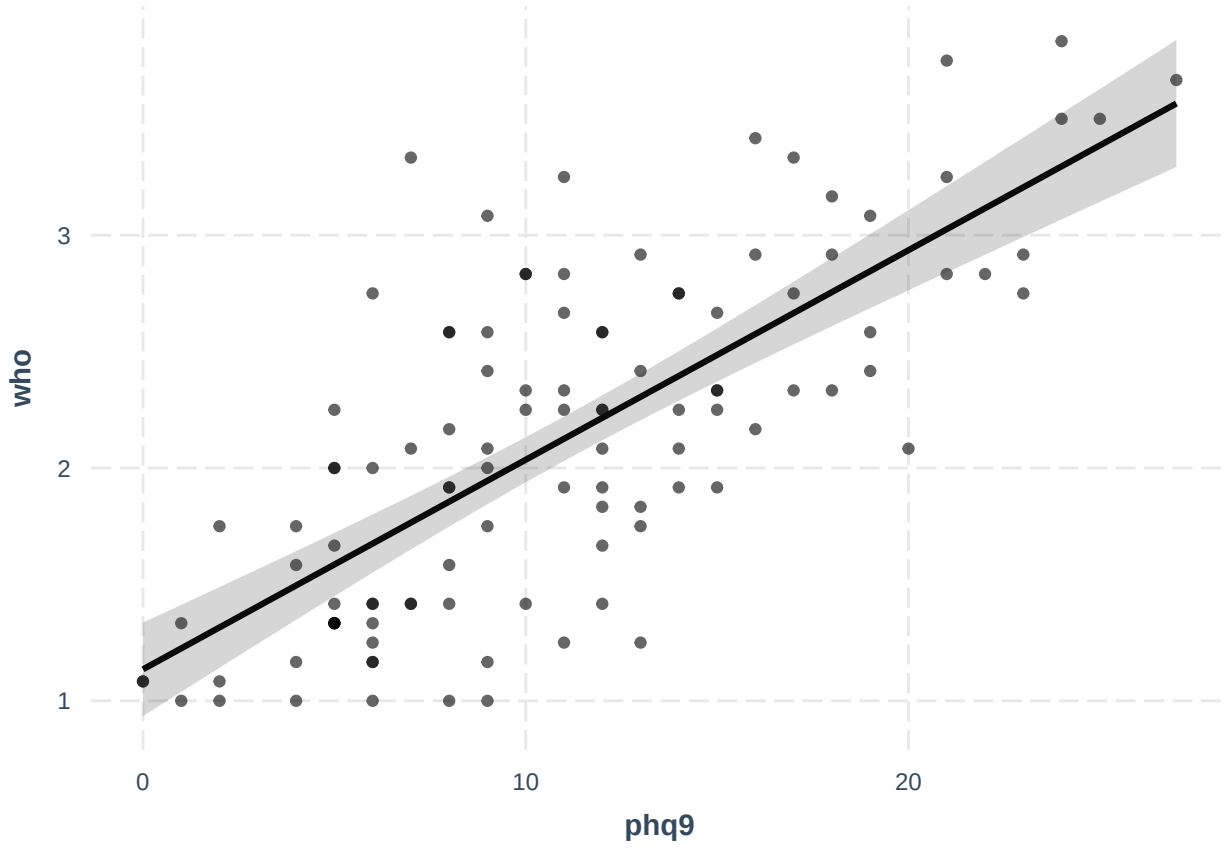



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(who ~ phq9)
15
69
41
Scale−Location
√|Standardized residuals|
Fitted values
lm(who ~ phq9)



Plotting a simple linear regression model can be done with the `jtools` package. The `effect_plot()` function takes in the model object name, the independent variable of interest, whether you want CIs around the regression line, and whether you want the actual data points to be plotted.

```
effect_plot(model1, pred = phq9, interval = T, plot.points = T)
```



3.2.2 Multiple Linear Regression

Here, we will run a multiple regression model where we would like to predict functional impairment from depression scores, anxiety scores, and personality pathology scores. Similar to the simple linear regression example, the following code shows how to estimate both unstandardized and standardized coefficients (with either SEs or CIs), along with indices of global model performance. To have more than one independent variable in the model, the names of the variables can be added into the `lm()` function using the `+` symbol.

```
# Model syntax
model2 <- lm(who ~ phq9 + gad7 + pid5, data = df)

# Model results (basic)
summ(model2)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

```
# Results with CIs
summ(model2, confint = T)
```

	Est.	S.E.	t val.	p
(Intercept)	0.89	0.15	5.89	0.00
phq9	0.07	0.02	3.77	0.00
gad7	0.02	0.02	0.73	0.47
pid5	0.33	0.16	2.00	0.05

Standard errors: OLS

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

	Est.	2.5%	97.5%	t val.	p
(Intercept)	0.89	0.59	1.19	5.89	0.00
phq9	0.07	0.03	0.10	3.77	0.00
gad7	0.02	-0.03	0.06	0.73	0.47
pid5	0.33	0.00	0.65	2.00	0.05

Standard errors: OLS

```
# Results with standardized coefficients
summ(model2, scale = T, transform.response = T)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

	Est.	S.E.	t val.	p
(Intercept)	0.00	0.06	0.00	1.00
phq9	0.56	0.15	3.77	0.00
gad7	0.10	0.14	0.73	0.47
pid5	0.15	0.08	2.00	0.05

Standard errors: OLS; Continuous variables are mean-centered and scaled by 1 s.d.

```
# Results with standardization and CIs
summ(model2, scale = T, transform.response = T, confint = T)
```

```
# Other model fit values
model_performance(model2)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

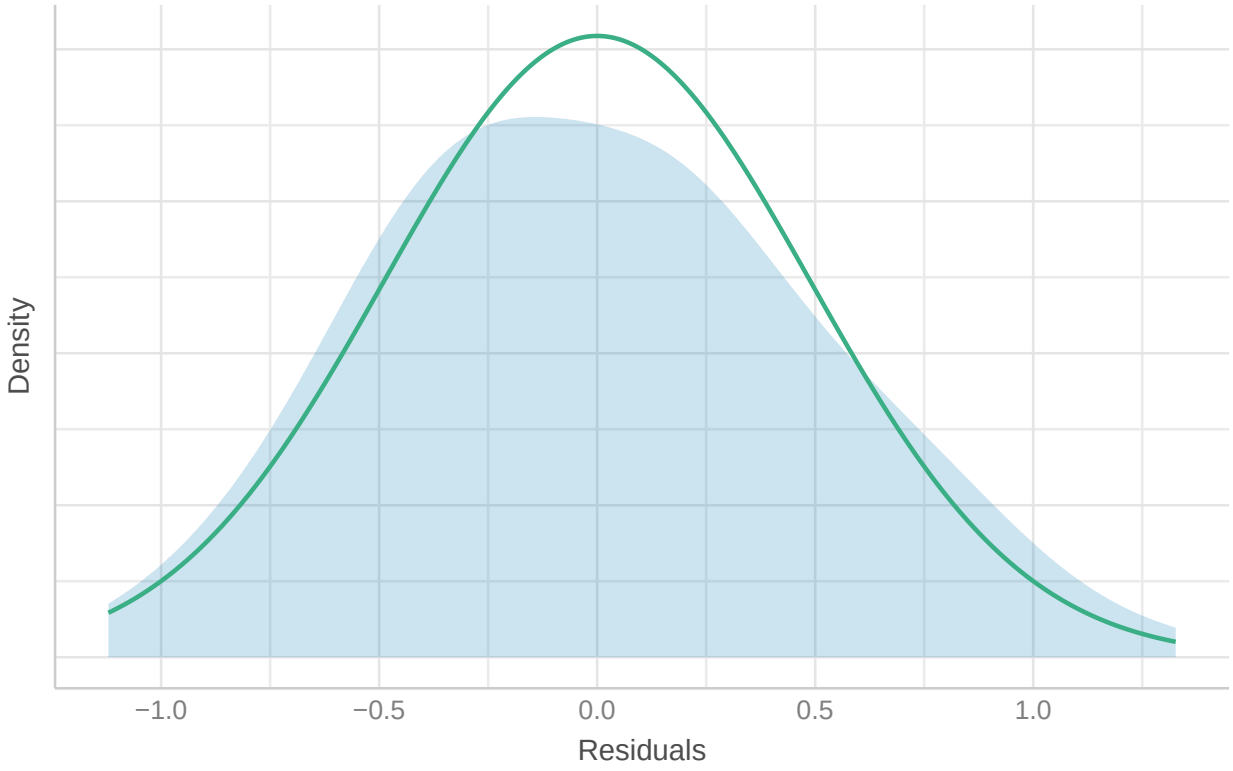
	Est.	2.5%	97.5%	t val.	p
(Intercept)	0.00	-0.13	0.13	0.00	1.00
phq9	0.56	0.26	0.85	3.77	0.00
gad7	0.10	-0.18	0.39	0.73	0.47
pid5	0.15	0.00	0.31	2.00	0.05

Standard errors: OLS; Continuous variables are mean-centered and scaled by 1 s.d.

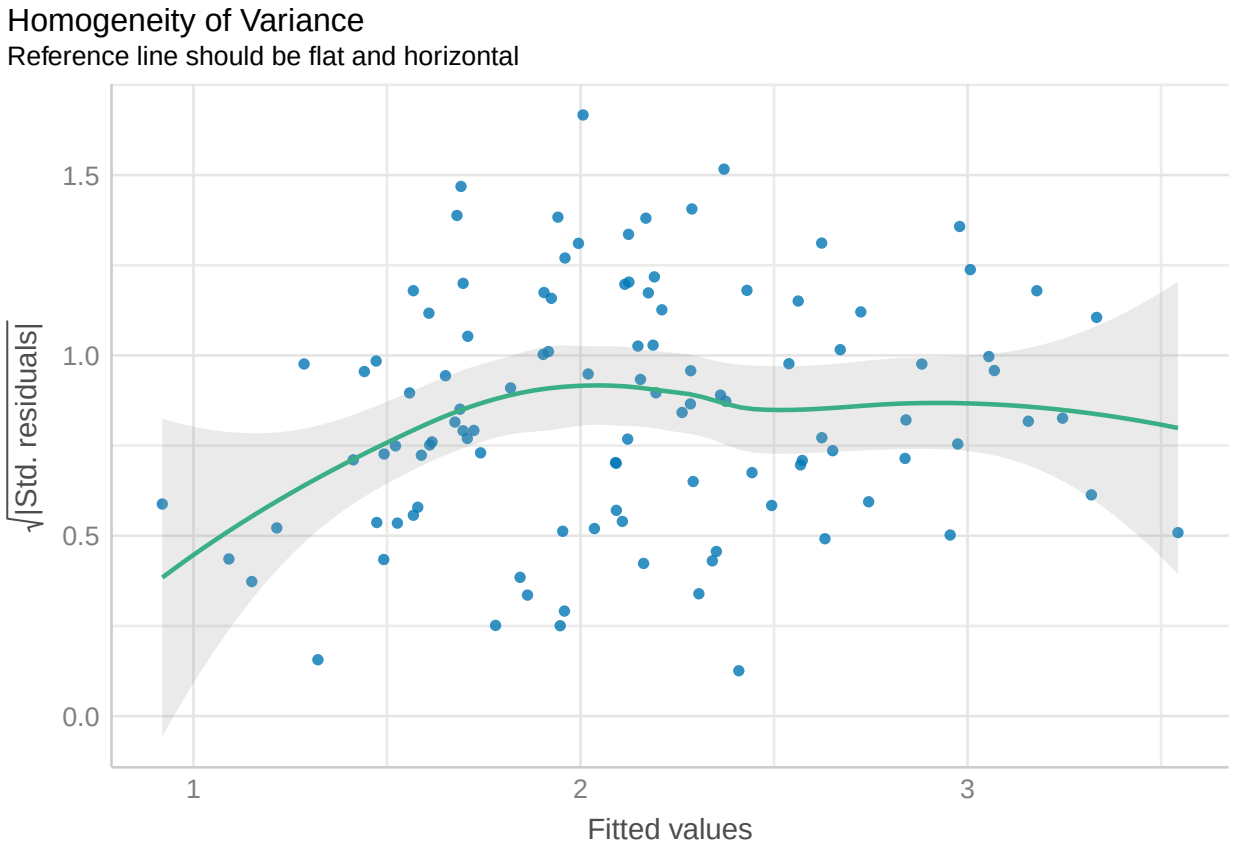
```
## # Indices of model performance
##
## AIC     |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
## ----------------------------------------------------------
## 160.521 | 173.931 | 0.560 |     0.547 | 0.486 | 0.495
```

Again, similar to the simple linear regression example, we can look at whether there is evidence of model assumptions being violated, along with other relevant model properties (e.g., collinearity in the case of multiple regression; outliers).

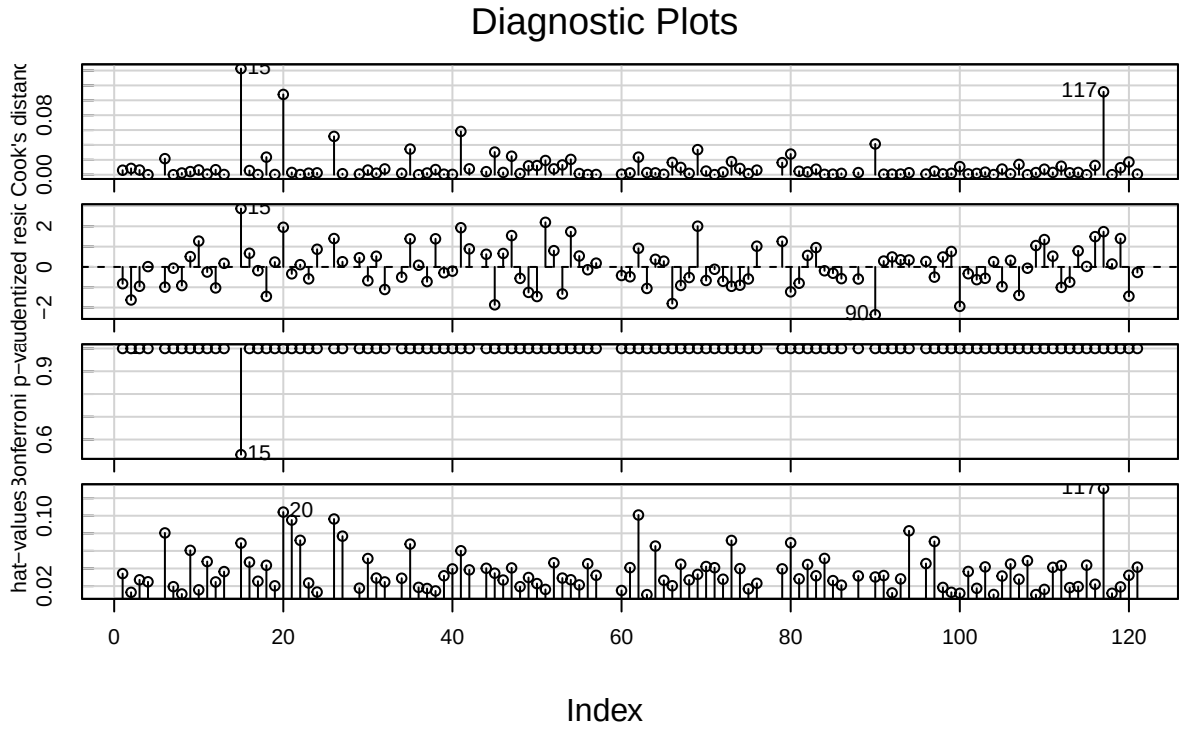
```
# Normality of residuals
check_normality(model2) %>% plot()
```



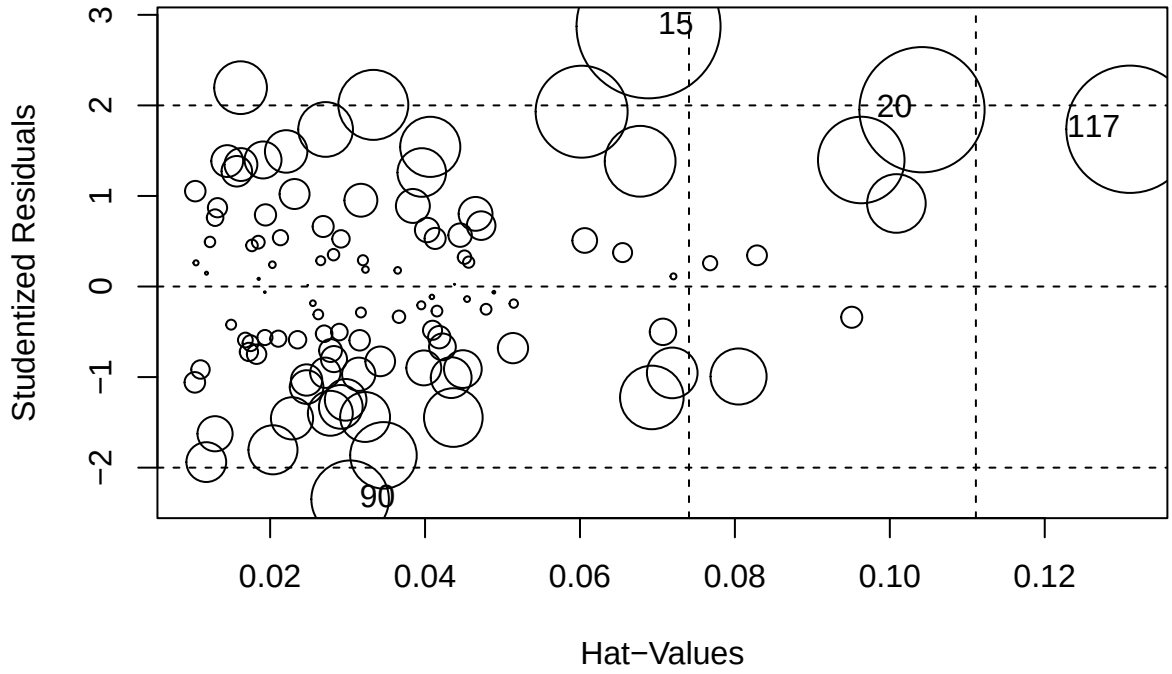

```
# Constant variance
check_heteroscedasticity(model2) %>% plot()
```




```
# Outliers
influenceIndexPlot(model2) # Shows high Cook's D, residual, and leverage (hat)
```

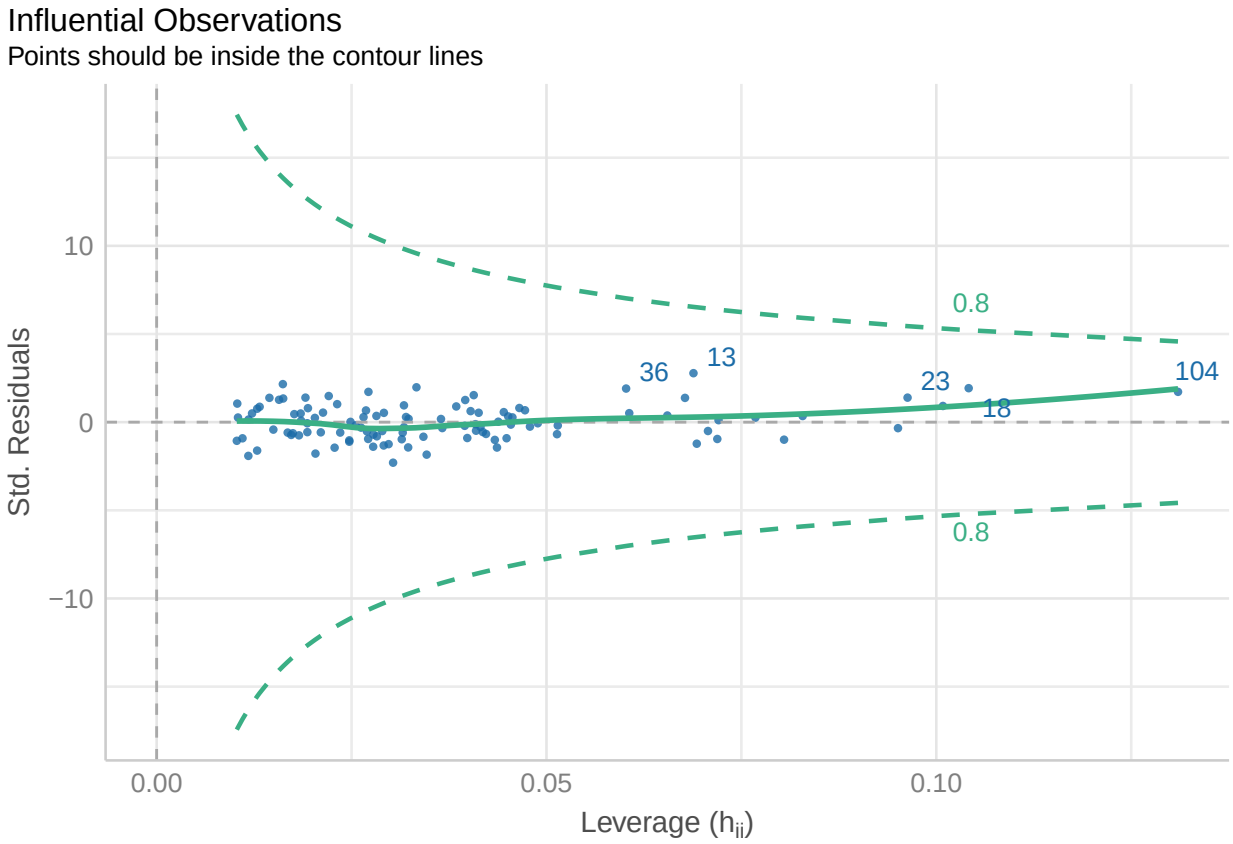


```
influencePlot(model2) # Plot of leverage (x-axis) and residuals (y-axis)
```




```
##       StudRes        Hat      CookD
## 15   2.872920 0.06885180 0.14262727
## 20   1.952430 0.10415347 0.10788085
## 90  -2.348872 0.03032572 0.04134078
## 117  1.736433 0.13100342 0.11147710
```

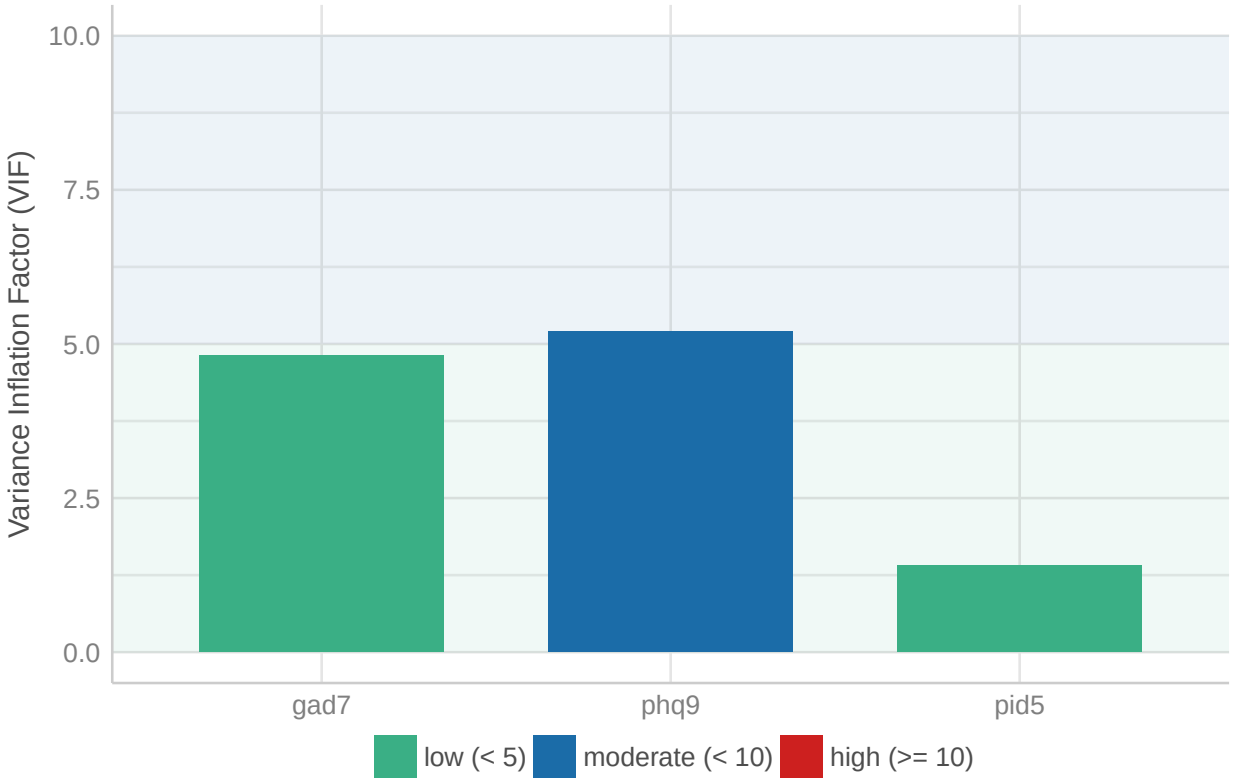
```
check_outliers(model2) %>% plot() # Similar to above
```



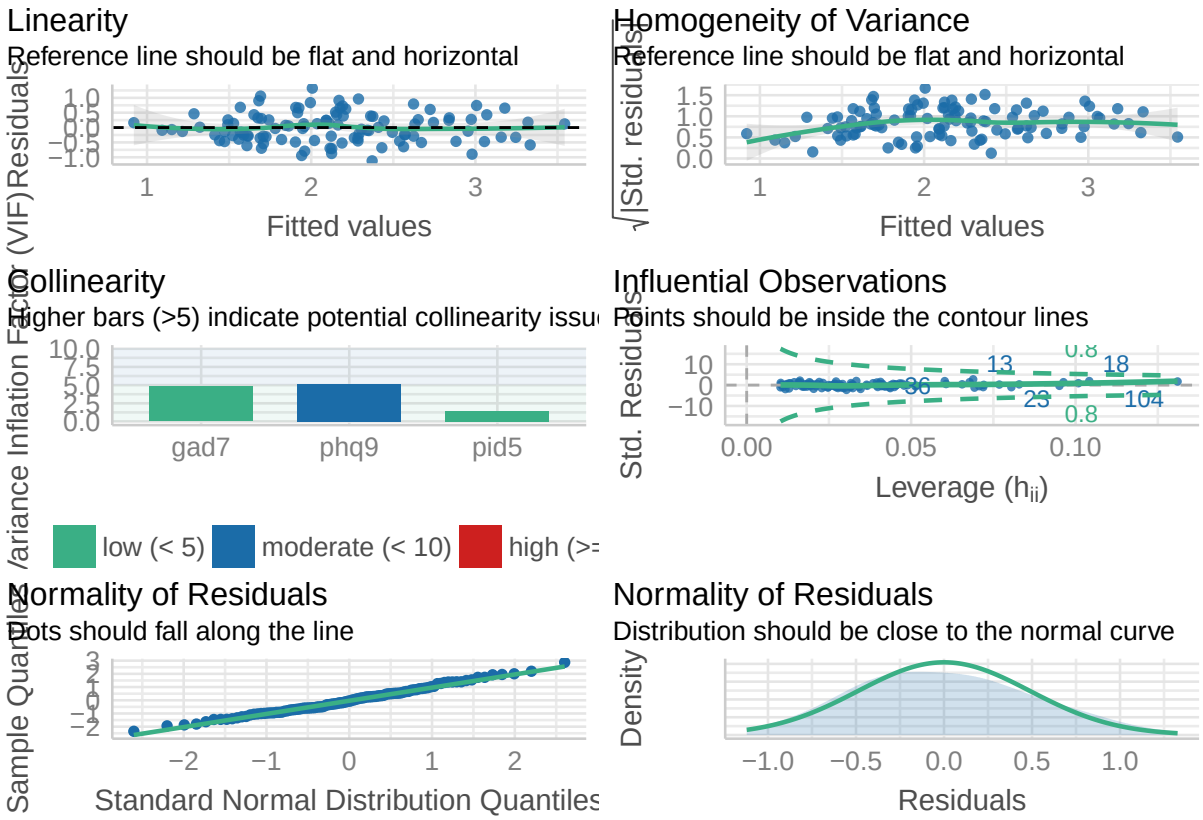

```
influence.measures(model2) %>% summary() # Includes DfBeta and DfFit
```

```
## Potentially influential observations of
##   lm(formula = who ~ phq9 + gad7 + pid5, data = df) :
##
##     dfb.1_ dfb.phq9 dfb.gad7 dfb.pid5 dffit   cov.r   cook.d hat
## 15  -0.32  -0.34     0.07     0.70     0.78_*  0.82_*  0.14   0.07
## 20   0.46   0.51    -0.52    -0.36     0.67_*  1.00    0.11   0.10
## 21  -0.05  -0.09     0.07     0.07    -0.11    1.14_*  0.00   0.10
## 22   0.03   0.01    -0.01    -0.03     0.03    1.12_*  0.00   0.07
## 27  -0.03   0.04    -0.01     0.00     0.07    1.12_*  0.00   0.08
## 51   0.15  -0.11     0.03     0.02     0.28    0.88_*  0.02   0.02
## 62   0.07   0.17    -0.03    -0.21     0.31    1.12_*  0.02   0.10
## 90   0.06   0.26    -0.34    -0.03    -0.42    0.87_*  0.04   0.03
## 94   0.10   0.01    -0.02    -0.07     0.10    1.13_*  0.00   0.08
## 117 -0.14   0.46    -0.56     0.30     0.67_*  1.07    0.11   0.13_*
```

```
# Collinearity
check_collinearity(model2) %>% plot()
```

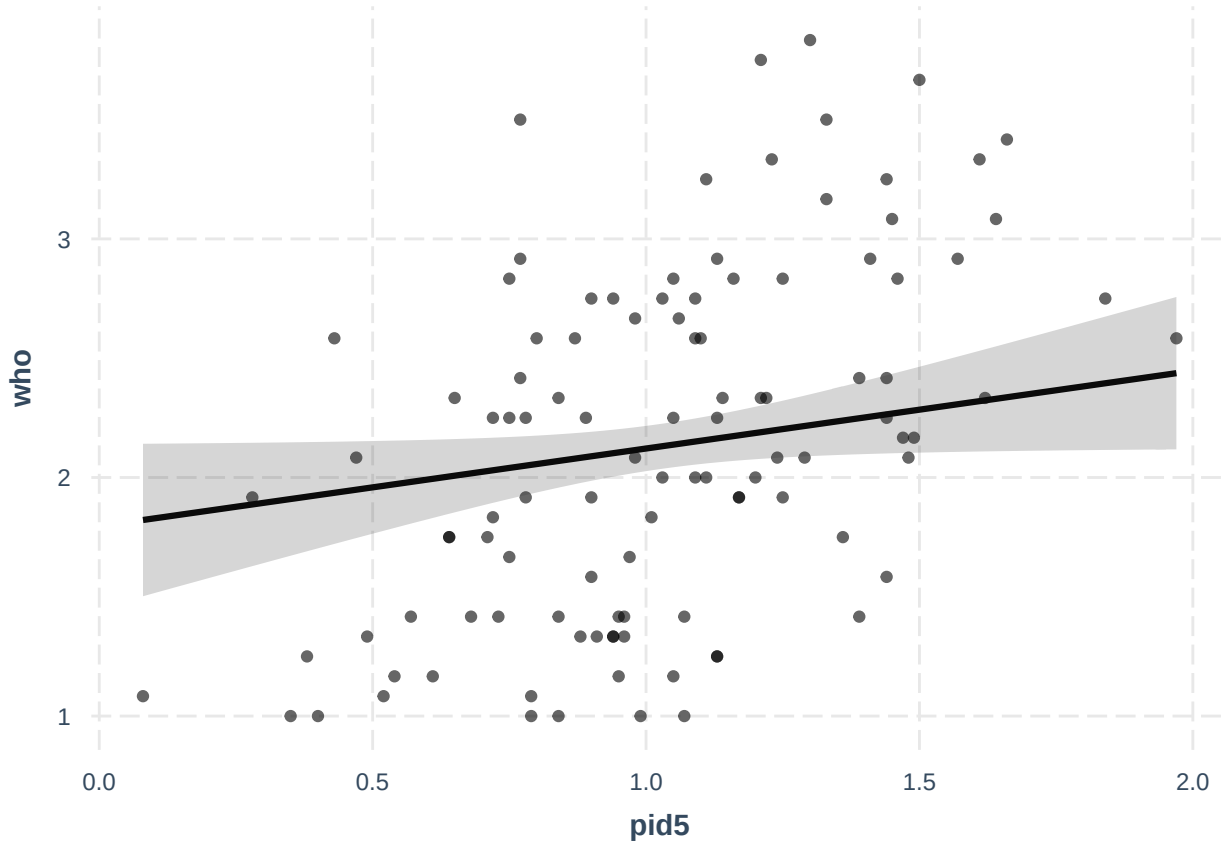



```
# All of the above + linearity
check_model(model2)
```

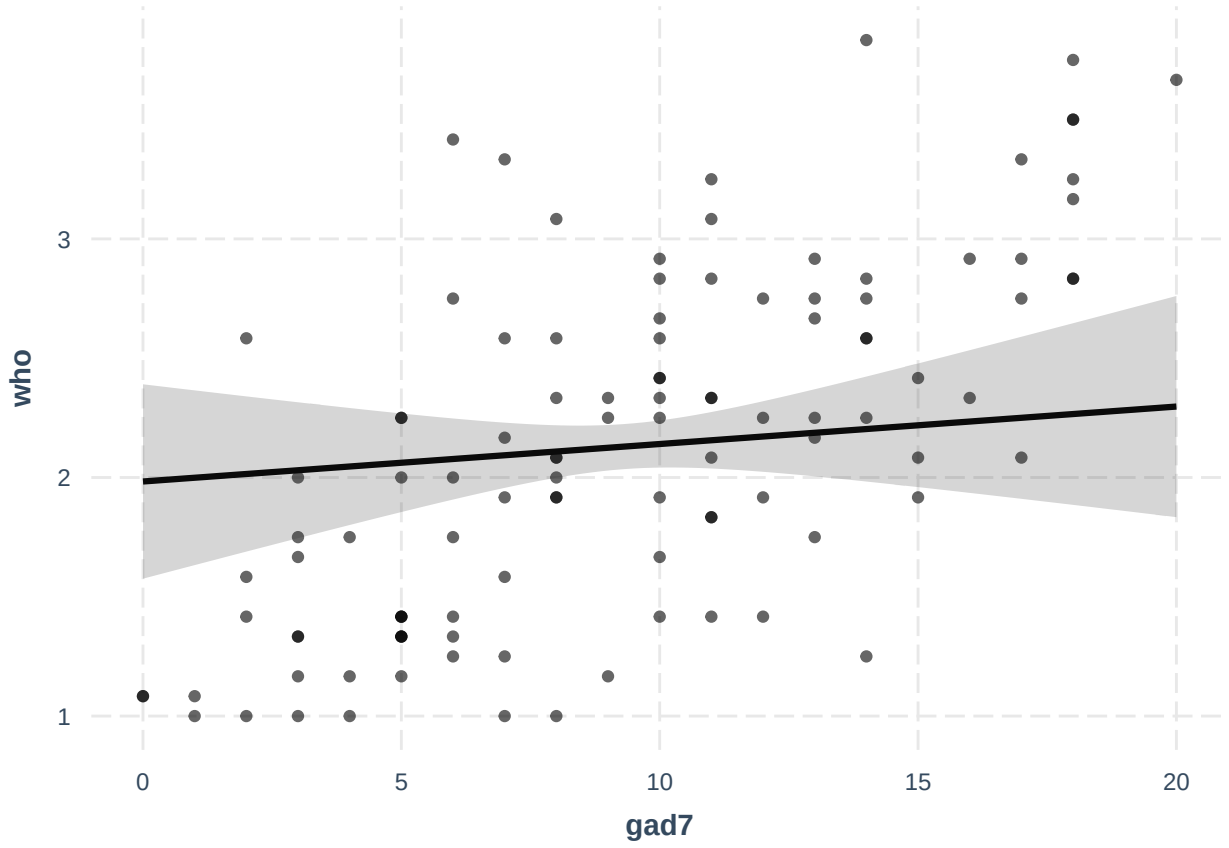


Like in simple linear regression, plotting can be done with the `jtools` package. In the case of multiple regression, you can visualize the effect of one independent variable on the dependent variable, while holding the other independent variables constant, using the `effect_plot()` function.

```
# Unique effect of pid5
effect_plot(model2, pred = pid5, interval = T, plot.points = T)
```




```
# Unique effect of gad7
effect_plot(model2, pred = gad7, interval = T, plot.points = T)
```



3.2.3 Hierarchical regression

Hierarchical regression is a methodology where you are interested in comparing the predictive performance of a larger model with a smaller model. In some psychological research, this is often called 'incremental validity', and basically tries to answer whether an additional independent variable explains a significant amount of variance in an outcome *above and beyond* another independent variable. For example, we can see whether the addition of GAD-7 and PID-5 scores provide incremental validity (with respect to statistical significance) for predicting functional impairment, above and beyond PHQ-9 scores. Note, hierarchical regression is not to be confused with hierarchical linear models, which is basically another name for multilevel models.

In the result below, we can see that GAD-7 and PID-5 scores do not account for a significant amount of variance in the outcome, above and beyond PHQ-9 scores. The p-value here is .108.

```
# Here, we are comparing model1 and model2 from above
anova(model1, model2)
```

```
## Analysis of Variance Table
##
## Model 1: who ~ phq9
## Model 2: who ~ phq9 + gad7 + pid5
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    106 26.598
## 2    104 25.482  2    1.1166 2.2786 0.1075
```

3.2.4 Robust regression

If the constant variance assumption is severely violated, then you can estimate robust standard errors within the `summ()` function by specifying `robust = T`.

```
summ(model1, robust = T)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(1,106)	124.61
R^2	0.54
Adj. R^2	0.54

	Est.	S.E.	t val.	p
(Intercept)	1.14	0.09	12.81	0.00
phq9	0.09	0.01	14.03	0.00

Standard errors: Robust, type = HC3

There are many robust regression options in R in addition to the above. See this link for more: https://cran.r-project.org/web/views/Robust.html

3.2.5 Semipartial and Partial correlations

With larger regression models, you get a R^2 value that represents the amount of variance explained in the outcome *attributable to the whole set of independent variables*. Although the R^2 value is useful, it does not tell you *which* independent variable explained *what* amount of variance. That is, which independent variable seems to be contributing the most to predicting the outcome?

Semipartial correlations can answer the question of: how correlated is an independent variable with a dependent variable, above and beyond other independent variables?

Partial correlations, on the other hand, are a bit trickier to interpret in my opinion. They can answer the question of: how correlated is an independent variable with a dependent variable, after accounting for the relation between the same dependent variable and other independent variables?

All of this being said, I find that it's useful to square semipartial correlations because that can tell you how much **variance** in a dependent variable is attributable to a given independent variable, above and beyond other independent variables.

The below syntax produces both semipartial correlations (called `part.r` in the output) and partial correlations. In this case, we are interested in the unique relations between functional impairment and the three psychopathology variables mentioned above (depression, anxiety, personality pathology).

Using these results, if we square the semipartial for pid5 (.13^2 = .017), this means that PID-5 scores account for an additional 1.7% of variance in functional impairment scores, above and beyond depression and anxiety scores.

```
summ(model2, part.corr = T)
```

Observations	108 (13 missing obs. deleted)
Dependent variable	who
Type	OLS linear regression

F(3,104)	44.06
R^2	0.56
Adj. R^2	0.55

	Est.	S.E.	t val.	p	partial.r	part.r
(Intercept)	0.89	0.15	5.89	0.00	NA	NA
phq9	0.07	0.02	3.77	0.00	0.35	0.25
gad7	0.02	0.02	0.73	0.47	0.07	0.05
pid5	0.33	0.16	2.00	0.05	0.19	0.13

Standard errors: OLS

3.3 Power Analysis Resources

Conducting a power analysis for regression models can be tricky because one can conduct their power analysis based on detecting a R^2 value of interest (i.e., a certain amount of variance explained) or based on detecting a particular regression coefficient in the model. Both choices, however, can be done in various software.

Two resources for conducting a power analysis for regression models in G*Power are the following:

- https://web.pdx.edu/~newsomj/mvclass/ho_sample%20size.pdf
- https://link.springer.com/article/10.3758/BRM.41.4.1149

In R, the one package that I know that can conduct a power analysis for individual regression coefficients is the `paramtest` R package. Although it is a powerful package, it is a somewhat complicated package to use. The author of the package provides a tutorial here: https://cran.r-project.org/web/packages/paramtest/vignettes/Simulating-Power.html

There are actually online applications, built from R, that may be easier to use instead of writing out R code:

- https://designingexperiments.shinyapps.io/BUCSS_ss_power_reg1/ (an online app for the power of a single coefficient)
- https://designingexperiments.shinyapps.io/BUCSS_ss_power_reg_all/ (an online app for the power of R^2)

4 t-tests and one-way ANOVA

In class, we cover **one-sample t-tests**, **independent samples t-tests**, **paired t-tests**, **one-way ANOVAs**, and associated effect sizes (e.g., Cohen's d, eta squared, omega squared). All of these methods are concerned with testing equality of means either between two or more separate groups (independent-samples t-test, ANOVA), across two time points (paired t-test), or with respect to a given value (one-sample t-test).

In this section, the dataset that will be used comes from an article titled "Negativity on two sides: Individuals with borderline personality disorder form negative first impressions of others and are perceived negatively by them" (link here: https://doi.org/10.1037/per0000412, and data were found here: https://osf.io/tqbka/). The data include a diagnosis variable of raters (BPD, social anxiety disorder, or healthy control), a variable representing the diagnosis of the target being rated (BPD or healthy control), and a variable representing the degree to which the rater *trusts* a target (0 to 5).

4.1 SPSS

4.1.1 One-sample t-test

Here, we will conduct a one-sample t-test to determine whether the mean of trust ratings is significantly different from a value of zero.

First, one should check whether the assumption of normal residuals is not severely violated. As a matter of fact, unlike the regression models previously discussed, examining residuals for one-sample t-tests is equivalent to looking at the histogram of the raw values of the variable of interest. If we think of the one-sample t-test in terms of the general linear model, you can say that the one-sample t-test essentially is 'predicting' the outcome variable values with the variable's own mean (recall that this is the null model in our previously discussed regression models). The distribution of residuals resulting from this model will look exactly the same as the raw values of the outcome variable with just a different set of values on the x-axis (i.e., raw values will have the original units, and the residuals will be 'mean-centered' values). Therefore, to examine the normality assumption with the one-sample t-test, you could simply look at a histogram of the variable of interest.

To actually conduct the one-sample t-test:

- Analyze -> Compare Means -> One-Sample T-Test
- Put the `trust_overall` variable in the `Test Variable(s)` box
- Test value can be any value technically but for typical applications it is set at a value of zero
- Check off `Estimate effect sizes`
- Click `OK`

In the output, you should see descriptive statistics (mean, SD) for the variable of interest in the first table. The second table provides the actual significance test for the one-sample t-test. Finally, the third table provides effect sizes for the difference between the sample mean and the null mean.

4.1.2 Independent-samples t-test

Here, we will conduct an independent-samples t-test to determine whether the mean of trust ratings in those with BPD is significantly different than the mean of trust ratings in healthy controls.

For assumptions, one should check whether the assumption of normal residuals is not severely violated. Similar to the one-sample t-test, examining the residuals of this test is equivalent to looking at the histograms of the raw outcome variable of interest *within* each group separately. For example, in this case, you would look at the histogram/descriptives of `trust_overall` for those in the BPD group, and separately `trust_overall` for those in the healthy control group. The reason why this is equivalent to looking at the residuals of the model is because in an independent-samples t-test you are essentially using each group's mean to 'predict' the outcome variable, and as a result the residuals would look exactly the same as the stratified raw values with just different values on the x-axis. To get this information (and more):

- Analyze -> Explore
- Put the outcome variable of interest in the `Dependent List` box, and the grouping variable under `Factor List`
- Under `Plots`, click 'Histogram' and 'Normality plots with tests'; Click `Continue`
- Click `OK`

In the output, you should get group-wise descriptive statistics for the outcome variable, tests of normality, and plots of normality.

With respect to the constant variance assumption, it's recommended in the literature to simply not test for this assumption and look at the results for Welch's t-test, which does not assume equal variances between groups.

To actually conduct the independent-samples t-test:

- Analyze -> Compare Means -> Independent-Samples T-Test
- Put the `trust_overall` variable in the `Test Variable(s)` box (i.e., the variable you want to compare groups on)
- Put the `rater_group` variable in the `Grouping Variable` box (i.e., the variable representing the groups themselves)
- Check off `Estimate effect sizes`; Click `OK`

The output of the test should include three different tables. The first table provides descriptive statistics for the outcome variable of interest stratified by group. The second table provides the actual significance test. Importantly, the second row that says `Equal variances not assumed` is where the Welch's t-test is, and this should be focused on. The third table provides the associated effect sizes with the test.

To plot the results, you can go to Graphs -> Chart Builder, select a bar plot, drag the variables to their appropriate spots (i.e., grouping variable should be on the x-axis), and check off `Display error bars` on the side.

4.1.3 One-way ANOVA

Here, we will conduct a one-way ANOVA to determine whether the mean of trust ratings are significantly different between the BPD group, the social anxiety group, and the healthy controls.

The assumption of normality can be examined using the same procedure as in the independent-samples t-test. That is, one can go to Analyze -> Explore, put in the appropriate variables (i.e., outcome in Dependent List, grouping variable in Factor List), and under Plots select Histogram and Normality plots. With respect to the constant variance assumption, similar to what is recommended for independent-samples t-test, you can look at the Welch's one-way ANOVA which can be estimated using the steps below.

To run the one-way ANOVA:

- Analyze -> Compare Means -> One-Way ANOVA
- Put the appropriate variables in their respective boxes
- Check off `Estimate effect sizes for overall tests`
- Under `Post Hoc` you can select the post-hoc tests that you would like. Typical tests that are used are the LSD and Bonferroni (equal variances assumed), and Games-Howell (equal variances not assumed); Click `Continue`
- Under `Options`, select Descriptives, Homogeneity of variance test, Welch test, and Means plot; Click `Continue`
- Click `OK`

In the output, the first table provides the group-wise descriptive statistics for the outcome variable of interest. The second table provides Levene's test for constant variance, but as mentioned above, this can be ignored if you would like to just use the robust Welch's one-way ANOVA. The third table provides the overall F-test for the ANOVA. The fourth table provides effect sizes associated with the one-way ANOVA. The fifth table provides the robust Welch's one-way ANOVA, which is basically the robust version of the F-test found in

the thrid table. The last table provides the post hoc tests between the actual groups. A plot is also provided that shows the means for the outcome variable stratified by group. You can also plot the results using the same procedure as in the independent-samples t-test.

4.1.4 Paired t-test

Here, we will conduct a paired t-test on some other variables in the dataset. In the dataset, raters were actually providing trust ratings of people who belonged to one of two groups: either they had a diagnosis of BPD or they were a healthy control. Because the same rater provided ratings for those in the BPD group and those in the healthy control group, this can be considered paired (or within-person) data, such that the participants provided ratings for two different conditions. As mentioned in the lecture, SPSS requires that the data are in wide format, such that each condition has its own column in the dataset and each participant only has one row.

To run the paired t-test:

- Analyze -> Compare Means -> Paired-Samples T-Test
- Drag the two condition variables into the box
- Check off `Estimate effect sizes`
- Click `OK`

In the output, the first table provides the by-condition descriptive statistics for the outcome variable of interest. The second table provides the correlation between the two condition's ratings. The third table provides the actual statistical test. The last table provides the associated effect size. To plot the results, follow the steps provided in the slides for adjusting the scores so that the error bars are correct.

4.2 R

The data are initially in long format (i.e., one row per trial, where one participant will have multiple rows/trials) and I want to convert it into wide format to do t-tests and the one-way ANOVA. Wide format is almost the opposite of long format, where each participant only has one row but each column refers to a different time point on a given variable.

```
library(tidymodels)

# Importing data
df <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/rater_target_data.csv",
    header = T
)

head(df) # First couple of values for what the data initially look like
```

```
##      rater   target rater_group target_group dg_given_est similar trust know
## 1 14011070 14011002         BPD   bpd_target          2.5       1     4    1
## 2 14011070 14011007         BPD   bpd_target          1.5       0     2    0
## 3 14011070 14010001         BPD    hc_target          2.0       0     3    0
## 4 14011070 14011004         BPD   bpd_target          2.5       0     4    0
## 5 14011070 14010005         BPD    hc_target          3.0       0     5    1
## 6 14011070 14011906         BPD   bpd_target          1.0       0     1    0
##   comorb_no any_mood bpd_target hc_target
## 1         1        1          1         0
## 2         1        1          1         0
## 3         1        1          0         1
## 4         1        1          1         0
## 5         1        1          0         1
## 6         1        1          1         0
```

```
# Here, I'm creating a mean trust score per participant (i.e., 'rater')
df <- df %>%
    group_by(rater) %>%
    mutate(trust_overall = mean(trust, na.rm = T)) %>%
    ungroup()

head(df) # Now there's a mean score for each rater
```

```
## # A tibble: 6 x 13
##      rater   target rater_group target_group dg_given_est similar trust  know
##      <int>    <int> <chr>       <chr>               <dbl>   <int> <int> <int>
## 1 14011070 14011002 BPD         bpd_target            2.5       1     4     1
## 2 14011070 14011007 BPD         bpd_target            1.5       0     2     0
## 3 14011070 14010001 BPD         hc_target             2         0     3     0
## 4 14011070 14011004 BPD         bpd_target            2.5       0     4     0
## 5 14011070 14010005 BPD         hc_target             3         0     5     1
## 6 14011070 14011906 BPD         bpd_target            1         0     1     0
## # ... with 5 more variables: comorb_no <int>, any_mood <int>, bpd_target <int>,
## #   hc_target <int>, trust_overall <dbl>
```

```
# Here, I'm creating a mean trust score per rater by the type of
# target they are rating (i.e., either BPD or healthy control)
df <- df %>%
    group_by(rater, target_group) %>%
    mutate(trust_group = mean(trust, na.rm = T)) %>%
```

```
    ungroup()

head(df)
```

```
## # A tibble: 6 x 14
##      rater   target rater_group target_group dg_given_est similar trust  know
##      <int>    <int> <chr>       <chr>               <dbl>   <int> <int> <int>
## 1 14011070 14011002 BPD         bpd_target            2.5       1     4     1
## 2 14011070 14011007 BPD         bpd_target            1.5       0     2     0
## 3 14011070 14010001 BPD         hc_target             2         0     3     0
## 4 14011070 14011004 BPD         bpd_target            2.5       0     4     0
## 5 14011070 14010005 BPD         hc_target             3         0     5     1
## 6 14011070 14011906 BPD         bpd_target            1         0     1     0
## # ... with 6 more variables: comorb_no <int>, any_mood <int>, bpd_target <int>,
## #   hc_target <int>, trust_overall <dbl>, trust_group <dbl>
```

```
# Finally converting the data to wide format
# Pivot_wider makes new columns in wide format, where the new column names
# come from categorical values of 'target_group', the actual values in the
# new column come from the 'trust_group' variable
df_wide <- df %>%
    dplyr::select(rater, rater_group, target_group, trust_overall, trust_group) %>%
    pivot_wider(
        names_from = target_group, values_from = trust_group,
        values_fn = list(trust_group = mean)
    )

# Data preview
head(df_wide)
```

```
## # A tibble: 6 x 5
##      rater rater_group trust_overall bpd_target hc_target
##      <int> <chr>               <dbl>      <dbl>     <dbl>
## 1 14011070 BPD                 2.73       2.81      2.65
## 2 14011071 BPD                 1.69       1.69      1.69
## 3 14011072 BPD                 2.04       1.46      2.62
## 4 14011073 BPD                 0.788      0.654     0.923
## 5 14011074 BPD                 2.04       2         2.08
## 6 14011076 BPD                 2.65       2.65      2.65
```

```
# Here, I'm making a modified version of the long format in order to run the
# subsequent paired t-test. That is, the paired t-test requires long format
df_long <- df %>%
    dplyr::select(rater, rater_group, target_group, trust_overall, trust_group) %>%
    distinct()

head(df_long) # Data preview
```

```
## # A tibble: 6 x 5
##      rater rater_group target_group trust_overall trust_group
##      <int> <chr>       <chr>               <dbl>       <dbl>
## 1 14011070 BPD         bpd_target           2.73        2.81
## 2 14011070 BPD         hc_target            2.73        2.65
## 3 14011071 BPD         hc_target            1.69        1.69
## 4 14011071 BPD         bpd_target           1.69        1.69
```

```
## 5 14011072 BPD         hc_target             2.04        2.62
## 6 14011072 BPD         bpd_target            2.04        1.46
```

4.2.1 One-sample t-test

A one-sample t-test compares the mean value of a given variable against a null value of 0. In the following code, I want to see whether the mean trust score given by raters is significantly different from zero. The `~1` means that I am just estimating an intercept (i.e., the mean value of the variable on the left of the `~`) and doing a hypothesis test on that value.

We can also calculate an effect size associated with this test. Cohen's d is usually used when comparing means and represents how far one mean is from another in terms of standard deviations (can think of the null mean as having a distribution and seeing where the estimated mean falls relative to this distribution). In this case, we are comparing the mean value of the trust variable with a mean value of zero. The `effectsize` package is nice for calculating effect sizes, including common langauge effect sizes.

```
t.test(trust_overall ~ 1, data = df_wide)
```

```
##
##  One Sample t-test
##
## data:  trust_overall
## t = 29.82, df = 97, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  2.413991 2.758241
## sample estimates:
## mean of x
##  2.586116
```

```
library(effectsize)
cohens_d(trust_overall ~ 1, data = df_wide)
```

```
## Cohen's d |       95% CI
## ------------------------
## 3.01      | [2.54, 3.48]
```

```
p_superiority(df_wide$trust_overall, mu = 0)
```

```
## Pr(superiority) |       95% CI
## -------------------------------
## 0.98            | [0.96, 0.99]
```

4.2.2 Independent-samples t-test

An independent-samples t-test compares the means of two independent groups on some outcome. In the following code, I want to see whether the mean of trust ratings given by BPD raters is significantly different from the mean of trust scores given by healthy control raters. Similar to the above, we can calculate Cohen's d for the effect size between the two groups, along with other common language effect sizes discussed in class.

The `t.test` function initially gives an error because the rater_group variable has more than 2 levels (recall that the study was interested in comparing BPD, healthy controls, *and* social anxiety). We need to create a subset of the full dataset with *only* the BPD and healthy control raters. Also note that the `t.test` function automatically calculates Welch's t-test (i.e., does not assume equal variances).

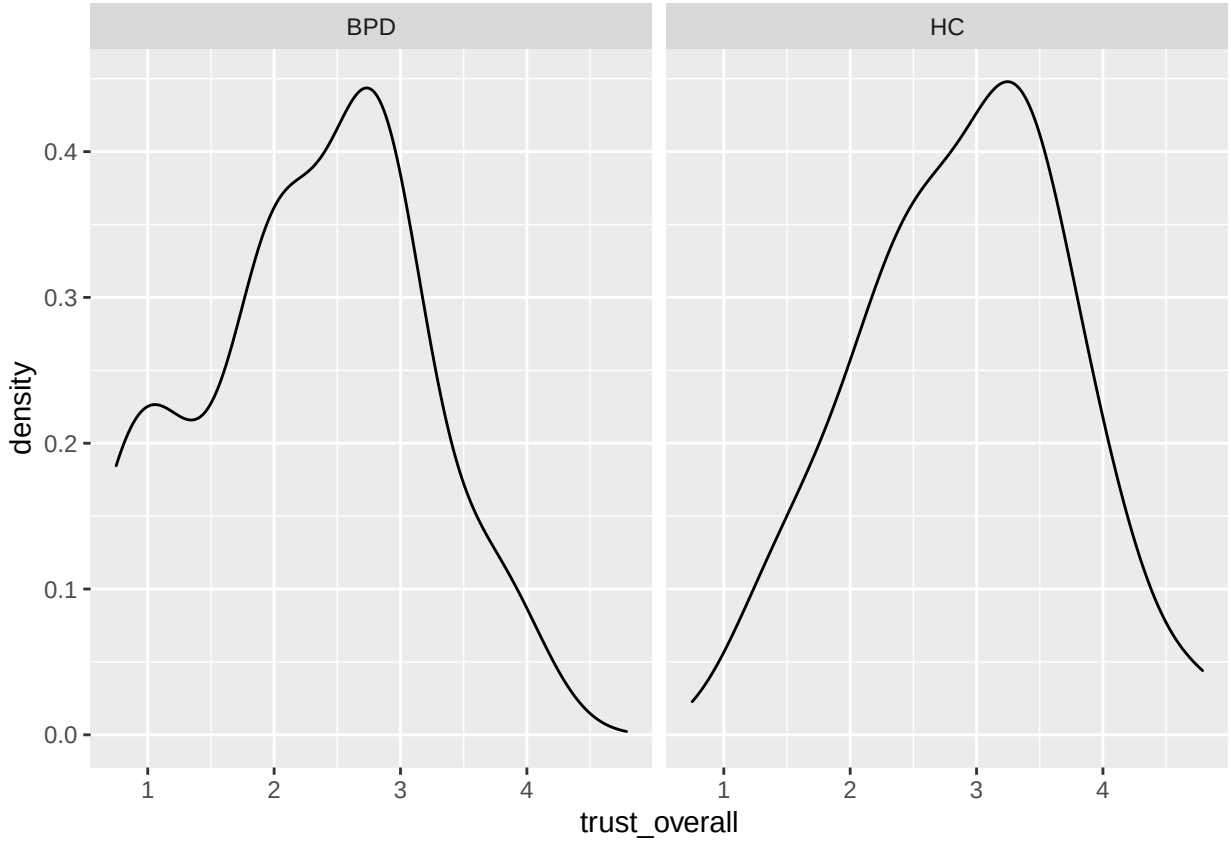
Before I run the t-test, I examine whether the assumption of normality is severely violated by looking at group-wise descriptives for the outcome variable of interest using: `describeBy()` to look at skew and kurtosis

values, and plotting separate density plots for each group using ggplot

```
df_ttest <- df_wide %>%
    filter(!rater_group == "SP") # Filtering out social anxiety participants

# Looking at normality
library(psych)
describeBy(df_ttest$trust_overall, group = df_ttest$rater_group)
```

```
##
##  Descriptive statistics by group
## group: BPD
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 32 2.27 0.88   2.32    2.28 0.76 0.75 3.98  3.23 -0.13    -0.89 0.16
## ------------------------------------------------------------
## group: HC
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 37 2.91 0.82   2.98    2.92 0.71 1.19 4.79   3.6 -0.08    -0.54 0.14
```

```
df_ttest %>%
    ggplot(aes(x = trust_overall)) + # Variable of interest
    geom_density() + # Density plots
    facet_wrap(~rater_group) # The grouping variable
```




```
# Using the t.test() function
t.test(trust_overall ~ rater_group, data = df_ttest)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  trust_overall by rater_group
## t = -3.0738, df = 64.072, p-value = 0.003103
## alternative hypothesis: true difference in means between group BPD and group HC is not equal to 0
## 95 percent confidence interval:
##  -1.0434930 -0.2214208
## sample estimates:
## mean in group BPD  mean in group HC
##          2.274248          2.906705
```

```
cohens_d(trust_overall ~ rater_group, data = df_ttest)
```

```
## Cohen's d |         95% CI
## -------------------------
## -0.75     | [-1.23, -0.25]
##
## - Estimated using pooled SD.
```

```
# Below shows that a randomly chosen BPD participant has a 30% chance of
# having a larger trust score than a healthy control
p_superiority(trust_overall ~ rater_group, data = df_ttest)
```

```
## Pr(superiority) |       95% CI
## -------------------------------
## 0.30            | [0.19, 0.43]
```

```
# Below shows that the two distributions of trust scores between BPD
# and healthy controls overlap 71%
p_overlap(trust_overall ~ rater_group, data = df_ttest)
```

```
## Overlap |       95% CI
## ----------------------
## 0.71    | [0.54, 0.90]
```

```
# Below shows that only 23% of BPD participants have trust scores
# larger than the healthy control's average
cohens_u3(trust_overall ~ rater_group, data = df_ttest)
```

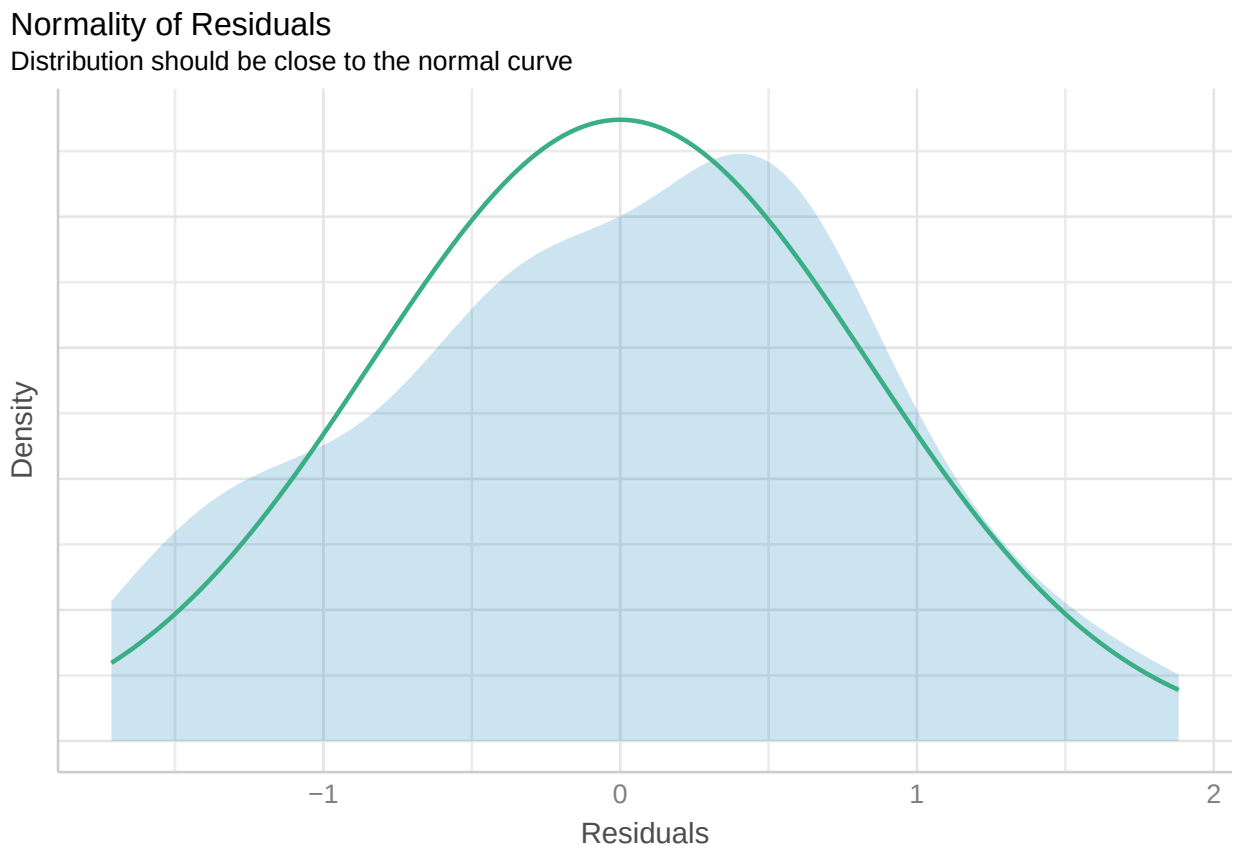
```
## Cohen's U3 |       95% CI
## -------------------------
## 0.23       | [0.11, 0.40]
```

```
# Using the lm() function for glm analogy
library(performance)
model1 <- lm(trust_overall ~ rater_group, data = df_ttest)
summary(model1) # difference between BPD and HC = .63
```

```
##
## Call:
## lm(formula = trust_overall ~ rater_group, data = df_ttest)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.71440 -0.57978  0.07406  0.53560  1.88176
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.2742     0.1499  15.168  < 2e-16 ***
```

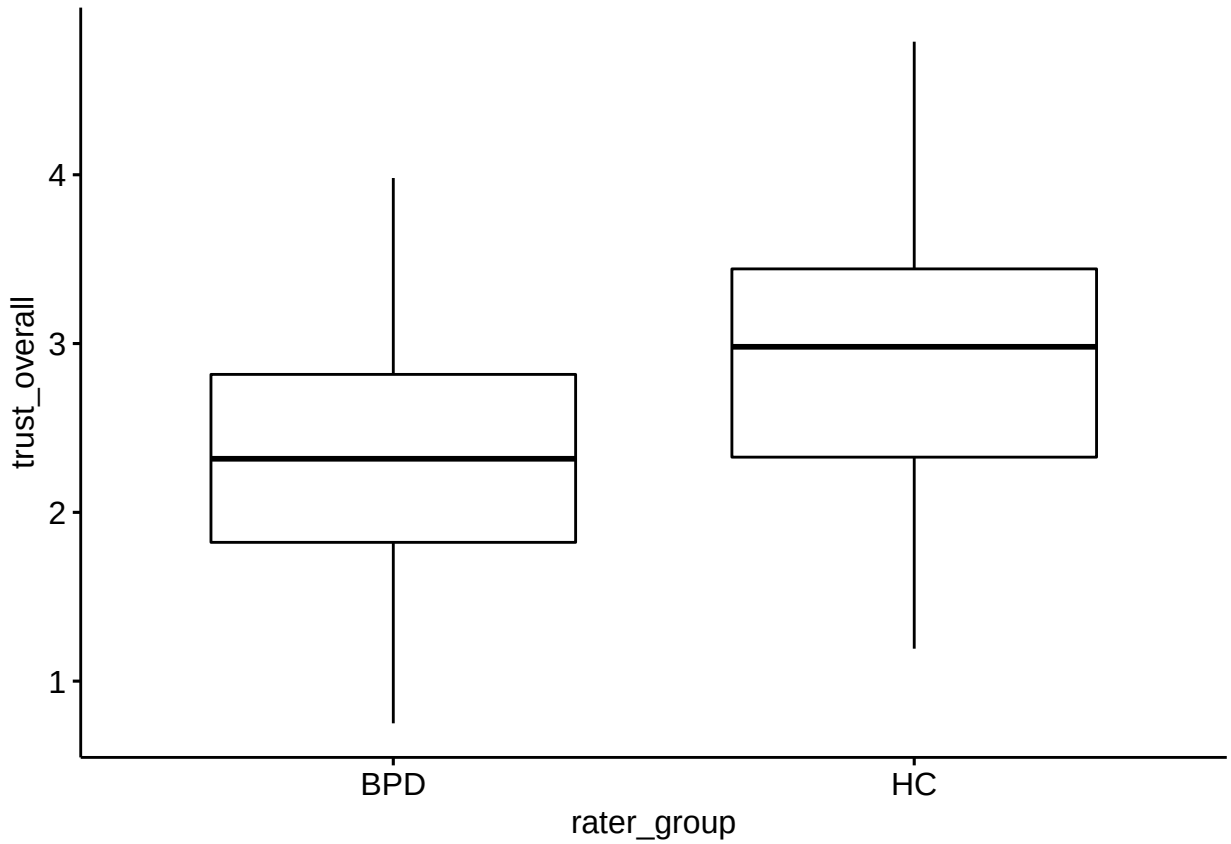
```
## rater_groupHC   0.6325     0.2047   3.089  0.00292 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8481 on 67 degrees of freedom
## Multiple R-squared:  0.1247, Adjusted R-squared:  0.1116
## F-statistic: 9.542 on 1 and 67 DF,  p-value: 0.002923
```

```
check_normality(model1) %>% plot()
```

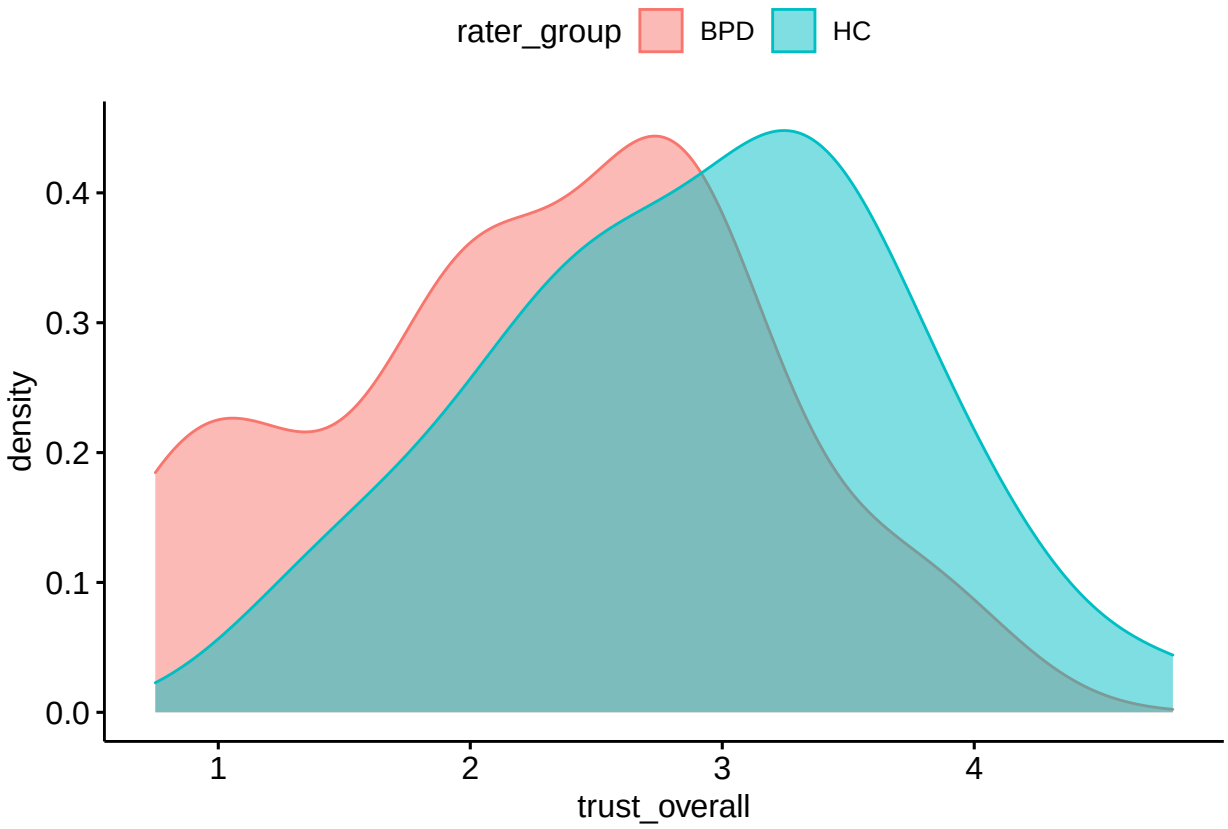


We can get a plot of the results using the `ggpubr` package.

```
library(ggpubr)

# Box plot
ggboxplot(df_ttest, x = "rater_group", y = "trust_overall")
```




```
# Density plot
ggdensity(df_ttest, x = "trust_overall", color = "rater_group", fill = "rater_group")
```



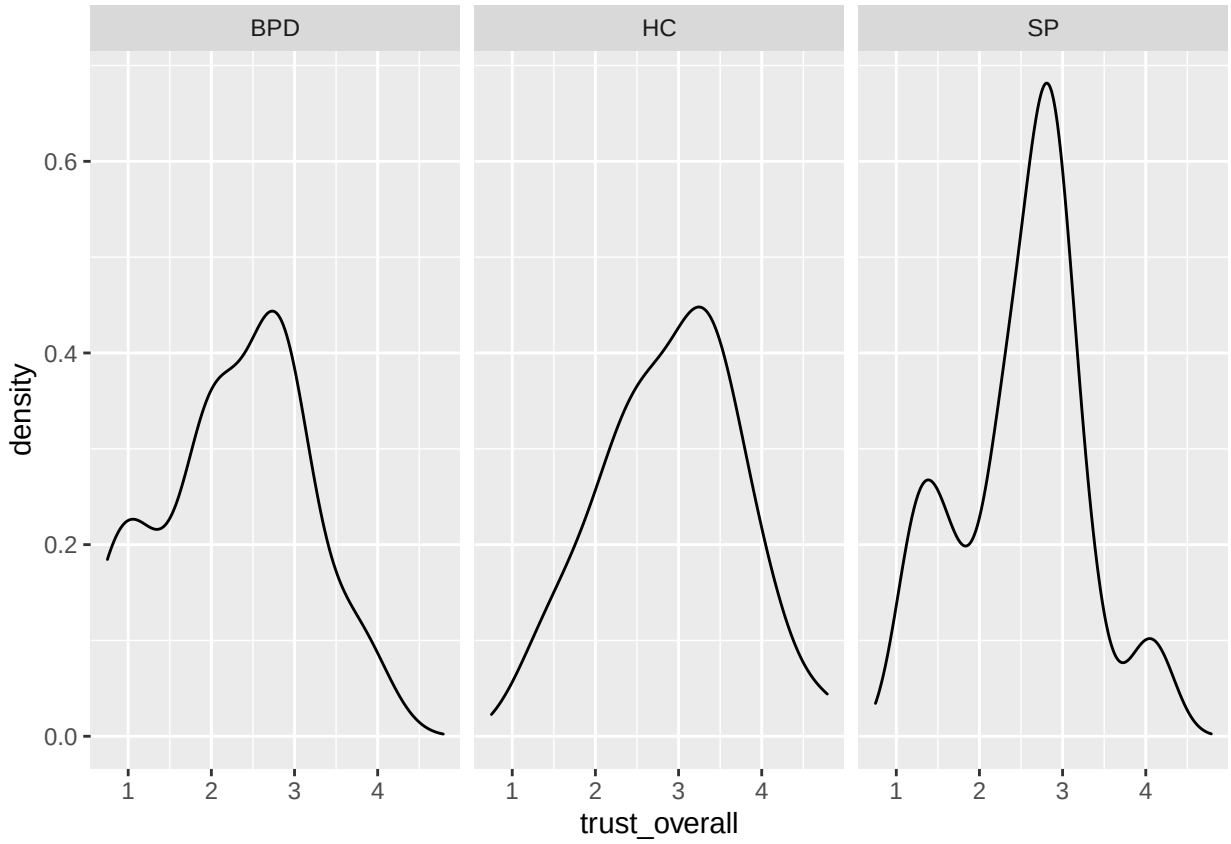
4.2.3 One-way ANOVA

Here, I want to now see whether the mean of trust ratings differs among the three diagnostic groups - BPD, healthy control, and social anxiety. First, the normality assumption is examined through descriptive statistics and plots. Second, Welch's ANOVA is run for the global F-test. Third, a one-way ANOVA is run using `lm()` to show how the ANOVA is under the GLM, and I also estimate various effect sizes for the whole model. Lastly, I look at post-hoc tests between the groups to see which particular groups are significantly different from each other.

```
library(modelbased)

# Checking normality
describeBy(df_wide$trust_overall, group = df_wide$rater_group)
```

```
##
##  Descriptive statistics by group
## group: BPD
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 32 2.27 0.88   2.32    2.28 0.76 0.75 3.98  3.23 -0.13    -0.89 0.16
## ------------------------------------------------------------
## group: HC
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 37 2.91 0.82   2.98    2.92 0.71 1.19 4.79   3.6 -0.08    -0.54 0.14
## ------------------------------------------------------------
## group: SP
##    vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
## X1    1 29 2.52 0.76   2.71    2.51 0.57 1.12 4.17  3.06 -0.1    -0.46 0.14
```

```
df_wide %>%
    ggplot(aes(x = trust_overall)) +
    geom_density() +
    facet_wrap(~rater_group)
```




```
# Welch's ANOVA
oneway.test(trust_overall ~ rater_group, data = df_wide)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  trust_overall and rater_group
## F = 4.8821, num df = 2.000, denom df = 62.164, p-value = 0.01073
```

```
# One-way ANOVA using lm() + effect sizes
model2 <- lm(trust_overall ~ rater_group, data = df_wide)
eta_squared(model2)
```

```
## # Effect Size for ANOVA
##
## Parameter   | Eta2 |       95% CI
## ---------------------------------
## rater_group | 0.10 | [0.02, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
omega_squared(model2)
```

```
## # Effect Size for ANOVA
```

```
##
## Parameter   | Omega2 |       95% CI
## ----------------------------------
## rater_group |   0.08 | [0.01, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
epsilon_squared(model2)
```

```
## # Effect Size for ANOVA
##
## Parameter   | Epsilon2 |       95% CI
## ------------------------------------
## rater_group |     0.08 | [0.01, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

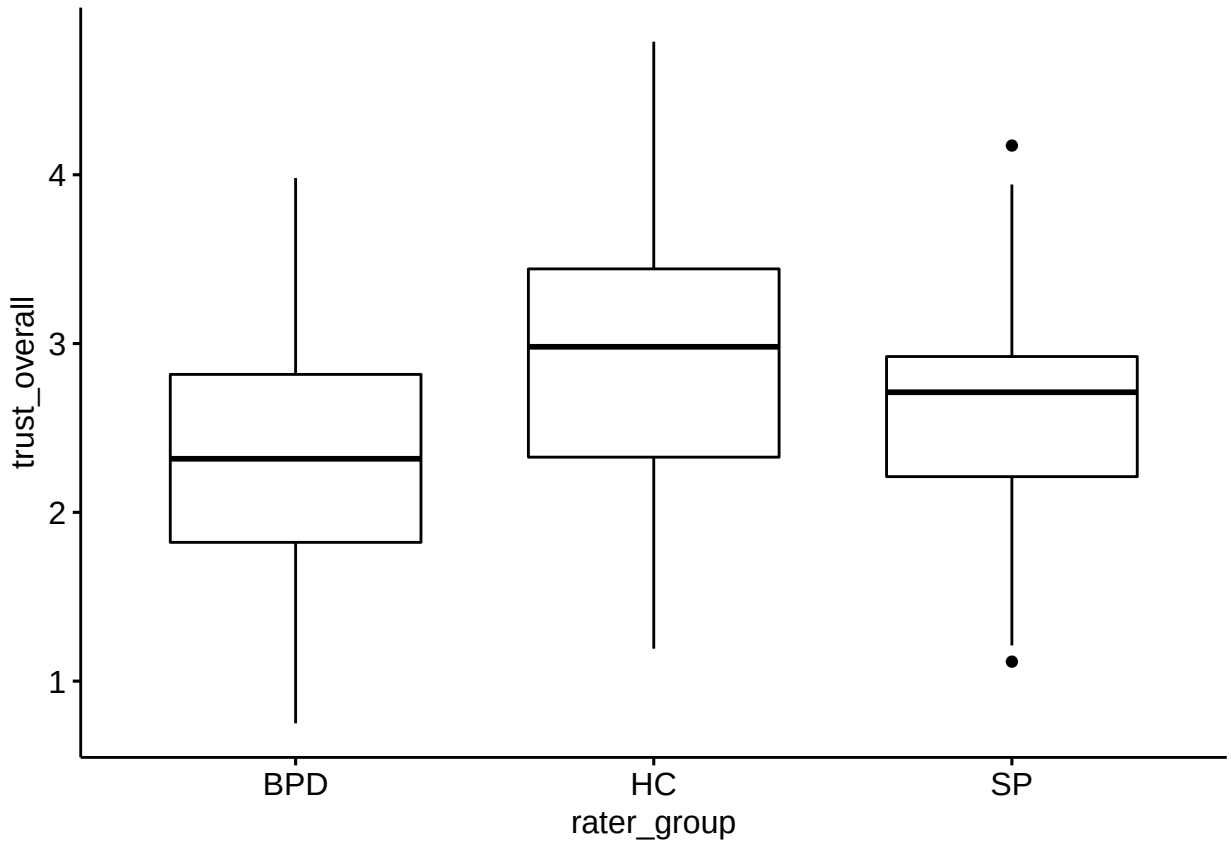
```
# Post-hoc tests
estimate_means(model2)
```

```
## Estimated Marginal Means
##
## rater_group | Mean |   SE |       95% CI
## ----------------------------------------
## BPD         | 2.27 | 0.15 | [1.99, 2.56]
## HC          | 2.91 | 0.14 | [2.64, 3.18]
## SP          | 2.52 | 0.15 | [2.22, 2.82]
##
## Marginal means estimated at rater_group
```

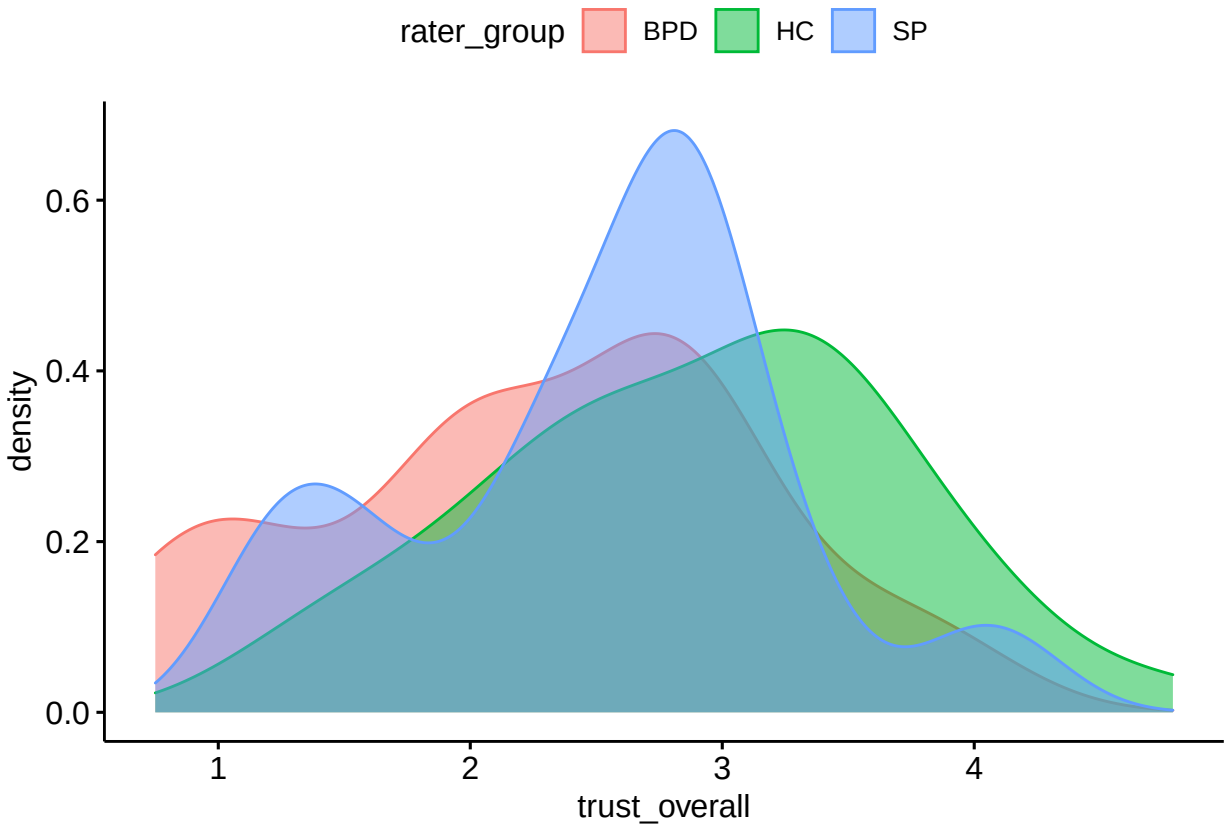
```
estimate_contrasts(model2)
```

```
## Marginal Contrasts Analysis
##
## Level1 | Level2 | Difference |         95% CI |   SE | t(95) |     p
## ------------------------------------------------------------------------
## BPD    |     HC |      -0.63 | [-1.12, -0.15] | 0.20 | -3.18 | 0.006
## BPD    |     SP |      -0.25 | [-0.76,  0.27] | 0.21 | -1.17 | 0.474
## HC     |     SP |       0.39 | [-0.11,  0.88] | 0.20 |  1.89 | 0.148
##
## Marginal contrasts estimated at rater_group
## p-value adjustment method: Holm (1979)
```

```
# Plots
ggboxplot(df_wide, x = "rater_group", y = "trust_overall")
```




```
ggdensity(df_wide, x = "trust_overall", color = "rater_group", fill = "rater_group")
```



4.2.4 Paired t-test

In the example dataset that is being used, raters actually provided trust ratings after observing someone who was either diagnosed with BPD or was a healthy control, though the raters didn't have information regarding diagnoses at all. One can think of this as a within-person design because the same rater provided trust ratings for two different groups of people (or in other words, across two conditions). Therefore, it might be of interest to test whether trust ratings were significantly different depending on whether the person being rated was diagnosed with BPD or not.

In R, the data must be in long format, where each row corresponds to an observation within a participant, rather than each row corresponding to every observation on that participant. Trial-level data is usually structured this way, where each trial has its own row in the dataset, but as multiple trials are administered to each participant, multiple rows are associated with a particular participant. Note that this is different to what is required in SPSS, where SPSS requires that the data are in wide format.

```
# Normality assumption
describeBy(df_long$trust_group, group = df_long$target_group)
```

```
##
##  Descriptive statistics by group
## group: bpd_target
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 98 2.42 0.86   2.54    2.43 0.86 0.46 4.69  4.23 -0.07    -0.41 0.09
## ------------------------------------------------------------
## group: hc_target
##    vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
## X1    1 98 2.75 0.91   2.81    2.77 0.88 0.85 4.88  4.04 -0.19    -0.47 0.09
```

```r
# Actual test and effect size
t.test(trust_group ~ target_group, data = df_long, paired = T)
```

```
##
##  Paired t-test
##
## data:  trust_group by target_group
## t = -7.515, df = 97, p-value = 2.837e-11
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.4146773 -0.2414047
## sample estimates:
## mean of the differences
##                -0.328041
```

```r
p_superiority(trust_group ~ target_group, data = df_long, paired = T)
```

```
## Pr(superiority) |       95% CI
## -------------------------------
## 0.30            | [0.24, 0.35]
```

```r
cohens_d(trust_group ~ target_group, data = df_long, paired = T)
```

```
## Cohen's d |         95% CI
## --------------------------
## -0.76     | [-0.98, -0.53]
```

4.3 Power Analysis Resources

G*Power can be used to conduct power analyses for independent-samples t-tests and paired t-tests (see the links provided in the Regression section).

In R, power analyses can be conducted using the `pwr` package. The `pwr.t.test` function is used for estimating the sample size for t-tests, and the `pwr.anova.test` is used for one-way ANOVAs. The following code provides an example of a power analysis done for a t-test, where I would like to determine the sample size needed to detect a Cohen's d of .50, given 80% statistical power. The output of the code tells me that I would need to collect 64 people (in each group) in order to detect a standardized difference of .50.

```
library(pwr)
pwr.t.test(d = .50, sig.level = .05, power = .80)
```

```
##
##      Two-sample t test power calculation
##
##               n = 63.76561
##               d = 0.5
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

There are a few other online resources that can be used for doing a power analysis for t-tests and one-way ANOVAs:

- https://designingexperiments.shinyapps.io/power_ttest2group/ (an online app for independent-samples t-test)
- https://designingexperiments.shinyapps.io/BUCSS_ss_power_it/ (another online app)
- https://designingexperiments.shinyapps.io/BUCSS_ss_power_dt/ (for paired t-tests)
- https://designingexperiments.shinyapps.io/BUCSS_ss_power_ba/ (for one-way ANOVAs)
- https://journals.sagepub.com/doi/full/10.1177/2515245920951503 (another app for ANOVAs)
- https://www.taylorfrancis.com/books/mono/10.4324/9781315171500/applied-power-analysis-behavioral-sciences-christopher-aberson

5 ANCOVA

In this section, the dataset that will be used comes from an article titled "Changing personality traits with the help of a digital personality change intervention" (link here: https://doi.org/10.1073/pnas.2017548118, and data were found here: https://osf.io/g3yfz/). The data include a variable representing whether participants received the intervention (personality change intervention; coded as 1 in the data) or received waitlist control (coded as 0 in the data), and two variables representing pre- and post- intervention Neuroticism scores.

5.1 SPSS

For instructions on how to examine ANCOVA assumptions and estimate an actual ANCOVA model, see the second half of the Class 7 slides.

5.2 R

The following code demonstrates multiple steps for estimating an ANCOVA model. First, the linearity assumption (i.e., a linear relation between the covariate and the DV within both groups) is assessed using graphical methods from the `ggplot2` package. Second, the homogeneity of slopes assumption is also assessed using graphical methods, and by estimating an initial ANCOVA model where there is an interaction between the covariate and the grouping variable (in order to see if the interaction is significant), using the `lm()` function. Third, the actual ANCOVA model is estimated using the `lm()` function, and the main results are provided by the `summary()` and `Anova()` function. Effect sizes were also estimated from this model, along with examining other model assumptions using the `check_model()` function, and along with post hoc tests for mean differences between the two groups, after controlling for the covariate. The use of the `jmv` package is also demonstrated for estimating ANCOVA models.

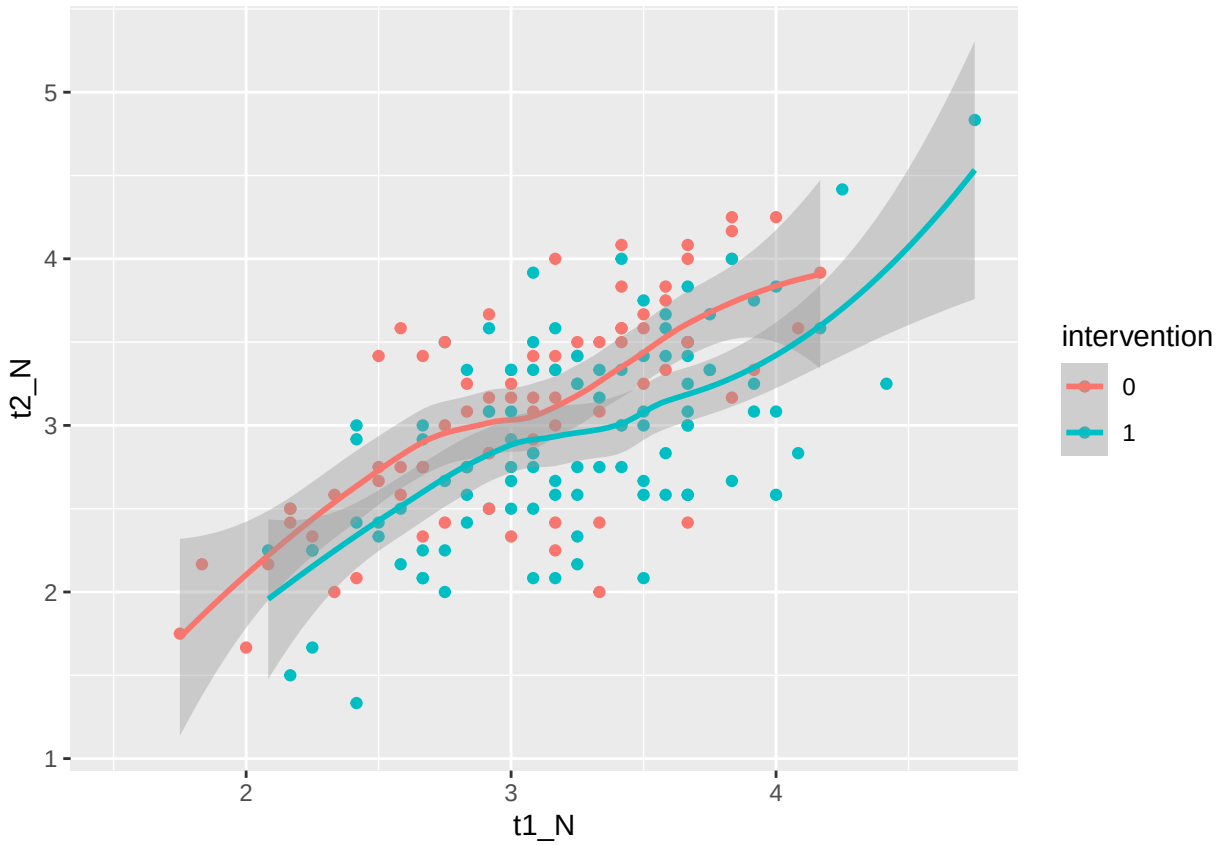
```
library(car)
library(modelbased)
library(effectsize)
library(performance)

# Import data
df_wide <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/Class 7 ANCOVA Data.csv",
                    header = T)

# Preview data
head(df_wide)
```

```
##   X    id intervention     t1_N     t2_N
## 1 1 p0005            0       NA       NA
## 2 2 p0006            1 3.916667       NA
## 3 3 p0009            1 3.500000       NA
## 4 4 p0012            0 3.250000 3.250000
## 5 5 p0021            1 4.750000 4.833333
## 6 6 p0034            0 3.416667 3.833333
```

```
# Making sure the group variable is indeed categorical
df_wide$intervention <- as.factor(df_wide$intervention)

# Assessing linearity assumption (in both groups)
ggplot(df_wide, aes(x = t1_N, y = t2_N)) +
    geom_point(aes(color = intervention)) +
    geom_smooth(aes(color = intervention))
```

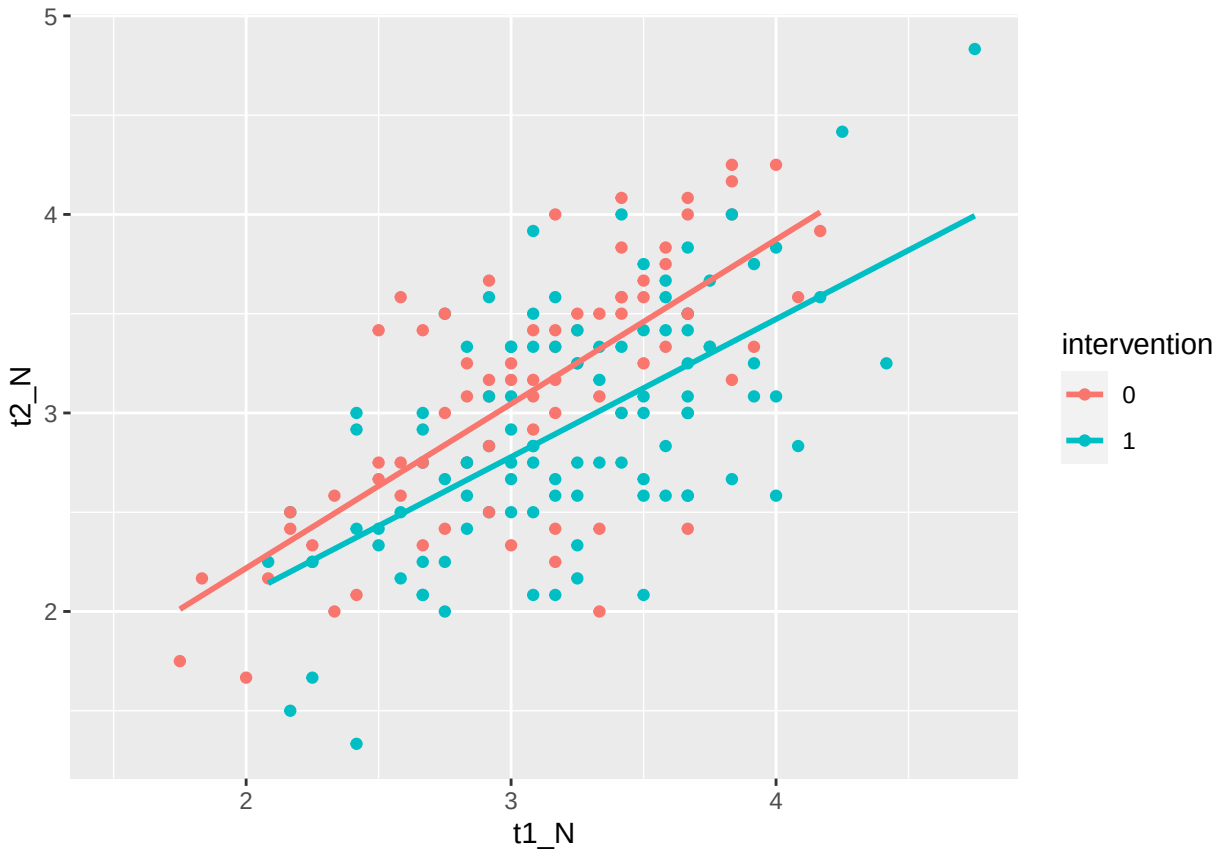
```
## Warning: Removed 227 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 227 rows containing missing values (geom_point).
```




```
# Assessing homogeneity of slopes assumption
ggplot(df_wide, aes(x = t1_N, y = t2_N)) +
    geom_point(aes(color = intervention)) +
    geom_smooth(aes(color = intervention), method = "lm", se = F)
```

```
## Warning: Removed 227 rows containing non-finite values (stat_smooth).
## Removed 227 rows containing missing values (geom_point).
```




```
assumptions <- lm(t2_N ~ intervention * t1_N, data = df_wide)
summary(assumptions)
```

```
##
## Call:
## lm(formula = t2_N ~ intervention * t1_N, data = df_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.32162 -0.30386  0.03429  0.30103  1.08043
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.5633     0.3076   1.831   0.0688 .
## intervention1        0.1324     0.4133   0.320   0.7491
## t1_N                 0.8275     0.0986   8.392 1.56e-14 ***
## intervention1:t1_N  -0.1333     0.1300  -1.025   0.3068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4631 on 175 degrees of freedom
##   (227 observations deleted due to missingness)
## Multiple R-squared:  0.4516, Adjusted R-squared:  0.4422
## F-statistic: 48.05 on 3 and 175 DF,  p-value: < 2.2e-16
```

```
estimate_slopes(assumptions, at = "intervention")
```

```
## Estimated Marginal Effects
```

```
##
## intervention | Coefficient |   SE |       95% CI | t(175) |      p
## ------------------------------------------------------------------
## 0            |        0.83 | 0.10 | [0.63, 1.02] |   8.39 | < .001
## 1            |        0.69 | 0.08 | [0.53, 0.86] |   8.19 | < .001
## Marginal effects estimated for t1_N
```

```
# Estimating ANCOVA model
model <- lm(t2_N ~ intervention + t1_N, data = df_wide)
summary(model)
```

```
##
## Call:
## lm(formula = t2_N ~ intervention + t1_N, data = df_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30161 -0.28417  0.01096  0.30940  1.08765
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.79874    0.20463   3.903 0.000135 ***
## intervention1 -0.28488    0.07073  -4.028 8.37e-05 ***
## t1_N           0.75086    0.06429  11.679  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4632 on 176 degrees of freedom
##   (227 observations deleted due to missingness)
## Multiple R-squared:  0.4484, Adjusted R-squared:  0.4421
## F-statistic: 71.52 on 2 and 176 DF,  p-value: < 2.2e-16
```

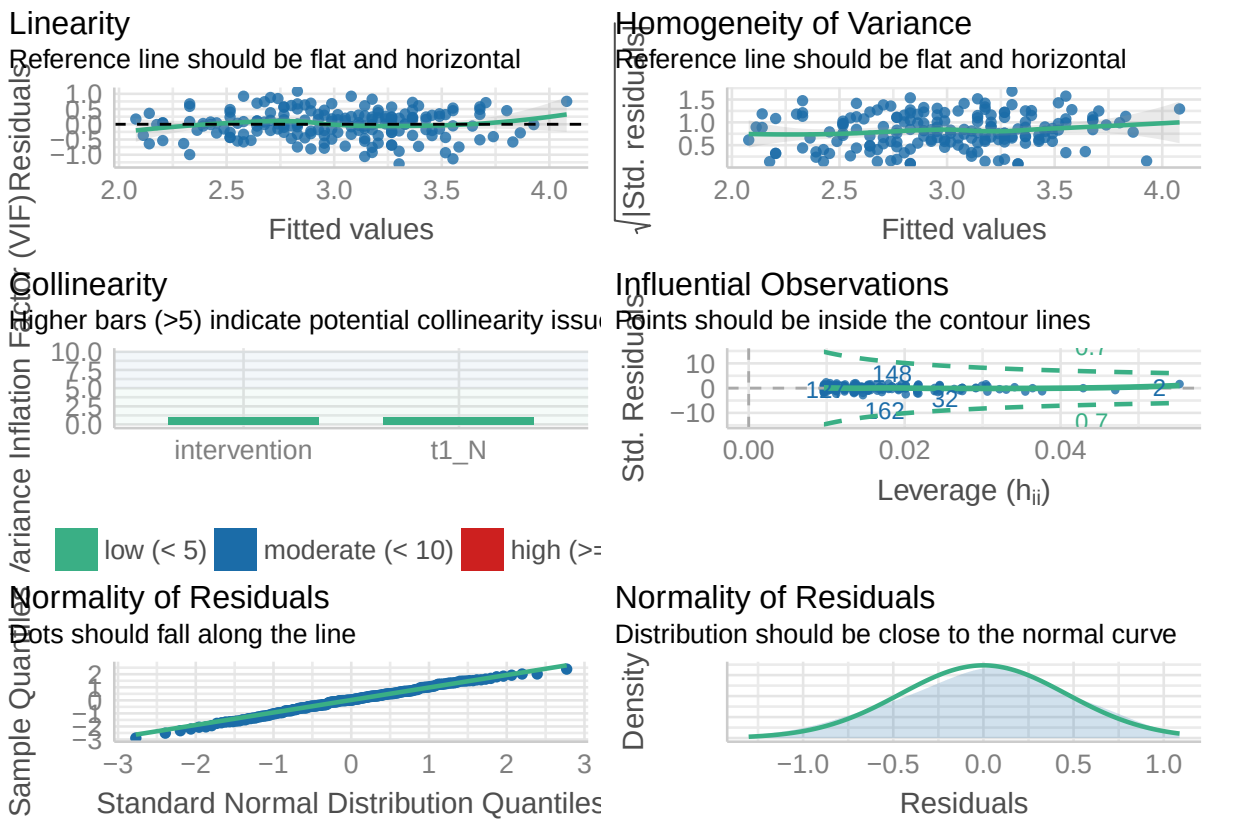
```
Anova(model)
```

```
## Anova Table (Type II tests)
##
## Response: t2_N
##              Sum Sq  Df F value    Pr(>F)
## intervention  3.480   1  16.222 8.367e-05 ***
## t1_N         29.265   1 136.400 < 2.2e-16 ***
## Residuals    37.761 176
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Effect sizes
eta_squared(model, partial = T)
```

```
## # Effect Size for ANOVA (Type I)
##
## Parameter    | Eta2 (partial) |       95% CI
## ---------------------------------------------
## intervention |           0.04 | [0.00, 1.00]
## t1_N         |           0.44 | [0.35, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
# Assessing other assumptions
check_model(model)
```




```
# Post hoc tests
estimate_means(model) # Actual mean values by group
```

```
## Estimated Marginal Means
##
## intervention | Mean |   SE |       95% CI
## ----------------------------------------
## 0            | 3.17 | 0.05 | [3.06, 3.27]
## 1            | 2.88 | 0.05 | [2.79, 2.97]
##
## Marginal means estimated at intervention
```

```
estimate_contrasts(model) # Testing significance of group mean differences
```

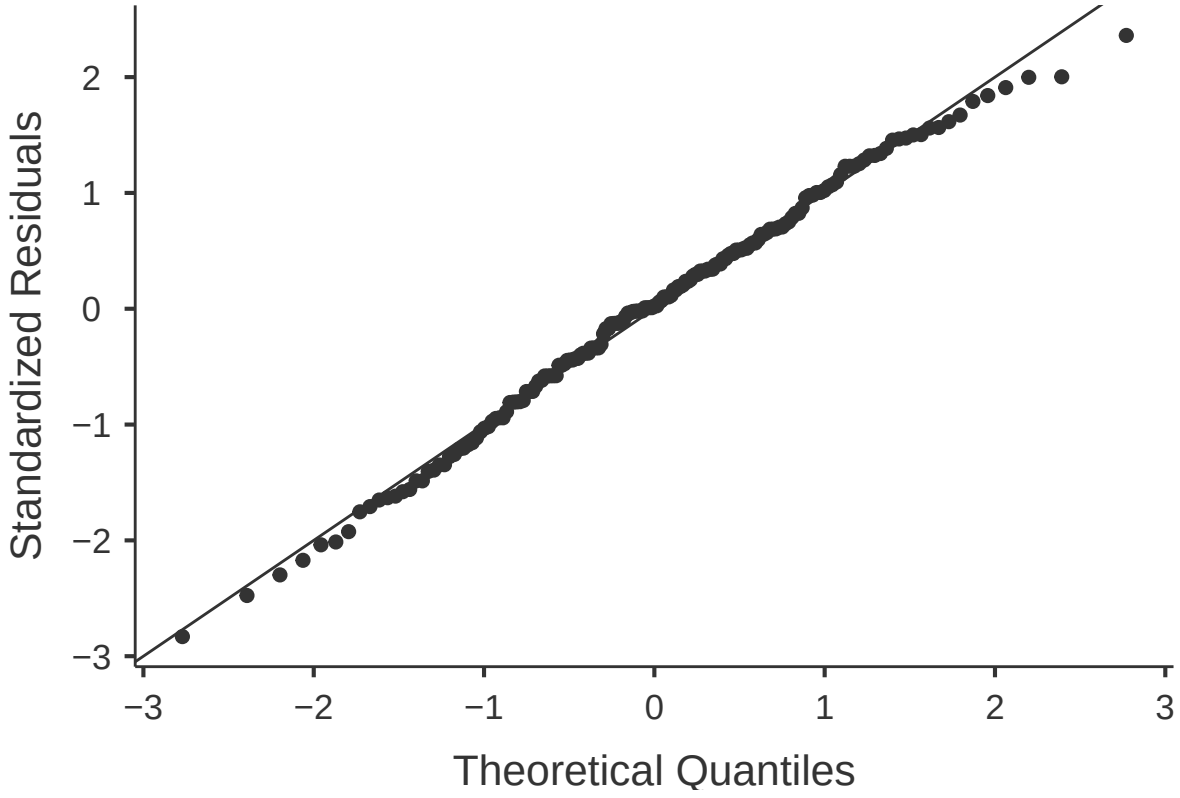
```
## Marginal Contrasts Analysis
##
## Level1 | Level2 | Difference |       95% CI |   SE | t(176) |      p
## ------------------------------------------------------------------------
## 0      |      1 |       0.28 | [0.15, 0.42] | 0.07 |   4.03 | < .001
##
## Marginal contrasts estimated at intervention
## p-value adjustment method: Holm (1979)
```

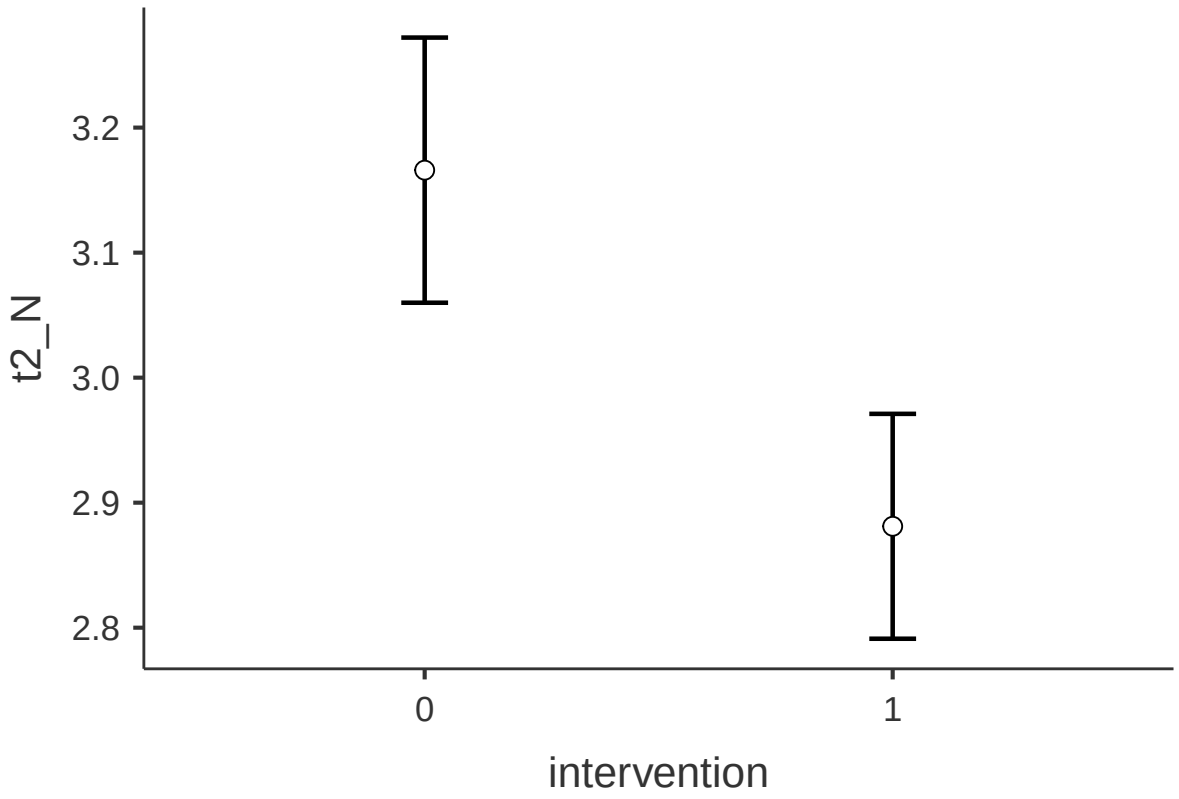
```
# Other way to estimate ANCOVA models
library(jmv)
ancova(
```

```
    data = df_wide,
    dep = t2_N,
    factors = intervention,
    covs = t1_N,
    effectSize = "partEta",
    homo = T, # Check for constant variance assumption
    norm = T, # Check for normality
    qq = T,
    postHoc = ~intervention, # Grouping variable for post hoc
    emMeans = ~intervention,
    emmTables = T
)
```

```
##
##  ANCOVA
##
##  ANCOVA - t2_N
##
##                    Sum of Squares    df     Mean Square    F            p              ²p
##
##    intervention          3.480417      1      3.4804168     16.22185     0.0000837    0.0843913
##    t1_N                 29.264733      1     29.2647329    136.39981    < .0000001    0.4366194
##    Residuals            37.760997    176      0.2145511
##
##
##
##  ASSUMPTION CHECKS
##
##  Homogeneity of Variances Test (Levene's)
##
##    F           df1    df2    p
##
##    1.190904      1    177    0.2766306
##
##
##
##  Normality Test (Shapiro-Wilk)
##
##    Statistic    p
##
##    0.9935997    0.6278124
##
##
##
##  POST HOC TESTS
##
##  Post Hoc Comparisons - intervention
##
##    intervention         intervention    Mean Difference    SE            df          t           p-tu
##
##    0               -    1                     0.2848817    0.07073173    176.0000    4.027636    0.0
##
##    Note. Comparisons are based on estimated marginal means
##
```

```
##
##  ESTIMATED MARGINAL MEANS
##
##  INTERVENTION
##
##  Estimated Marginal Means - intervention
##
##    intervention    Mean        SE            Lower       Upper
##
##    0               3.165983    0.05373502    3.059935    3.272031
##    1               2.881102    0.04557317    2.791161    2.971042
##
```





5.3 Power Analysis Resources

For G*Power, see the following slides: https://med.und.edu/research/daccota/_files/pdfs/berdc_resource_pdfs/sample_size_gpower_module.pdf

Unfortunately, there are not a lot of readily available implementations of power analysis methods for ANCOVA models in R. The following articles may be helpful for determining sample sizes in ANCOVA designs:

- https://www.sciencedirect.com/science/article/abs/pii/S0895435607000613
- https://link.springer.com/article/10.1007/s11336-019-09692-3
- https://www.tandfonline.com/doi/abs/10.1080/00273171.2016.1219841

6 Factorial Designs and Repeated Measures

6.1 SPSS

For instructions on how to estimate both a repeated-measures ANOVA and a factorial ANOVA, see the Class 8 slides and Field's text.

6.2 R

In this section, the dataset that will be used will be the same as that used in the ANCOVA example. The data come from an article titled "Changing personality traits with the help of a digital personality change intervention" (link here: https://doi.org/10.1073/pnas.2017548118, and data were found here: https://osf.io/g3yfz/). For this example, we will use the following variables:

- An intervention variable representing whether participants received the intervention (personality change intervention; coded as 1) or received waitlist control (coded as 0)
- A variable representing whether the goal of the participant was to reduce Neuroticism or whether it was to increase Conscientiousness
- A variable representing post-intervention Neuroticism scores

The first two variables can be considered categorical and the third variable can be considered continuous.

6.2.1 Factorial Designs

Here, we will estimate a factorical ANOVA model using the above mentioned data. This model may also be labeled as a 2x2 ANOVA model. The intervention variable and the goal variable (described above) will serve as the independent variables and Time 2 Neuroticism scores will serve as the dependent variable.

We will first estimate a model *without* an interaction between intervention and goal (i.e., to examine the unique effects of each variable in predicting the outcome) using both the `aov()` function. It should be noted that the same model can also be estimated using the `lm()` function because ANOVA models are subsumed under the general linear model, as emphasized in lecture and in the previous ANOVA examples above. The assumptions of this model can then be checked graphically using the `check_model()` function. Follow-up analyses are then calculated using the `estimate_means()` function, which looks at mean T2 Neuroticism scores by group, and actual hypothesis tests regarding these group mean differences can be calculated using the `estimate_contrasts()` function.

The `ANOVA()` function is then used to more compactly estimate a full factorial ANOVA with interactions, along with assumption tests. The output unfortunately gets cut off in the rendering of the document, but if you were to run this on your own you would see the results more clearly.

```
library(car)
library(modelbased)
library(effectsize)
library(performance)
library(jmv)

# Importing data
df_wide <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/Class 8 ANOVA Data.csv",
                    header = T)

# Preview data
head(df_wide)
```

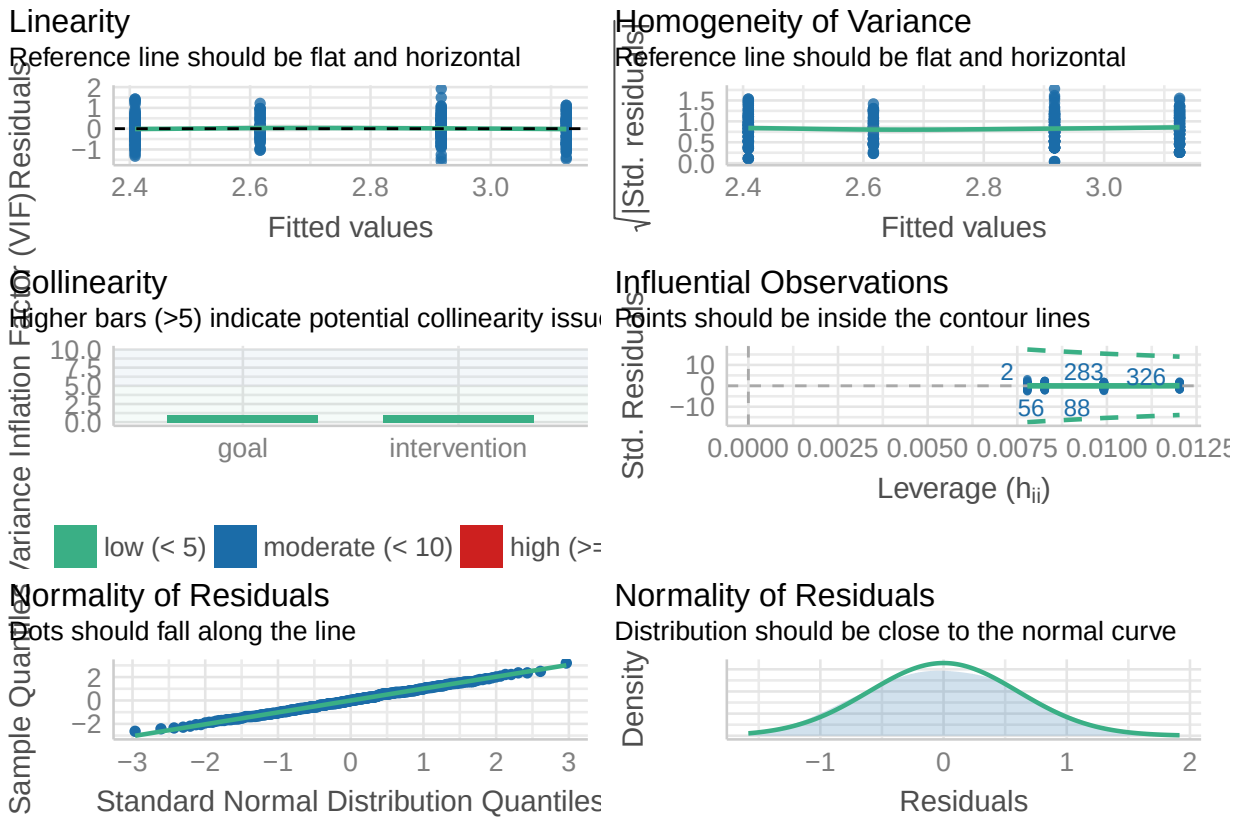
```
##   X    id intervention  goal     t1_N t2_N
## 1 1 p0004            0 plusC       NA   NA
## 2 2 p0005            0  minN       NA   NA
## 3 3 p0006            1  minN 3.916667   NA
## 4 4 p0008            1 plusC 2.000000   NA
## 5 5 p0009            1  minN 3.500000   NA
## 6 6 p0012            0  minN 3.250000 3.25
```

```
# Making sure that the categorical variables are indeed read by R as categorical
df_wide$intervention <- as.factor(df_wide$intervention)
df_wide$goal <- as.factor(df_wide$goal)
```

```r
# Factorial ANOVA without interaction
model1 <- aov(t2_N ~ intervention + goal, data = df_wide)
summary(model1)
```

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## intervention    1   4.57   4.568   12.29 0.000517 ***
## goal            1  21.09  21.091   56.77 4.83e-13 ***
## Residuals     327 121.48   0.371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 473 observations deleted due to missingness
```

```r
check_model(model1)
```




```r
# Factorial ANOVA without interaction - lm() version
model1 <- lm(t2_N ~ intervention + goal, data = df_wide)
summary(model1) # Provides some coefficients and global model fit indices
```

```
##
## Call:
## lm(formula = t2_N ~ intervention + goal, data = df_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58425 -0.41758 -0.00092  0.41575  1.91575
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    2.76709    0.03461  79.943  < 2e-16 ***
## intervention1  0.10390    0.03450   3.011   0.0028 **
## goal1          0.25439    0.03376   7.535 4.83e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6095 on 327 degrees of freedom
##   (473 observations deleted due to missingness)
## Multiple R-squared:  0.1744, Adjusted R-squared:  0.1693
## F-statistic: 34.53 on 2 and 327 DF,  p-value: 2.471e-14
```

```
Anova(model1) # Provides ANOVA tables for models run using lm()
```

```
## Anova Table (Type II tests)
##
## Response: t2_N
##               Sum Sq  Df F value    Pr(>F)
## intervention   3.369   1   9.069  0.002802 **
## goal          21.091   1  56.774 4.833e-13 ***
## Residuals    121.476 327
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Follow-up analyses on Factorial ANOVA without interaction
estimate_means(model1, at = c("intervention")) # Intervention differences
```

```
## Estimated Marginal Means
##
## intervention | Mean |   SE |       95% CI
## ----------------------------------------
## 0            | 2.87 | 0.05 | [2.76, 2.98]
## 1            | 2.66 | 0.04 | [2.58, 2.75]
##
## Marginal means estimated at intervention
```

```
estimate_contrasts(model1, contrast = c("intervention")) # Hypothesis test
```

```
## Marginal Contrasts Analysis
##
## Level1 | Level2 | Difference |       95% CI |   SE | t(327) |     p
## ------------------------------------------------------------------------
## 0      |      1 |       0.21 | [0.07, 0.34] | 0.07 |   3.01 | 0.003
##
## Marginal contrasts estimated at intervention
## p-value adjustment method: Holm (1979)
```

```
estimate_means(model1, at = c("goal")) # Goal differences
```

```
## Estimated Marginal Means
##
## goal  | Mean |   SE |       95% CI
## ----------------------------------
## minN  | 3.02 | 0.05 | [2.93, 3.11]
## plusC | 2.51 | 0.05 | [2.41, 2.61]
##
## Marginal means estimated at goal
```

```r
estimate_contrasts(model1, contrast = c("goal")) # Hypothesis test
```

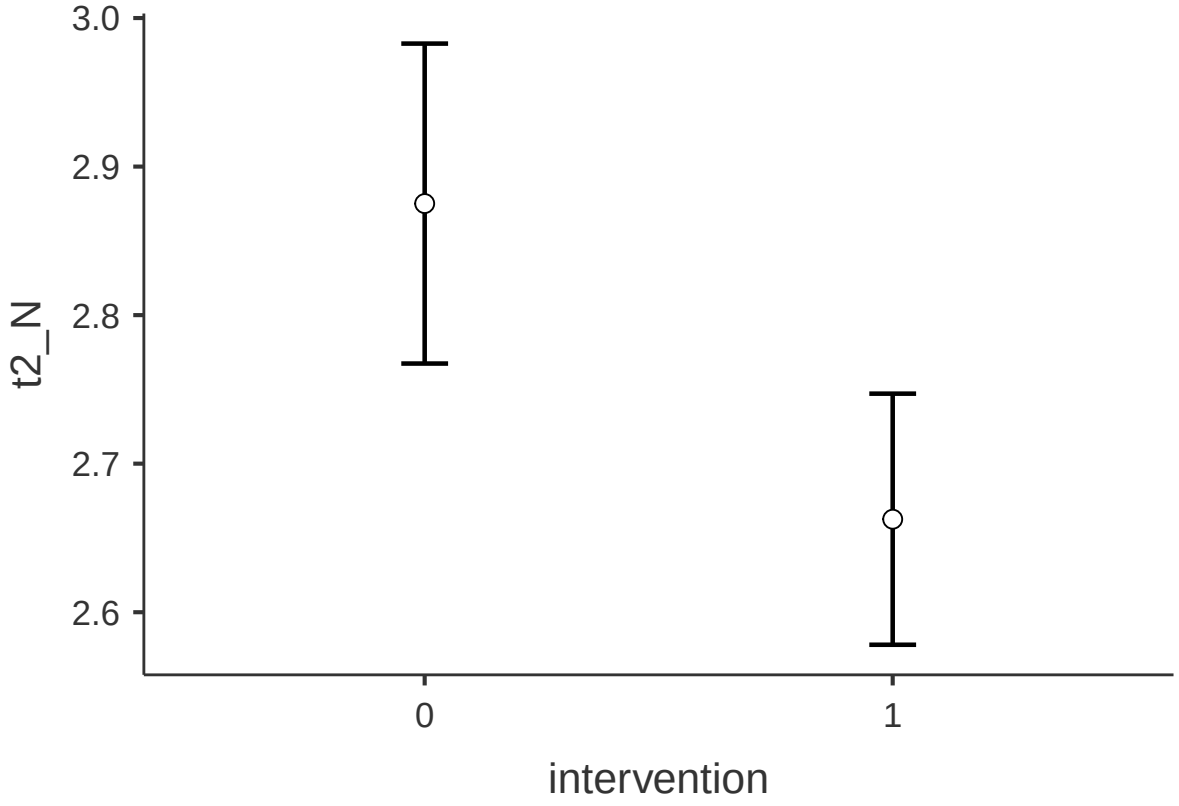
```
## Marginal Contrasts Analysis
##
## Level1 | Level2 | Difference |       95% CI |   SE | t(327) |      p
## ------------------------------------------------------------------------
## minN   |  plusC |       0.51 | [0.38, 0.64] | 0.07 |   7.53 | < .001
##
## Marginal contrasts estimated at goal
## p-value adjustment method: Holm (1979)
```

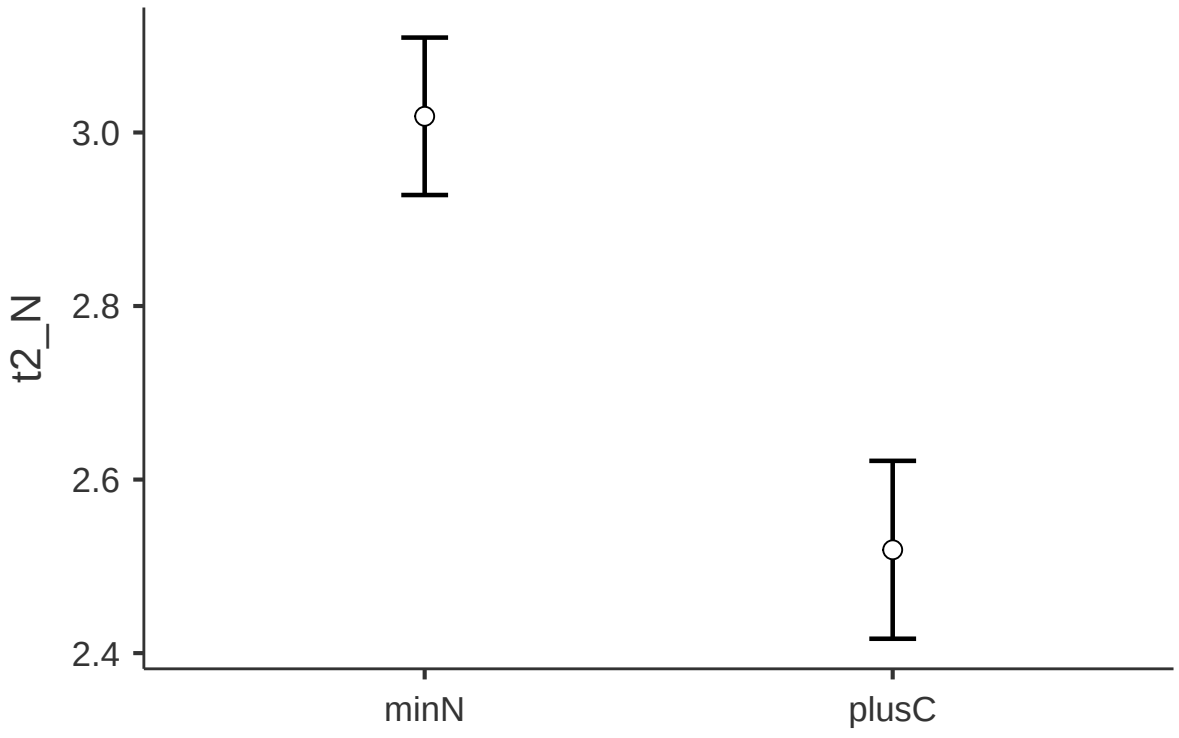
```r
# Factorial ANOVA with interaction plus follow-up analyses
ANOVA(
    data = df_wide,
    dep = t2_N,
    factors = c("intervention", "goal"),
    modelTest = T,
    ss = 2,
    homo = T,
    norm = T,
    postHoc = ~ intervention + goal + intervention:goal, # Comparisons to make
    emMeans = ~ intervention + goal + intervention:goal, # Actual mean values
    emmPlots = T,
    emmTables = T
)
```

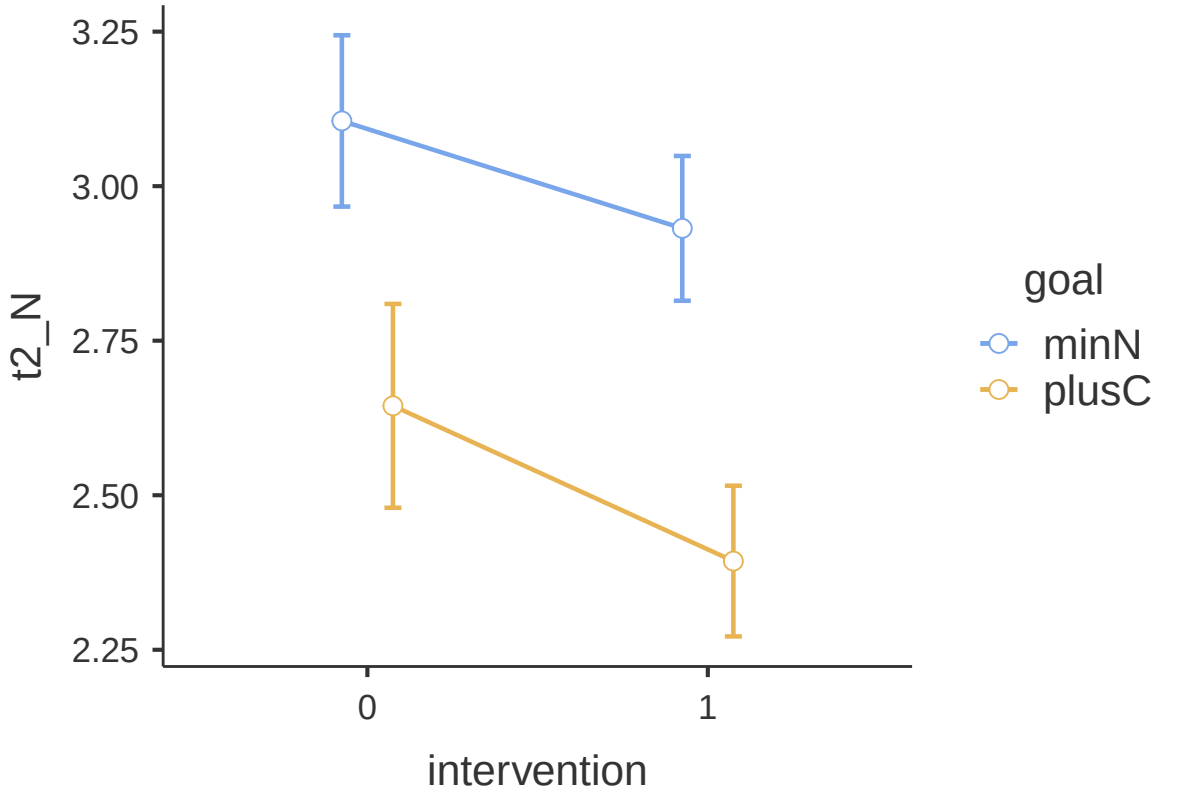
```
##
##  ANOVA
##
##  ANOVA - t2_N
##
##                          Sum of Squares    df     Mean Square    F             p
##
##    Overall model            24.5749173      3      8.1916391    23.0775442    < .0000001
##    intervention              3.3690328      1      3.3690328     9.0498822     0.0028316
##    goal                     21.0908316      1     21.0908316    56.6541056    < .0000001
##    intervention:goal         0.1150529      1      0.1150529     0.3090546     0.5786423
##    Residuals               121.3612152    326      0.3722737
##
##
##
##  ASSUMPTION CHECKS
##
##  Homogeneity of Variances Test (Levene's)
##
##    F            df1    df2    p
##
##    0.3566361      3    326    0.7843709
##
##
##
##  Normality Test (Shapiro-Wilk)
##
##    Statistic    p
##
```

```
##    0.9973091    0.8669997
##
##
##
##  POST HOC TESTS
##
##  Post Hoc Comparisons - intervention
##
##    intervention         intervention    Mean Difference    SE            df          t           p-tu
##
##    0               -    1                     0.2124964    0.06958990    326.0000    3.053552    0.0
##
##    Note. Comparisons are based on estimated marginal means
##
##
##  Post Hoc Comparisons - goal
##
##    goal         goal     Mean Difference    SE            df          t           p-tukey
##
##    minN    -    plusC          0.4995884    0.06958990    326.0000    7.179035    < .0000001
##
##    Note. Comparisons are based on estimated marginal means
##
##
##  Post Hoc Comparisons - intervention:goal
##
##    intervention    goal          intervention    goal     Mean Difference    SE            df
##
##    0               minN     -    0               plusC          0.4609015    0.10948827    326.0000
##                             -    1               minN           0.1738095    0.09224485    326.0000
##                             -    1               plusC          0.7120848    0.09381643    326.0000
##                    plusC    -    1               minN          -0.2870919    0.10280805    326.0000
##                             -    1               plusC          0.2511833    0.10422046    326.0000
##    1               minN     -    1               plusC          0.5382752    0.08592634    326.0000
##
##    Note. Comparisons are based on estimated marginal means
##
##
##  ESTIMATED MARGINAL MEANS
##
##  INTERVENTION
##
##  Estimated Marginal Means - intervention
##
##    intervention    Mean        SE            Lower       Upper
##
##    0               2.875105    0.05474413    2.767408    2.982801
##    1               2.662608    0.04296317    2.578088    2.747128
##
##
##
##  GOAL
##
##  Estimated Marginal Means - goal
```

```
##
##    goal     Mean        SE            Lower       Upper
##
##    minN     3.018651    0.04612243    2.927916    3.109386
##    plusC    2.519062    0.05211023    2.416548    2.621577
##
##
##
##  INTERVENTION:GOAL
##
##  Estimated Marginal Means - intervention:goal
##
##    goal     intervention    Mean        SE            Lower       Upper
##
##    minN     0               3.105556    0.07045317    2.966955    3.244156
##             1               2.931746    0.05954379    2.814607    3.048885
##    plusC    0               2.644654    0.08380949    2.479778    2.809530
##             1               2.393471    0.06195057    2.271597    2.515344
##
```







6.2.2 Repeated-measures ANOVA

In this section, the dataset that will be used comes from an article titled "Personality change through a digital-coaching intervention: Using measurement invariance testing to distinguish between trait domain,

facet, and nuance change" (see: https://doi.org/10.1177/08902070221145088), and the data were found here: https://osf.io/q7vgf/. Participants in this study underwent a personality change intervention, and the data include Neuroticism scores at pre-intervention, post-intervention, and at a follow-up assessment.

Here, we will examine whether there is an effect of time (i.e., the three time points) on Neuroticism scores using a repeated-measures ANOVA. In other words, we want to see if there are any significant differences in Neuroticism scores between any of the time points.

The `aov_ez()` function is a compact way of estimating ANOVA models that contain either within-person variables (e.g., time) or a combination of within-person and between-person variables. Similar to the other ANOVA models estimated in this document, follow-up analyses to the repeated-measures ANOVA can be done using the `estimate_means()` and `estimate_contrasts()` functions, which examine mean values at each time point and hypothesis tests comparing these mean values, respectively.

```
library(modelbased)
library(afex)

# Importing data
df_long <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/Class 8 RMANOVA Data.csv",
                    header = T)

# Preview data
head(df_long)
```

```
##   X    id time Neuroticism
## 1 1 p0005   t1    2.166667
## 2 2 p0005   t2    3.666667
## 3 3 p0005   t3    3.333333
## 4 4 p0006   t1    3.916667
## 5 5 p0006   t2          NA
## 6 6 p0006   t3          NA
```

```
# Estimating the RM-ANOVA
model <- aov_ez(
    id = "id",
    dv = "Neuroticism",
    data = df_long,
    within = "time"
)
```

```
## Warning: Missing values for following ID(s):
## p0006, p0009, p0034, p0048, p0093, p0108, p0120, p0160, p0161, p0163, p0176, p0178, p0181, p0186, p0
## Removing those cases from the analysis.
```

```
# Model results
summary(model)
```

```
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##             Sum Sq num Df Error SS den Df  F value    Pr(>F)
## (Intercept) 3338.1      1   94.753    126 4438.928 < 2.2e-16 ***
## time          10.4      2   35.807    252   36.664 1.057e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Mauchly Tests for Sphericity
##
##      Test statistic    p-value
## time        0.88892 0.00063647
##
##
## Greenhouse-Geisser and Huynh-Feldt Corrections
##  for Departure from Sphericity
##
##       GG eps Pr(>F[GG])
## time 0.90002  1.779e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##         HF eps   Pr(>F[HF])
## time 0.9122606 1.259164e-13
```

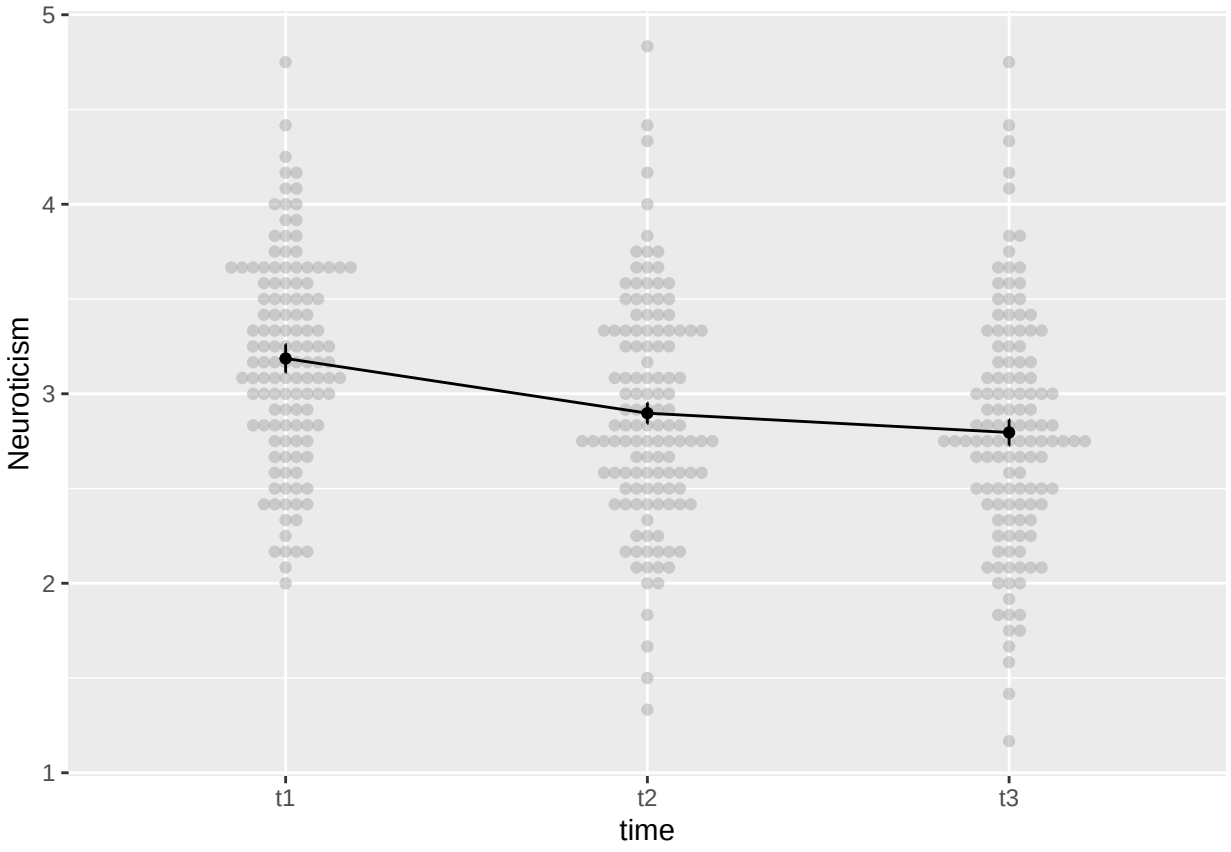
```
# Follow-up tests
estimate_means(model, at = "time")
```

```
## Estimated Marginal Means
##
## time | Mean |   SE |       95% CI
## ---------------------------------
## t1   | 3.19 | 0.05 | [3.09, 3.28]
## t2   | 2.90 | 0.05 | [2.79, 3.00]
## t3   | 2.80 | 0.06 | [2.69, 2.91]
##
## Marginal means estimated at time
```

```
estimate_contrasts(model)
```

```
## Marginal Contrasts Analysis
##
## Level1 | Level2 | Difference |       95% CI |   SE | t(126) |      p
## ------------------------------------------------------------------------
## t1     |     t2 |       0.29 | [0.18, 0.40] | 0.05 |   6.30 | < .001
## t1     |     t3 |       0.39 | [0.26, 0.52] | 0.05 |   7.20 | < .001
## t2     |     t3 |       0.10 | [0.00, 0.20] | 0.04 |   2.49 | 0.037
##
## Marginal contrasts estimated at time
## p-value adjustment method: Holm (1979)
```

```
# Graph
afex_plot(model, x = "time", error = "within") +
    geom_line(group = 1)
```



6.3 Power Analysis Resources

GPower is able to conduct power analyses for factorial and repeated designs, see their documentation on their website. Another resource that may be useful for GPower can be found here https://core.ecu.edu/wuenschk/docs30/GPower3-ANOVA-Factorial.pdf, as well as here https://med.und.edu/research/daccota/_files/pdfs/berdc_resource_pdfs/sample_size_gpower_module.pdf.

In R, the `superpower` package and its associated resources are likely the most contemporary, powerful, and useful methods for conducting power analyses of factorial and repeated-measures designs. See here: https://journals.sagepub.com/doi/full/10.1177/2515245920951503

Here is another book that may be useful and uses a different R package: https://www.taylorfrancis.com/books/mono/10.4324/9781315171500/applied-power-analysis-behavioral-sciences-christopher-aberson

7 Mixed Designs

7.1 R

We are now going to switch back to the *other* personality change dataset that was used for the ANCOVA and 2x2 ANOVA examples. Mixed ANOVAs allow us to look at the effects of both within-person and between-person variables, and the interactions between them. In this case, we might be interested in determining whether there is an interaction between time point (independent variable 1) and intervention status (independent variable 2) on Neuroticism scores (outcome). As with any other ANOVA model, the independent variables must be categorical - time point has two levels (pre-intervention, post-intervention) and intervention has two levels (1 = treatment, 0 = control). A significant interaction between these two variables would suggest that the degree of change in Neuroticism scores from Time 1 to Time 2 *depends* on whether a participant is in the treatment group or the control group. Another important point to note is that the data must be in long format, at least in R, in order to correctly run a mixed ANOVA.

Similar to the repeated-measures ANOVA, the `aov_ez()` function can be used to compactly estimate a mixed ANOVA.

```
library(modelbased)
library(afex)

# Importing the data
df <- read.csv("Class 8 ANOVA Data.csv", header = T)

# Selecting variables of interest
df <- select(df, id, intervention, t1_N, t2_N)
head(df) # Data are in wide format now
```

```
##      id intervention     t1_N t2_N
## 1 p0004            0       NA   NA
## 2 p0005            0       NA   NA
## 3 p0006            1 3.916667   NA
## 4 p0008            1 2.000000   NA
## 5 p0009            1 3.500000   NA
## 6 p0012            0 3.250000 3.25
```

```
# Making the dataset into long format
df_long <- df %>%
    pivot_longer(cols = t1_N:t2_N, names_to = "time", values_to = "Neuroticism")
head(df_long) # Good
```

```
## # A tibble: 6 x 4
##   id    intervention time  Neuroticism
##   <chr>        <int> <chr>       <dbl>
## 1 p0004            0 t1_N        NA
## 2 p0004            0 t2_N        NA
## 3 p0005            0 t1_N        NA
## 4 p0005            0 t2_N        NA
## 5 p0006            1 t1_N         3.92
## 6 p0006            1 t2_N        NA
```

```
# Estimating the mixed ANOVA and its results
model <- aov_ez(
    id = "id",
    dv = "Neuroticism",
    data = df_long,
    within = "time",
```

```
    between = "intervention"
)
```

```
## Warning: Missing values for following ID(s):
## p0004, p0005, p0006, p0008, p0009, p0018, p0036, p0042, p0048, p0072, p0093, p0105, p0112, p0120, p0
## Removing those cases from the analysis.
```

```
summary(model)
```

```
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##                   Sum Sq num Df Error SS den Df   F value    Pr(>F)
## (Intercept)       4958.6      1  253.973    327 6384.3825 < 2.2e-16 ***
## intervention         1.8      1  253.973    327    2.3607   0.12540
## time                 0.4      1   36.325    327    3.1930   0.07488 .
## intervention:time    3.0      1   36.325    327   26.8581 3.841e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Follow-up tests
estimate_means(model)
```

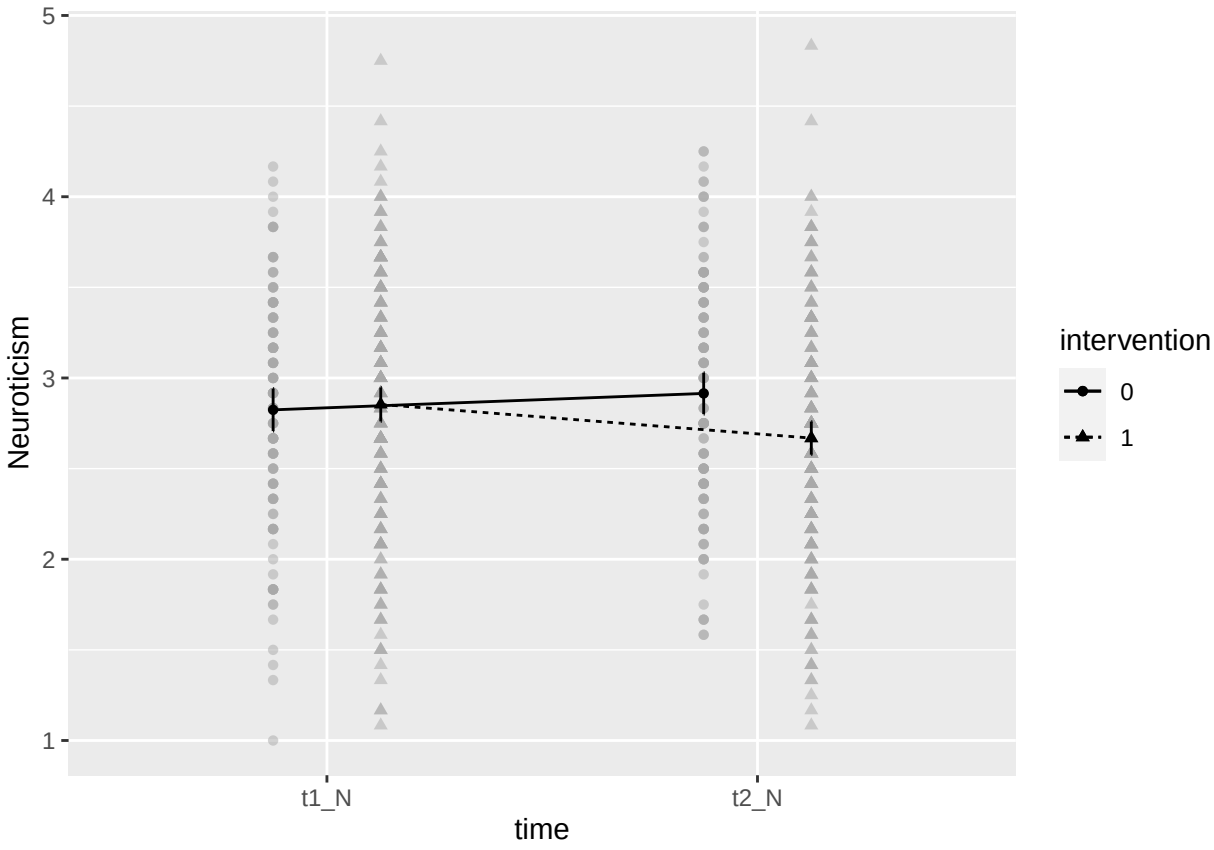
```
## Estimated Marginal Means
##
## intervention | time | Mean |   SE |       95% CI
## ------------------------------------------------
## 0            | t1_N | 2.82 | 0.06 | [2.71, 2.94]
## 1            | t1_N | 2.85 | 0.05 | [2.76, 2.95]
## 0            | t2_N | 2.91 | 0.06 | [2.80, 3.03]
## 1            | t2_N | 2.67 | 0.05 | [2.58, 2.76]
##
## Marginal means estimated at intervention, time
```

```
estimate_contrasts(model, contrast = c("intervention", "time"))
```

```
## Marginal Contrasts Analysis
##
## Level1   |   Level2 | Difference |        95% CI |   SE | t(327) |      p
## ------------------------------------------------------------------------------
## 0 - t1_N | 0 - t2_N |      -0.09 | [-0.20, 0.02] | 0.04 |  -2.17 | 0.133
## 0 - t1_N | 1 - t1_N |      -0.03 | [-0.23, 0.17] | 0.08 |  -0.39 | 0.980
## 0 - t1_N | 1 - t2_N |       0.16 | [-0.04, 0.36] | 0.08 |   2.06 | 0.167
## 0 - t2_N | 1 - t2_N |       0.25 | [ 0.05, 0.44] | 0.07 |   3.31 | 0.006
## 1 - t1_N | 0 - t2_N |      -0.06 | [-0.26, 0.14] | 0.08 |  -0.81 | 0.851
## 1 - t1_N | 1 - t2_N |       0.19 | [ 0.10, 0.27] | 0.03 |   5.59 | < .001
##
## Marginal contrasts estimated at intervention, time
## p-value adjustment method: Holm (1979)
```

```
# Graph
afex_plot(model, x = "time", trace = "intervention")
```

```
## Warning: Panel(s) show a mixed within-between-design.
## Error bars do not allow comparisons across all means.
## Suppress error bars with: error = "none"
```


5
4
3
2
1
Neuroticism
t1_N
t2_N
time
intervention
0
1

7.2 Power Analysis Resources

Similar to factorial designs, in R, the `superpower` package and its associated resources are likely the most contemporary, powerful, and useful methods for conducting power analyses. See here: https://journals.sagepub.com/doi/full/10.1177/2515245920951503

More online resources are the following: - https://designingexperiments.shinyapps.io/BUCSS_ss_power_wa/ (online app for mixed ANOVA) - https://www.taylorfrancis.com/books/mono/10.4324/9781315171500/applied-power-analysis-behavioral-sciences-christopher-aberson

8 Mediation and Moderation

8.1 SPSS

See class slides for information about how to run mediation and moderation models in SPSS.

8.2 R

In this section, the dataset that will be used comes from a paper titled "A Hierarchical Taxonomy of Psychopathology (HiTOP) Primer for Mental Health Researchers" (https://doi.org/10.1177/21677026211017834) and the data were found here: https://osf.io/8myzw/?view_only=. This dataset contains variables corresponding to various symptom measures, a relationship quality measure, and demographic information. For the depression symptom measure, I converted total scores into a binary variable representing whether a participant met or didn't meet the cut-off for a probable depression diagnosis.

8.2.1 Moderation

```
library(tidymodels)

# Import data
df <- read.csv("Tutorial2_raw.csv", header = T)

# Creating new variables
df <- df %>%
    mutate(
        phq_total = phq1 + phq2 + phq3 + phq4 + phq5 + phq7 + phq8 + phq9,
        rq_total = qmi1 + qmi2 + qmi3 + qmi4 + qmi5 + qmi6,
        gad_total = gad1 + gad2 + gad3 + gad4 + gad5 + gad6 + gad7,
        sp_total = spi1 + spi2 + spi3 + spi4 + spi5 + spi6 + spi7 + spi8 + spi9 +
            spi10 + spi11 + spi12 + spi13 + spi14 + spi15 + spi16 + spi17
    ) %>%
    mutate(
        dep_dx = ifelse(
            phq_total >= 10,
            "Yes",
            "No"
        )
    ) %>%
    mutate(
        dep_dx = as.factor(dep_dx),
        gender = as.factor(gender) # 0 = Male, 1 = Female
    )
```

First, I estimate a moderation model involving two categorical IVs (gender and depression diagnosis) and their interaction predicting relationship quality.

The results here suggest a few things:

- The intercept (27.54) represents mean RQ scores for those who are male and do not have a depression diagnsosis. This value is also statistically significant.
- The `gender1` coefficient represents the difference in mean RQ scores between females and males that do not have a depression diagnosis. That is, among those without a depression diagnosis, women tend to have RQ scores that are 1.11 units less than men. In other words, women tend to have worse relationship quality than men, provided that there is no depression diagnosis.
- The `dep_dx1` coefficient represents the difference in mean RQ scores between those with and without a depression, *among men only*. That is, men with a depression diagnosis tend to have RQ scores that

are 2.50 units larger than men without a depression diagnosis. In other words, men with depression tend to have better relationship quality compared to men without depression.
- The last coefficient can be thought of in terms of how much gender buffers or accentuates differences between those with and without depression. In this case, because the coefficient is positive, the difference in relationship quality between those with and without depression is .30 units greater in women than in men. Specifically, the difference in relationship quality between men who have depression vs not was 2.50 (as mentioned above), but this difference increases to 2.80 between women who have depression vs not. This coefficient can also be thought of as representing the 'difference in differences'.

```
library(jtools)
library(interactions)

# Estimate model and results
model1 <- lm(rq_total ~ gender * dep_dx, data = df)
summ(model1)
```

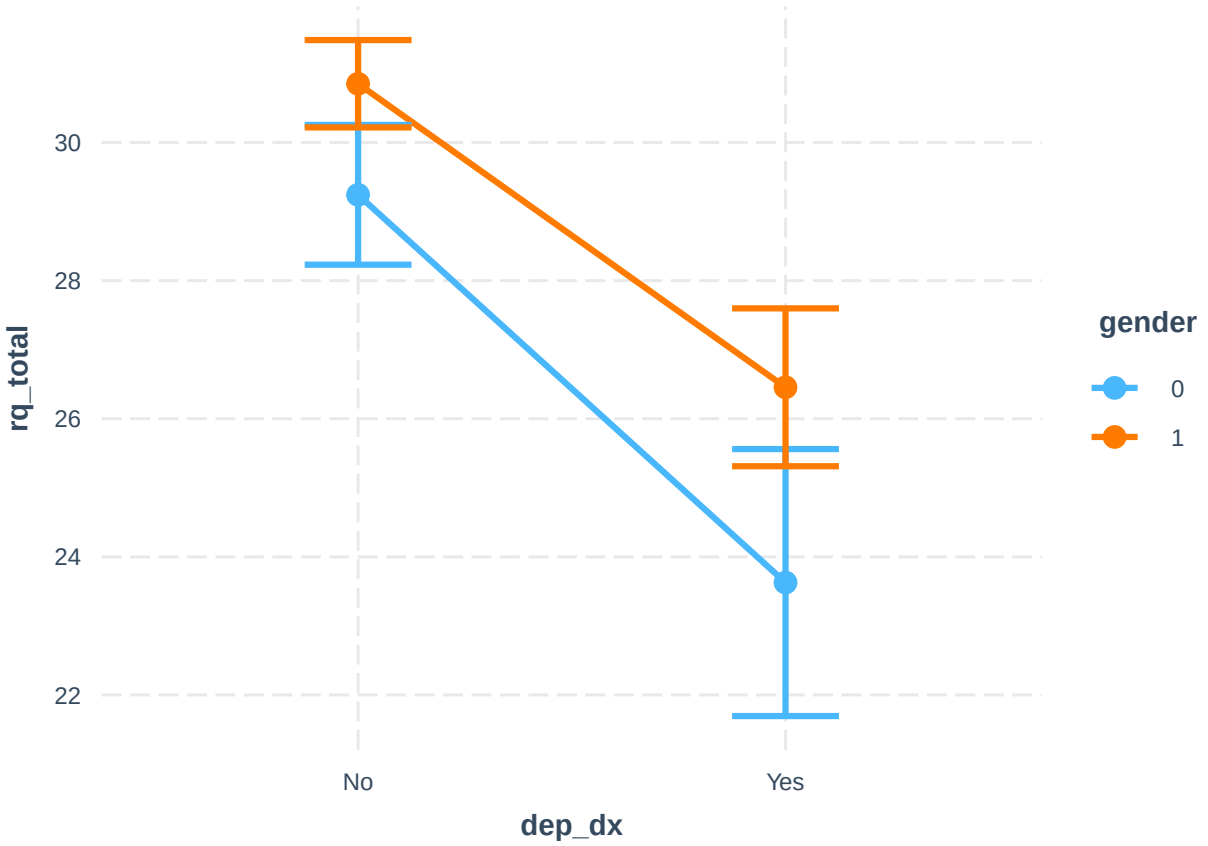
Observations	725
Dependent variable	rq_total
Type	OLS linear regression

F(3,721)	26.75
R^2	0.10
Adj. R^2	0.10

	Est.	S.E.	t val.	p
(Intercept)	27.54	0.32	85.07	0.00
gender1	-1.11	0.32	-3.43	0.00
dep_dx1	2.50	0.32	7.73	0.00
gender1:dep_dx1	0.30	0.32	0.94	0.35

Standard errors: OLS

```
# Graph
cat_plot(model1, pred = dep_dx, modx = gender, geom = "line")
```



Second, I estimate another moderation model involving one continuous IV (depression severity) and one categorical IV (gender), and their interaction, predicting relationship quality.

The results here suggest a few things:

- The intercept represents mean RQ scores among males who have an average score on the PHQ-9.
- The `gender` coefficient represents *the difference* in mean RQ scores between males and females who have an average score on the PHQ-9. That is, among those with average depression, women tend to have better relationship quality than men (by 1.98 units). In other words, this coefficient can be considered as the 'independent additive' effect of being in one group vs the other in terms of generally decreasing or increasing predicted outcome scores.
- The `phq_total` coefficient represents the strength of the relation between depression severity and relationship quality, *among men only*. This coefficient is negative which suggests that men with more depression tend to have worse relationship quality. That is, if Andrew has a score of 20 on the PHQ-9, and Anthony has a score of 21 on the PHQ-9, it is expected that Anthony will have a relationship quality score that is .62 units lower than Andrew.
- The interaction coefficient represents the difference in the slopes of depression with relationship quality if one were to calculate separate slopes for women and men. You can maybe interpret this coefficient as the degree to which the X-Y relation in the reference group is buffered or accentuated in the second group. Let's go back to the previous example with Andrew and Anthony. Recall that Anthony has lower relationship quality (by .62 units) because he has more depression than Anthony (by 1 unit). Let's now pretend we have two women, Mandy and Jessica, who have the same depression scores of 20 and 21, respectively. If we plug in the respective values into the regression equation, it is expected that Jessica will have a relationship quality score that is **.44 units** lower than Mandy. Compare this to the case with Anthony and Andrew, where the difference in relationship quality scores was **.62 units**. Therefore, one way to interpret the interaction is that there within women and men there is a negative relation between depression severity and relationship quality, but this relation is *more pronounced* in men.

```r
# Estimate model and results
model2 <- lm(rq_total ~ gender * phq_total, data = df)
summ(model2, center = T)
```

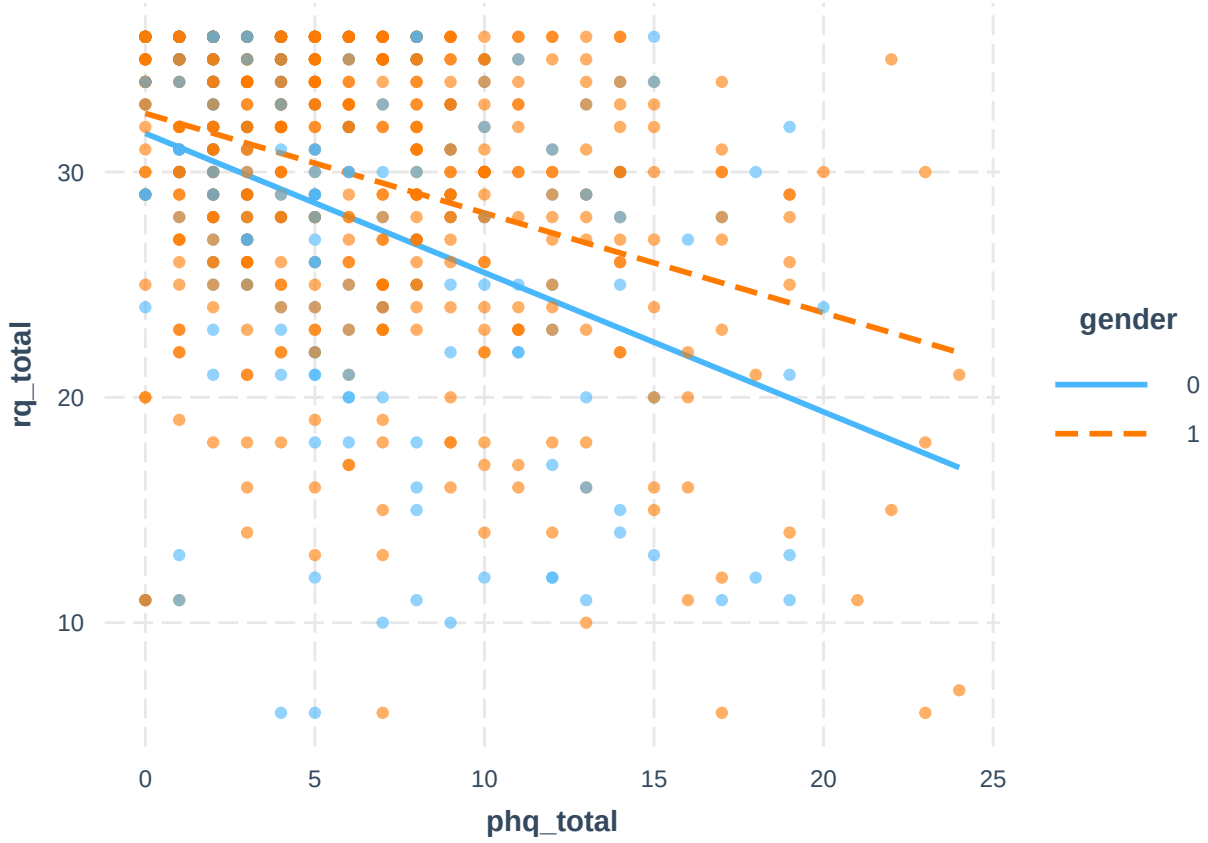
Observations	725
Dependent variable	rq_total
Type	OLS linear regression

F(3,721)	40.70
R^2	0.14
Adj. R^2	0.14

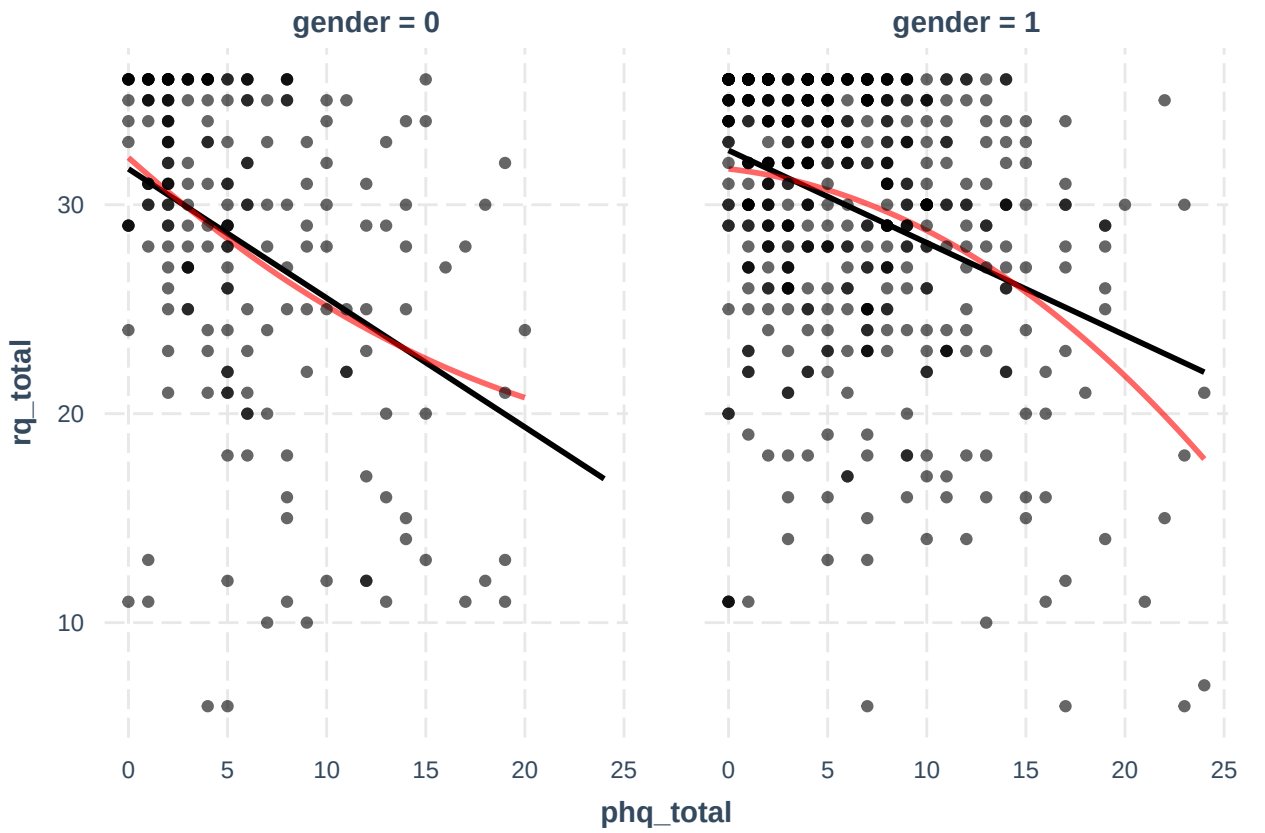
	Est.	S.E.	t val.	p
(Intercept)	27.89	0.45	62.58	0.00
gender	1.98	0.52	3.78	0.00
phq_total	-0.62	0.09	-6.71	0.00
gender:phq_total	0.18	0.11	1.64	0.10

Standard errors: OLS; Continuous predictors are mean-centered.

```r
# Graphs
interact_plot(model2, pred = phq_total, modx = gender, plot.points = T)
```



```
interact_plot(model2, pred = phq_total, modx = gender, plot.points = T, linearity.check = T)
```




```
sim_slopes(model2, pred = phq_total, modx = gender)
```

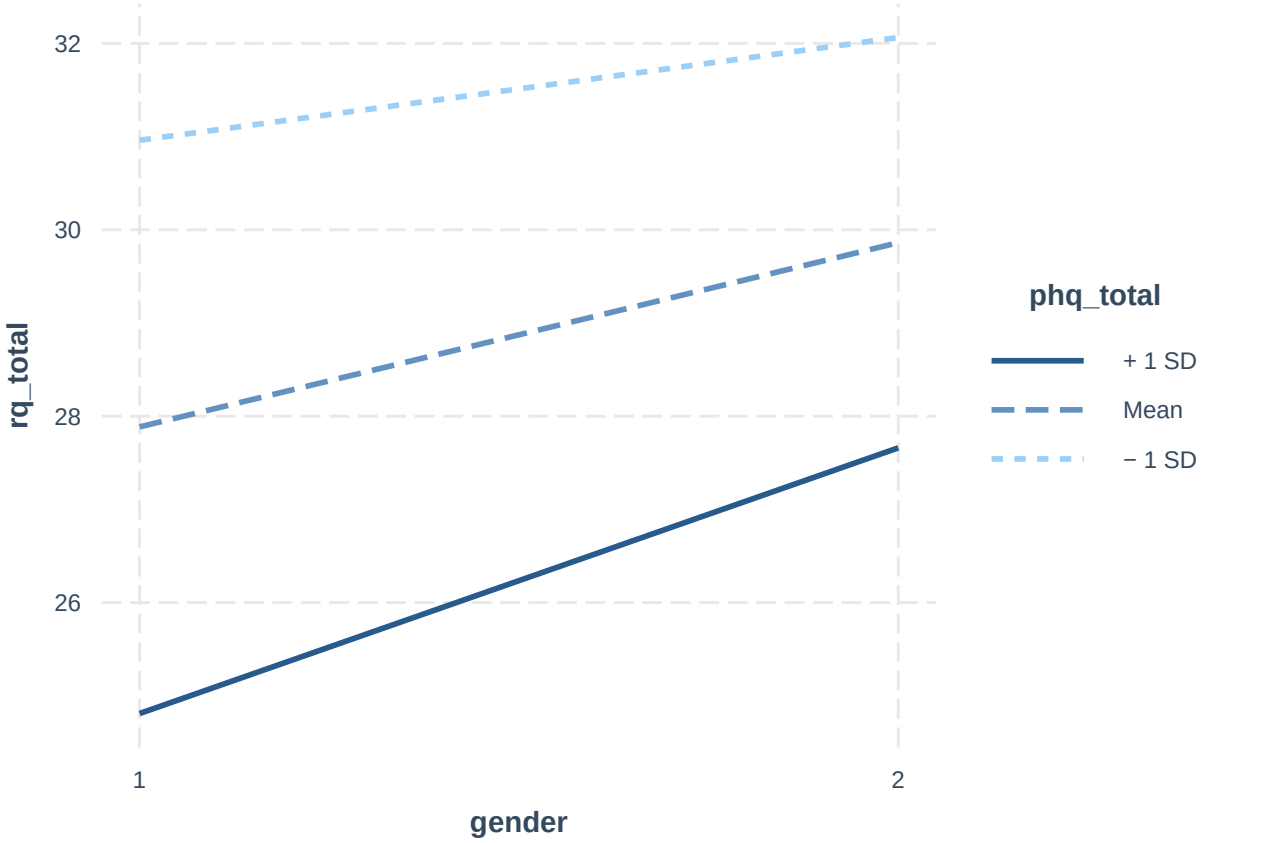
```
## Warning: Johnson-Neyman intervals are not available for factor moderators.
```

```
## SIMPLE SLOPES ANALYSIS
##
## Slope of phq_total when gender = 1:
##
##    Est.   S.E.   t val.      p
## ------- ------ -------- ------
##   -0.44   0.05    -8.09   0.00
##
## Slope of phq_total when gender = 0:
##
##    Est.   S.E.   t val.      p
## ------- ------ -------- ------
##   -0.62   0.09    -6.71   0.00
```

You can also flip the interpretation around if your research question has to do with whether existing group differences depend on some continuous variable (rather than the other way around as in the example just demonstrated). In this case, one might be interested in determining whether gender differences in relationship quality depend/change depending on certain simultaneous values of depression severity.

```
# Converting gender to numeric to run the plots
df$gender = as.numeric(df$gender)

# Model again
```

```r
model2 <- lm(rq_total ~ gender * phq_total, data = df)

interact_plot(model2, pred = gender, modx = phq_total)
```




```r
sim_slopes(model2, pred = gender, modx = phq_total, johnson_neyman = F)
```

```
## SIMPLE SLOPES ANALYSIS
##
## Slope of gender when phq_total =  1.220945 (- 1 SD):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   1.10   0.74     1.50   0.13
##
## Slope of gender when phq_total =  6.197241 (Mean):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   1.98   0.52     3.78   0.00
##
## Slope of gender when phq_total = 11.173538 (+ 1 SD):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   2.85   0.76     3.76   0.00
```

Third, I estimate a moderation model involving two continuous variables (depression severity and anxiety

severity), and their interaction, predicting relationship quality.

Here, the results suggest a few things:

- The intercept coefficient represents mean RQ scores when someone has average depression and average anxiety.
- The `phq_total` coefficient represents the main effect of depression while holding anxiety fixed at its average. In other words, among two people with average anxiety, if one person has a PHQ-9 score that is one unit larger than the other person, then that first person is expected to have lower relationship quality (by .43 units).
- The `gad_total` coefficient holds the same interpretation as the `phq_total` coefficient, with the difference being that the variables are switched around (i.e., main effect of anxiety while holding depression fixed at its average).
- The interaction coefficient represents how much the initial depression-RQ association is either buffered or accentuated with increasing levels of anxiety (the moderating variable). More technically, the initial association of -.50 becomes even more negative for every one-unit increase in anxiety scores. Using the example with Andrew, Anthony, Mandy, and Jessica above (this time we are disregarding gender), let's say that Andrew and Anthony have scores of 20 and 21 on the PHQ-9 (respectively), and they both have average anxiety. Anthony is expected to have a relationship quality score that is .50 units lower than Andrew. On the other hand, now let's say Mandy and Jessica have scores of 20 and 21 on the PHQ-9, but both have anxiety scores one unit above the average. Jessica is expected to have a relationship quality score that is **.51** units lower than Mandy - compare this with Anthony and Andrew where their difference was **.50**. In other words, one can interpret this moderation effect as suggesting that the effect of depression on relationship quality becomes more pronounced if you have more anxiety.

```
# Estimate model and results
model3 <- lm(rq_total ~ phq_total * gad_total, data = df)
summ(model3, center = T)
```

Observations	725
Dependent variable	rq_total
Type	OLS linear regression

F(3,721)	34.64
R^2	0.13
Adj. R^2	0.12

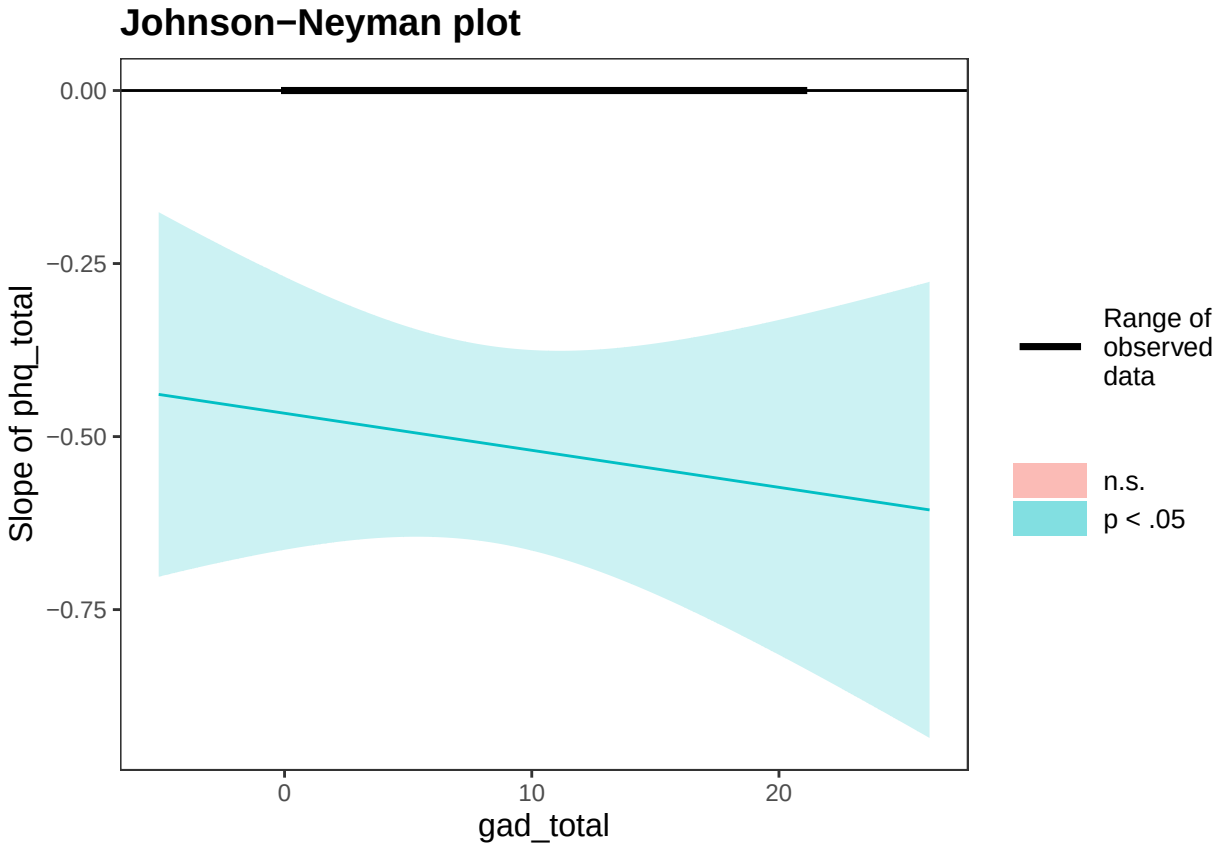
	Est.	S.E.	t val.	p
(Intercept)	29.43	0.29	102.79	0.00
phq_total	-0.50	0.07	-6.71	0.00
gad_total	0.05	0.07	0.63	0.53
phq_total:gad_total	-0.01	0.01	-0.63	0.53

Standard errors: OLS; Continuous predictors are mean-centered.

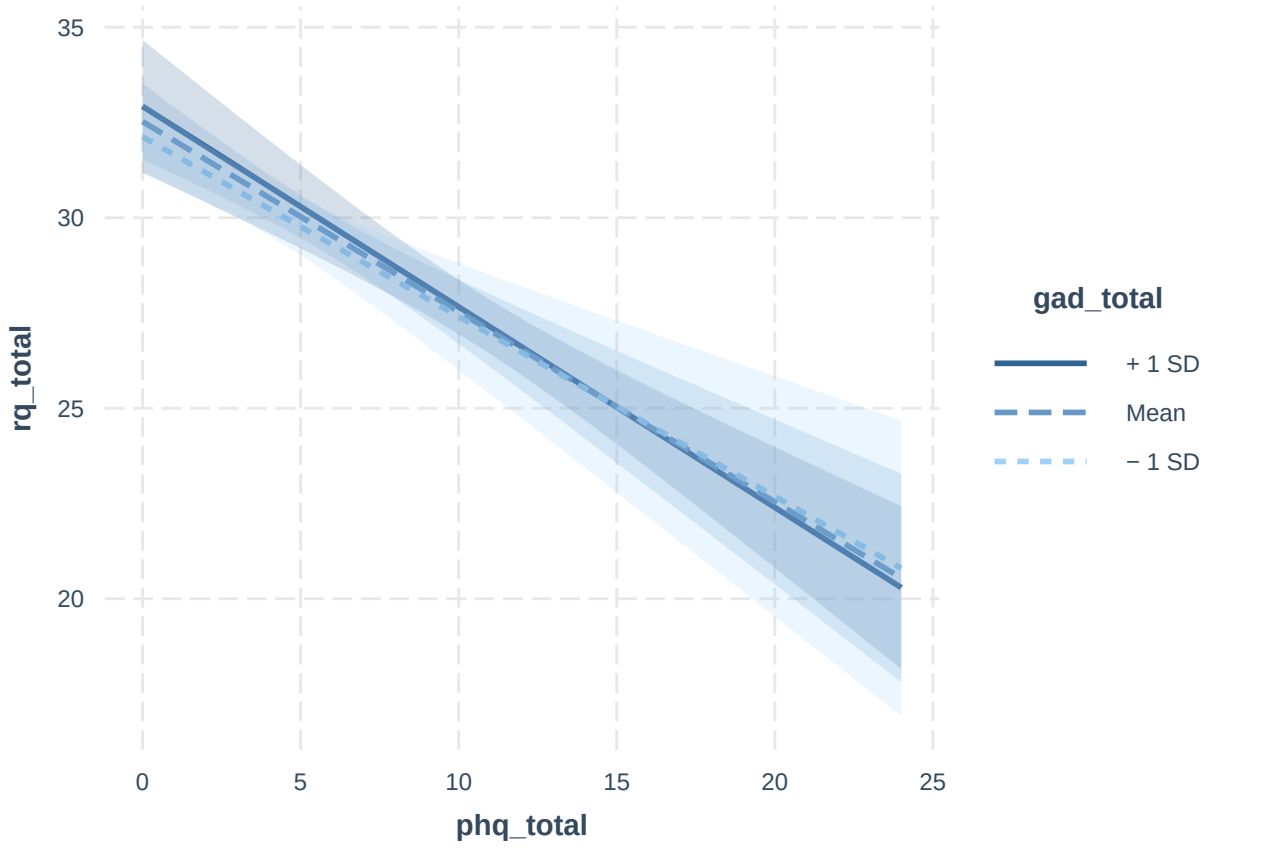
```
# Follow-up analyses
probe_interaction(model3, pred = phq_total, modx = gad_total, cond.int = T, interval = T, jnplot = T)
```

```
## JOHNSON-NEYMAN INTERVAL
##
## When gad_total is INSIDE the interval [-13.76, 52.18], the slope of
## phq_total is p < .05.
##
```

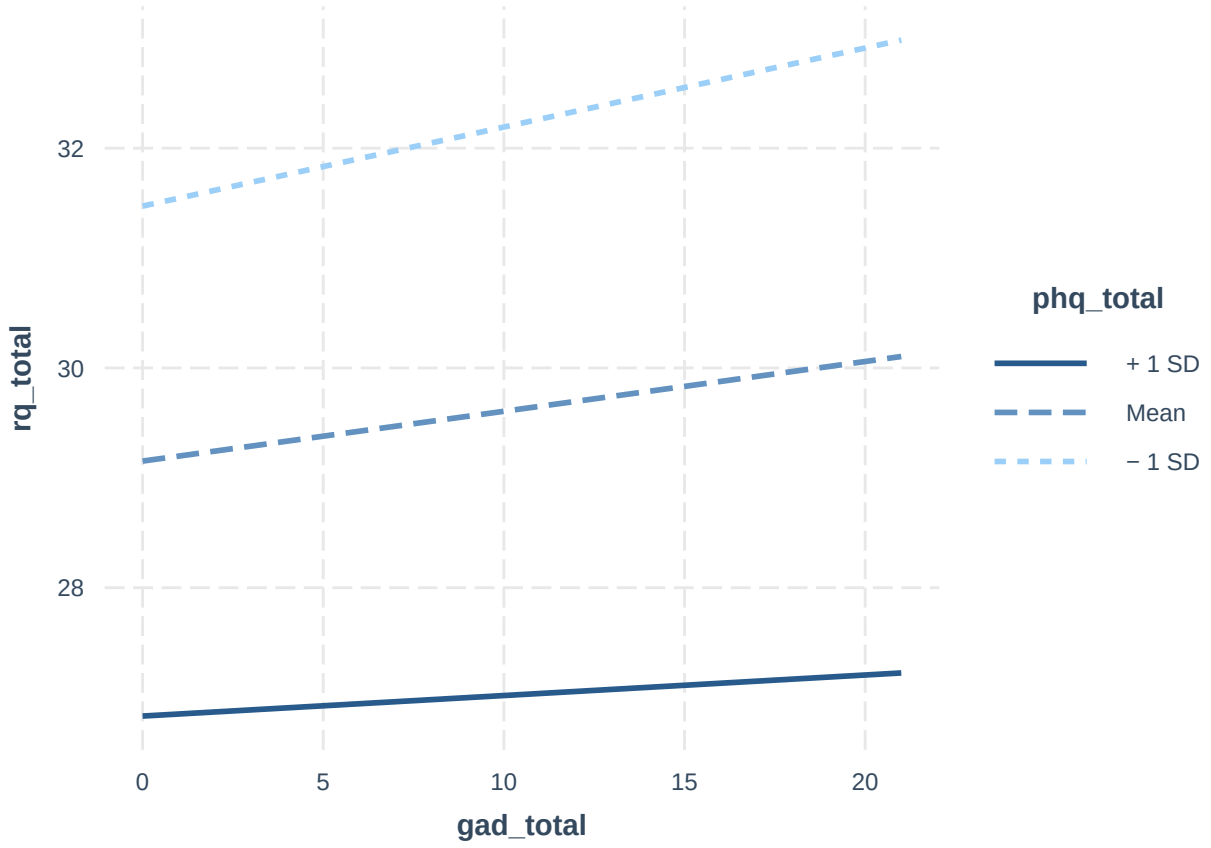
```
## Note: The range of observed values of gad_total is [0.00, 21.00]
```




```
## SIMPLE SLOPES ANALYSIS
##
## When gad_total =  1.039530 (- 1 SD):
##
##                              Est.   S.E.   t val.      p
## -------------------------- ------- ------ -------- ------
## Slope of phq_total           -0.47   0.09    -4.99   0.00
## Conditional intercept        29.20   0.44    66.63   0.00
##
## When gad_total =  6.131034 (Mean):
##
##                              Est.   S.E.   t val.      p
## -------------------------- ------- ------ -------- ------
## Slope of phq_total           -0.50   0.07    -6.71   0.00
## Conditional intercept        29.43   0.29   102.79   0.00
##
## When gad_total = 11.222539 (+ 1 SD):
##
##                              Est.   S.E.   t val.      p
## -------------------------- ------- ------ -------- ------
## Slope of phq_total           -0.53   0.08    -6.87   0.00
## Conditional intercept        29.66   0.49    60.21   0.00
```




```
interact_plot(model3, pred = gad_total, modx = phq_total)
```




```
sim_slopes(model3, pred = phq_total, modx = gad_total)
```

```
## JOHNSON-NEYMAN INTERVAL
##
## When gad_total is INSIDE the interval [-13.76, 52.18], the slope of
## phq_total is p < .05.
##
## Note: The range of observed values of gad_total is [0.00, 21.00]
##
## SIMPLE SLOPES ANALYSIS
##
## Slope of phq_total when gad_total =  1.039530 (- 1 SD):
##
##    Est.   S.E.   t val.      p
## ------- ------ -------- ------
##   -0.47   0.09    -4.99   0.00
##
## Slope of phq_total when gad_total =  6.131034 (Mean):
##
##    Est.   S.E.   t val.      p
## ------- ------ -------- ------
##   -0.50   0.07    -6.71   0.00
##
## Slope of phq_total when gad_total = 11.222539 (+ 1 SD):
##
##    Est.   S.E.   t val.      p
## ------- ------ -------- ------
```

```
##   -0.53   0.08    -6.87   0.00
```

8.2.2 Mediation

Here, we will test out a model where the association between social phobia and relationship quality is *mediated* by anxiety symptoms. Note that the original data are cross-sectional which suggests that inferences of causality from these results are not warranted.

To run a simple mediation model in R, we have to specify two separate regression models first and then we combine them in the final `mediate()` function. The first regression model we have to specify is one where the outcome is the mediating variable (in this case, anxiety) and the IV is the X variable (social phobia). The second regression model we have to specify is one where the outcome is Y (RQ) and the IVs are X and M.

After estimating the mediation model, we can view the results with `summary()`. There are a few notable things in the output:

- The `Total Effect` coefficient represents the zero-order unstandardized association between social phobia scores and relationship quality (i.e., `c`). This represents the initial effect that may or may not be mediated.
- The `ADE` coefficient represents the association between social phobia and relationship quality *after* accounting for the mediating variable of anxiety (i.e., `c'`). As with typical mediation analyses, this `ADE` coefficient value is compared with the `Total Effect` coefficient value, where a mediation effect *may* be significant if the ADE coefficient is substantially lower than the Total Effect coefficient (which is formally tested in the `ACME` coefficient).
- The `ACME` coefficient represents the indirect effect (i.e., `c - c'` or `a X b`).

```
library(mediation)

# First model (A -> B)
med_fit <- lm(gad_total ~ sp_total, data = df)

# Second mdoel (A + B -> C)
out_fit <- lm(rq_total ~ gad_total + sp_total, data = df)

# Results
med_out <- mediate(med_fit, out_fit, treat = "sp_total", mediator = "gad_total", robustSE = T)
summary(med_out)
```

```
##
## Causal Mediation Analysis
##
## Quasi-Bayesian Confidence Intervals
##
##                Estimate 95% CI Lower 95% CI Upper p-value
## ACME            -0.0520      -0.0801        -0.02  <2e-16 ***
## ADE             -0.0733      -0.1224        -0.03   0.004 **
## Total Effect    -0.1254      -0.1672        -0.08  <2e-16 ***
## Prop. Mediated   0.4159       0.1841         0.73  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Sample Size Used: 725
##
##
## Simulations: 1000
```

8.3 Power Analysis Resources

For moderation, there is a recent paper with an associated application that allows you to conduct power analyses. Links to the paper, the application, and more can be found here: https://dbaranger.github.io/InteractionPoweR/

For mediation, the following book has a chapter that talks about how to conduct a power analysis for mediation models in R specifically: https://www.taylorfrancis.com/books/mono/10.4324/9781315171500/applied-power-analysis-behavioral-sciences-christopher-aberson

This paper may also be useful to look at for mediation power analysis and apparently contains an application: https://journals.sagepub.com/doi/abs/10.1177/1948550617715068

9 Categorical Outcomes

9.1 SPSS

See lecture slides for more information on how to run the below tests in SPSS.

9.2 R

Here, I am using the same dataset from the Moderation and Mediation section above, where participants were assessed on a variety of symptom measures. For the purposes of the categorical analyses, I created two new variables representing probable depression and probably anxiety diagnoses that were dependent on whether a participant met or didn't meet a particular cut-off score on the measures.

```
library(tidymodels)

# Import data
df <- read.csv("Tutorial2_raw.csv", header = T)

# Creating new variables
df <- df %>%
    mutate(
        phq_total = phq1 + phq2 + phq3 + phq4 + phq5 + phq7 + phq8 + phq9,
        rq_total = qmi1 + qmi2 + qmi3 + qmi4 + qmi5 + qmi6,
        gad_total = gad1 + gad2 + gad3 + gad4 + gad5 + gad6 + gad7,
        sp_total = spi1 + spi2 + spi3 + spi4 + spi5 + spi6 + spi7 + spi8 + spi9 +
            spi10 + spi11 + spi12 + spi13 + spi14 + spi15 + spi16 + spi17
    ) %>%
    mutate(
        dep_dx = ifelse(
            phq_total >= 10,
            "Yes MDD",
            "No MDD"
        ),
        gad_dx = ifelse(
            gad_total >= 10,
            "Yes GAD",
            "No GAD"
        )
    ) %>%
    mutate(
        dep_dx = as.factor(dep_dx),
        gender = as.factor(gender), # 0 = Male, 1 = Female
        gad_dx = as.factor(gad_dx)
    )
```

9.2.1 Chi-square test of independence

Here, we want to test whether there is an association between having a depression diagnosis and having an anxiety diagnosis. Recall from lecture that you can frame a chi-square test as testing whether the ratio of anxiety diagnosed vs non-diagnosed participants significantly differs depending on whether said participants have or don't have a simultaneous depression diagnosis. One way to run a chi-square test is to first create a 2x2 contingency table showing the counts of those with different levels of the two categorical variables. This table then is passed into the `chisq.test()` function where the actual test is run. Various effect sizes for the test can be estimated - such as the odds ratio or phi.

```r
# Unconditional ratio of anxiety dx vs non-dx
table(df$gad_dx)
```

```
##
##  No GAD Yes GAD
##     549     176
```

```r
# Actual test
table <- table(df$dep_dx, df$gad_dx)
table
```

```
##
##           No GAD Yes GAD
##   No MDD     500      59
##   Yes MDD     49     117
```

```r
out <- chisq.test(table, correct = F)
out
```

```
##
##  Pearson's Chi-squared test
##
## data:  table
## X-squared = 250.05, df = 1, p-value < 2.2e-16
```

```r
# Follow-up analyses
out$expected
```

```
##
##             No GAD   Yes GAD
##   No MDD  423.2979 135.70207
##   Yes MDD 125.7021  40.29793
```

```r
out$observed
```

```
##
##           No GAD Yes GAD
##   No MDD     500      59
##   Yes MDD     49     117
```

```r
oddsratio(table)
```

```
## Odds ratio |         95% CI
## ---------------------------
## 20.24      | [13.18, 31.07]
```

```r
# Effect size
library(effectsize)
phi(out)
```

```
## Phi (adj.) |       95% CI
## -------------------------
## 0.59       | [0.52, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

9.2.2 Logistic regression

Here, we will run a logistic regression model where we would like to predict the presence of a depression diagnosis (i.e., a binary outcome) using anxiety and social anxiety severity scores as independent variables. In other words, you can view this model as similar to a multiple regression model with the exception of the outcome being binary rather than continuous. Whereas regular regression models were run using the `lm()` function, in order to estimate models involving categorical outcomes, the `glm()` function must be used along with specifying the *type* of outcome that is being used using the `family` parameter. In the case of logistic regression, the distribution of the outcome can be considered binomial, and therefore the binomial family is specified.

The results of the model, as mentioned in lecture, can be expressed in several ways. By default, the coefficients are expressed in terms of logits, but the coefficients can then be exponentiated to express the coefficients in terms of odds ratios.

```
library(jtools)

# Estimating the model
model <- glm(dep_dx ~ gad_total + sp_total, data = df, family = binomial())

# Results in terms of logits
summ(model)
```

Observations	725
Dependent variable	dep_dx
Type	Generalized linear model
Family	binomial
Link	logit

$\chi^2(2)$	313.40
Pseudo-R^2 (Cragg-Uhler)	0.53
Pseudo-R^2 (McFadden)	0.40
AIC	472.74
BIC	486.49

	Est.	S.E.	z val.	p
(Intercept)	-4.31	0.29	-14.85	0.00
gad_total	0.33	0.03	11.11	0.00
sp_total	0.03	0.01	3.32	0.00

Standard errors: MLE

```
# Results in terms of ORs
summ(model, exp = T)
```

Observations	725
Dependent variable	dep_dx
Type	Generalized linear model
Family	binomial
Link	logit

$\chi^2(2)$	313.40
Pseudo-R² (Cragg-Uhler)	0.53
Pseudo-R² (McFadden)	0.40
AIC	472.74
BIC	486.49

	exp(Est.)	2.5%	97.5%	z val.	p
(Intercept)	0.01	0.01	0.02	-14.85	0.00
gad_total	1.39	1.31	1.47	11.11	0.00
sp_total	1.03	1.01	1.05	3.32	0.00

Standard errors: MLE

9.3 Power Analysis Resources

The following set of slides has good instructions about how to conduct power analyses for various categorical methods: https://med.und.edu/research/daccota/_files/pdfs/berdc_resource_pdfs/sample_size_gpower_module.pdf

In R, the following book has a few chapters that discuss how to do a power analysis for various categorical methods: https://www.taylorfrancis.com/books/mono/10.4324/9781315171500/applied-power-analysis-behavioral-sciences-christopher-aberson

10 Non-Parametric Analyses

10.1 SPSS

See the supplemental slides that were provided on Quercus for information on how to run some of the tests below.

10.2 R

Here, we are using the same data as in the t-test and ANOVA section above.

```
library(tidymodels)

# Importing data
df <- read.csv("/Users/michaelcarnovale/Documents/Stats I Fall 2022/rater_target_data.csv",
    header = T
)

# Here, I'm creating a mean trust score per participant (i.e., 'rater')
df <- df %>%
    group_by(rater) %>%
    mutate(trust_overall = mean(trust, na.rm = T)) %>%
    ungroup()

# Here, I'm creating a mean trust score per rater by the type of
# target they are rating (i.e., either BPD or healthy control)
df <- df %>%
    group_by(rater, target_group) %>%
    mutate(trust_group = mean(trust, na.rm = T)) %>%
    ungroup()

# Finally converting the data to wide format
# Pivot_wider makes new columns in wide format, where the new column names
# come from categorical values of 'target_group', the actual values in the
# new column come from the 'trust_group' variable
df_wide <- df %>%
    dplyr::select(rater, rater_group, target_group, trust_overall, trust_group) %>%
    pivot_wider(
        names_from = target_group, values_from = trust_group,
        values_fn = list(trust_group = mean)
    )

# Here, I'm making a modified version of the long format
df_long <- df %>%
    dplyr::select(rater, rater_group, target_group, trust_overall, trust_group) %>%
    distinct()
```

10.2.1 Mann Whitney test

This test can be somewhat considered a nonparametric version of an independent-samples t-test, at least conceptually, in the sense that this test is appropriate when you are interested in comparing average scores between two independent groups. This test uses ranks instead, and compares groups on such, rather than means. Therefore, you can't technically say that you are comparing means between two groups if you are using this test. As well, this test *does not* test the null hypothesis of equality in medians.

```r
library(coin)

# First removing one of the groups so that there are only two groups to compare
df_wide2 <- df_wide %>%
    filter(!rater_group == "SP") %>%
    mutate(
        rater_group = as.factor(rater_group)
    )

# Actual test
wilcox_test(trust_overall ~ rater_group, data = df_wide2)
```

```
##
##  Asymptotic Wilcoxon-Mann-Whitney Test
##
## data:  trust_overall by rater_group (BPD, HC)
## Z = -2.7798, p-value = 0.00544
## alternative hypothesis: true mu is not equal to 0
```

10.2.2 Mood's Median test

Mood's median test allows you to test the null hypothesis of equal medians between two independent groups. You can think of this test as conceptually similar to the independent-samples t-test in the sense that Mood's median test is appropriate for research questions related to comparisons of averages between two groups, but is concerned with the median estimate rather than the mean.

```r
median_test(trust_overall ~ rater_group, data = df_wide2)
```

```
##
##  Asymptotic Two-Sample Brown-Mood Median Test
##
## data:  trust_overall by rater_group (BPD, HC)
## Z = -1.3269, p-value = 0.1845
## alternative hypothesis: true mu is not equal to 0
```

10.2.3 Wilcoxon Signed Rank test

Wilcoxon's signed rank test allows you to test the null hypothesis of equal ranks between two paired groups. Conceptually, you can think of this test as similar to a paired t-test, in the sense that you might be interested in whether average scores between two paired groups may be different (e.g., average scores between two time points). While a paired t-test is concerned with the means of difference scores, Wilcoxon Signed Rank test converts the difference scores into ranks and then tests the null of equal mean ranks.

```r
wilcox.test(trust_group ~ target_group, data = df_long, paired = T)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  trust_group by target_group
## V = 529, p-value = 3.661e-10
## alternative hypothesis: true location shift is not equal to 0
```

10.2.4 McNemar's test

McNemar's test can be thought of as a paired-samples version of the chi-squared test, such that this test is appropriate if you are interested in whether scores on some binary variable are significantly different

between two paired groups (e.g., whether diagnosis rates are different at two time points). For an example of McNemar's test, see here: https://www.statology.org/mcnemars-test-r/.

10.2.5 Cochran's Q test

Cochran's Q test is basically an extension to McNemar's test in the case when there are more than two paired groups (e.g., three time points). Similar to McNemar's test, Cochran's Q test requires that the outcome variable of interest is binary. See the syntax at the bottom of this package for an example of Cochran's Q test in R: https://rpkgs.datanovia.com/rstatix/reference/cochran_qtest.html.

10.2.6 Kruskal-Wallis test

The Kruskal-Wallis test can be thought of as a nonparametric version of a one-way ANOVA, but rather than focusing on raw means (like an ANOVA), this test is focused on differences in mean *ranks* between two or more independent groups. Similar to an ANOVA, however, this test initially provides a *global* statistical test which suggests to you whether there is some significant difference(s) between the groups. Post-hoc tests must be conducted in order to probe these differences, and the Mann Whitney test mentioned above can be used to test groupwise differences, much like how t-tests are used for groupwise differences in parametric ANOVAs.

```
# Global test
kruskal.test(trust_overall ~ rater_group, data = df_wide)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  trust_overall by rater_group
## Kruskal-Wallis chi-squared = 8.8107, df = 2, p-value = 0.01221
```

```
# Post-hoc tests
pairwise.wilcox.test(df_wide$trust_overall, df_wide$rater_group)
```

```
## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot compute
## exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot compute
## exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot compute
## exact p-value with ties
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  df_wide$trust_overall and df_wide$rater_group
##
##    BPD   HC
## HC 0.017 -
## SP 0.248 0.107
##
## P value adjustment method: holm
```

10.2.7 Friedman's test

Friedman's test can be thought of as a nonparametric version of a repeated-measures ANOVA that is focused on mean *ranks* rather than raw means. An example of Friedman's test in R can be found here: https://www.datanovia.com/en/lessons/friedman-test-in-r/.

10.3 Power Analysis Resources

The following sets of slides have good instructions on how to conduct power analyses for various non-parametric tests, in both GPower and R, respectively:

- https://med.und.edu/research/daccota/_files/pdfs/berdc_resource_pdfs/sample_size_gpower_module.pdf
- https://med.und.edu/research/daccota/_files/pdfs/berdc_resource_pdfs/sample_size_r_module.pdf

11 Factor Analysis/Structural Equation Modeling (SEM)

11.1 R

A goal of factor analysis (and structural equation modeling more broadly) is to try and account for why a bunch of variables are substantially correlated with each other, and to use this information for more abstract analyses involving latent variables (i.e., variables that are unobserved but inferred from the data). In a lot of our psychological measures, we assume that the set of items within the measure are 'tapping into' or 'assessing' a more abstract and unobserved psychological construct (latent variable) - for example, the GAD-7 contains seven items that assess different criteria of GAD, but it is assumed that all of these items are generally representative of the larger construct of 'anxiety' (this is especially reinforced with the fact that a total score is calculated to represent general anxiety severity). We may be interested in research questions involving this larger construct of 'anxiety', rather than the individual items, and although the total score is meant to represent this larger construct, it is assumed that this total score contains measurement error and does not represent the 'true' level of anxiety that an individual has (i.e., an anxiety score without measurement error). If measurement error is not accounted for, then the inferences that we might make from our raw data may not be correct. How can we try and model and analyze a more 'clean' version of the construct of anxiety, and make more correct inferences? This is where SEM comes in, of which factor analysis can be considered a subset of the type of analyses that can be done under the SEM umbrella. SEM can be considered to be a larger framework for all sorts of **multivariate** analyses (e.g., multiple regression, path analyses) because it is generally concerned with modeling the relations among multiple variables, but the term 'SEM' is usually thought of when one is analyzing latent variables. Factor analysis allows you to estimate the latent variables in the first place, and these latent variables then can be placed in a larger, more complex model using SEM. Therefore, one can say that SEM is like a vehicle itself and factor analysis is a type of engine for the vehicle.

Factor analysis proceeds almost like a model fitting exercise where you invoke a particular model and see if the data conform to this model. In this case, you invoke a latent variable structure that may underlie a set of raw variables. For example, using the GAD-7 example above, you might have a hypothesis that the seven items are a result of *a single latent variable* (otherwise known as a unidimensional structure). You would then construct this model, estimate what the data *should* look like if the model was true, and see how well *your* data matches the estimated data. If there is a good match, then your data support a unidimensional structure. In factor analysis, there are a few different indices that quantify how much your data fit the proposed model, all of which should be reported: (a) chi-square test for model fit (should ideally be **non-significant**), (b) CFI (should ideally be above .90), (c) TLI (should also ideally be above .90), (d) RMSEA (should be less than .08), and (e) SRMR (should also be less than .08). With these models, you are also given something called 'factor loadings', which are basically regression coefficients between your raw variables and your latent variables. Although beyond the scope of this document, it should be mentioned that there are two types of factor analysis - exploratory (EFA) and confirmatory (CFA). The distinction between using EFA vs CFA becomes more important when you are estimating larger factor analysis models with more than one latent variable underlying the same set of items (e.g., modeling a set of items as a function of *two* latent variables that represent *two subscales* of a given measure). EFA allows your items to have loadings on *all* of the latent variables (e.g., if you think that each item contributes to a particular latent variable), while CFA allows your items to have loadings on *only one* latent variable.

This above procedure of estimation remains similar when you are interested in constructing a larger SEM, where you might be interested in whether certain latent variables are associated with each other, rather than whether a set of raw variables conforms to a certain latent variable structure. For example, you may model the GAD-7 items as a unidimensional latent variable, the PHQ-9 items as another unidimensional latent variable, and ultimately you want to see how related the two latent variables are. This entire model would be constructed, you would then estimate what the data *should* look like if this whole model was true, then see how well your data matches the estimated data. You would then use the same indices mentioned above to quantify data-model fit.

Another important thing to keep in mind with factor analysis and SEM is the scale of the raw variables that are being used to model the latent variables. In factor analysis and SEM, there are a variety of estimation

methods (i.e., methods that assist in constructing the model and comparing your data to the model), some of which are more appropriate for certain types of raw variables. For example, if your raw variables can be considered continuous (e.g., total scores from different, but related, measures), one would ideally use the robust maximum likelihood estimation method. On the other hand, if your raw variables can be considered discrete (e.g., ordinal Likert items, binary items), one would ideally use the WLSMV estimation method.

As a first example, we will run a unidimensional factor analysis on the set of GAD-7 items. Below, we first specify the structure of the latent variable as a function of the raw items (in the `model` line). Second, we then fit the model to the data using the `cfa()` function, where we specify the model structure, the dataset, and whether the variables are ordinal (in this case they are). Lastly, we look at the actual results of the model to examine fit indices and loadings, using the `summary()` function.

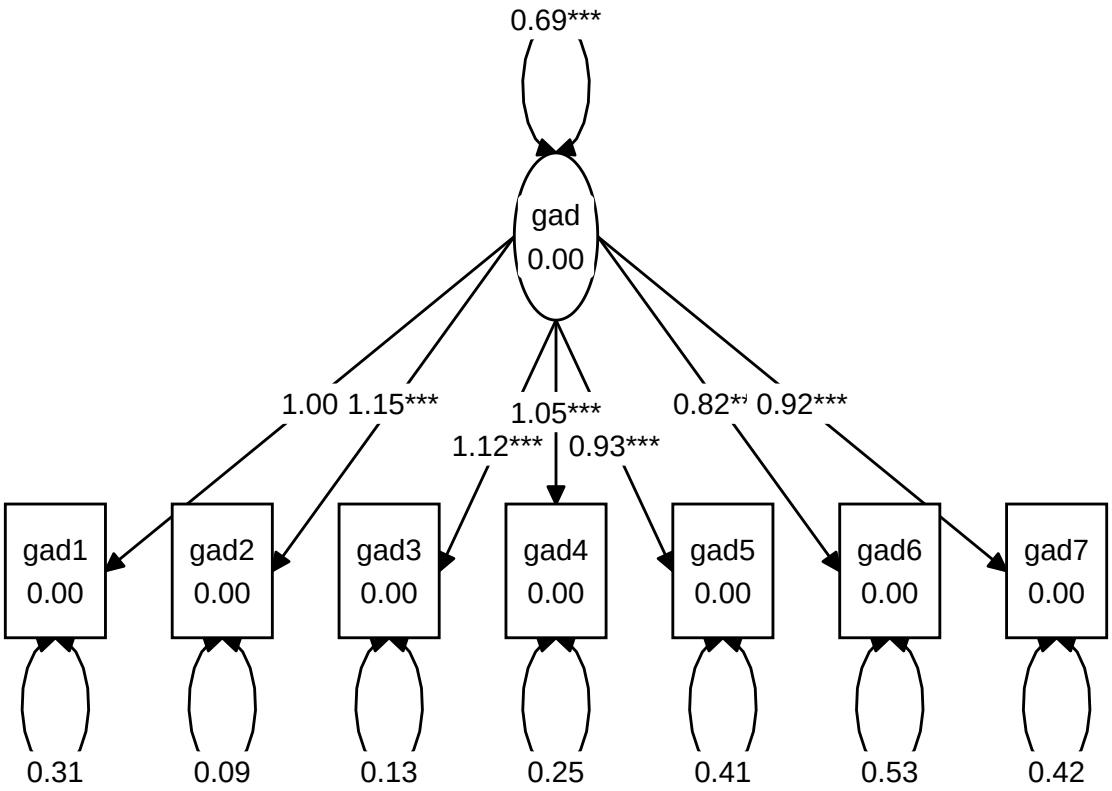
```
library(lavaan)
library(semTools)
library(tidySEM)

# Importing data
df <- read.csv("Tutorial2_raw.csv", header = T)

# Specifying the model
model <- "gad =~ gad1 + gad2 + gad3 + gad4 + gad5 + gad6 + gad7"

# Fitting the model
model_fit <- cfa(model, data = df, ordered = T)

# Model results
summary(model_fit, fit.measures = T, standardized = T)
```

```
## lavaan 0.6-12 ended normally after 20 iterations
##
##   Estimator                                         DWLS
##   Optimization method                           NLMINB
##   Number of model parameters                        28
##
##   Number of observations                           725
##
## Model Test User Model:
##                                               Standard      Robust
##   Test Statistic                                48.875     110.164
##   Degrees of freedom                                14          14
##   P-value (Chi-square)                           0.000       0.000
##   Scaling correction factor                                  0.450
##   Shift parameter                                            1.489
##     simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                             27278.327   14907.354
##   Degrees of freedom                                21          21
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  1.831
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.999       0.994
```

```
##   Tucker-Lewis Index (TLI)                       0.998       0.990
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.059       0.097
##   90 Percent confidence interval - lower         0.041       0.081
##   90 Percent confidence interval - upper         0.077       0.115
##   P-value RMSEA <= 0.05                          0.193       0.000
##
##   Robust RMSEA                                                  NA
##   90 Percent confidence interval - lower                        NA
##   90 Percent confidence interval - upper                        NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.036       0.036
##
## Parameter Estimates:
##
##   Standard errors                           Robust.sem
##   Information                                 Expected
##   Information saturated (h1) model        Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   gad =~
##     gad1              1.000                               0.830    0.830
##     gad2              1.147    0.020   56.404    0.000    0.951    0.951
##     gad3              1.122    0.020   55.178    0.000    0.931    0.931
##     gad4              1.045    0.021   49.993    0.000    0.867    0.867
##     gad5              0.925    0.029   32.067    0.000    0.768    0.768
##     gad6              0.824    0.030   27.164    0.000    0.684    0.684
##     gad7              0.921    0.028   32.940    0.000    0.764    0.764
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .gad1              0.000                               0.000    0.000
##    .gad2              0.000                               0.000    0.000
##    .gad3              0.000                               0.000    0.000
##    .gad4              0.000                               0.000    0.000
##    .gad5              0.000                               0.000    0.000
##    .gad6              0.000                               0.000    0.000
##    .gad7              0.000                               0.000    0.000
##     gad               0.000                               0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     gad1|t1         -0.487    0.049  -10.013    0.000   -0.487   -0.487
##     gad1|t2          0.720    0.051   14.045    0.000    0.720    0.720
##     gad1|t3          1.405    0.068   20.718    0.000    1.405    1.405
##     gad2|t1         -0.113    0.047   -2.412    0.016   -0.113   -0.113
```

```
##     gad2|t2           0.779    0.052   14.956    0.000    0.779    0.779
##     gad2|t3           1.414    0.068   20.747    0.000    1.414    1.414
##     gad3|t1          -0.392    0.048   -8.180    0.000   -0.392   -0.392
##     gad3|t2           0.641    0.050   12.765    0.000    0.641    0.641
##     gad3|t3           1.310    0.064   20.322    0.000    1.310    1.310
##     gad4|t1          -0.411    0.048   -8.548    0.000   -0.411   -0.411
##     gad4|t2           0.608    0.050   12.191    0.000    0.608    0.608
##     gad4|t3           1.278    0.063   20.150    0.000    1.278    1.278
##     gad5|t1           0.399    0.048    8.327    0.000    0.399    0.399
##     gad5|t2           1.217    0.062   19.771    0.000    1.217    1.217
##     gad5|t3           1.819    0.089   20.471    0.000    1.819    1.819
##     gad6|t1          -0.650    0.050  -12.908    0.000   -0.650   -0.650
##     gad6|t2           0.558    0.049   11.323    0.000    0.558    0.558
##     gad6|t3           1.247    0.062   19.966    0.000    1.247    1.247
##     gad7|t1           0.246    0.047    5.228    0.000    0.246    0.246
##     gad7|t2           0.989    0.056   17.696    0.000    0.989    0.989
##     gad7|t3           1.597    0.076   20.984    0.000    1.597    1.597
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .gad1             0.311                              0.311    0.311
##    .gad2             0.095                              0.095    0.095
##    .gad3             0.134                              0.134    0.134
##    .gad4             0.248                              0.248    0.248
##    .gad5             0.410                              0.410    0.410
##    .gad6             0.532                              0.532    0.532
##    .gad7             0.416                              0.416    0.416
##     gad              0.689    0.024   28.282    0.000    1.000    1.000
##
## Scales y*:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     gad1             1.000                              1.000    1.000
##     gad2             1.000                              1.000    1.000
##     gad3             1.000                              1.000    1.000
##     gad4             1.000                              1.000    1.000
##     gad5             1.000                              1.000    1.000
##     gad6             1.000                              1.000    1.000
##     gad7             1.000                              1.000    1.000
```

```
# Graph
graph_sem(model_fit)
```



Here, we will now run a SEM where we are interested in the association between an anxiety latent variable and a depression latent variable. The anxiety latent variable is modeled as a function of all 7 GAD-7 items, and the depression latent variable is modeled as a function of all PHQ-9 items. Model fitting proceeds using the `sem()` function, and the actual results of the model can be found with the `summary()` function. Importantly, the regression coefficient of interest between the anxiety and depression latent variable can be found under the 'Regressions' section of the output.

```
# Specifying the model
model <- "
gad =~ gad1 + gad2 + gad3 + gad4 + gad5 + gad6 + gad7
dep =~ phq1 + phq2 + phq3 + phq4 + phq5 + phq6 + phq7 + phq8 + phq9

gad ~ dep
"

# Fitting the model
model_fit <- sem(model, data = df, ordered = T)

# Model results
summary(model_fit, fit.measures = T, standardized = T)
```

```
## lavaan 0.6-12 ended normally after 33 iterations
##
##   Estimator                                         DWLS
##   Optimization method                           NLMINB
##   Number of model parameters                        65
##
##   Number of observations                           725
##
## Model Test User Model:
```

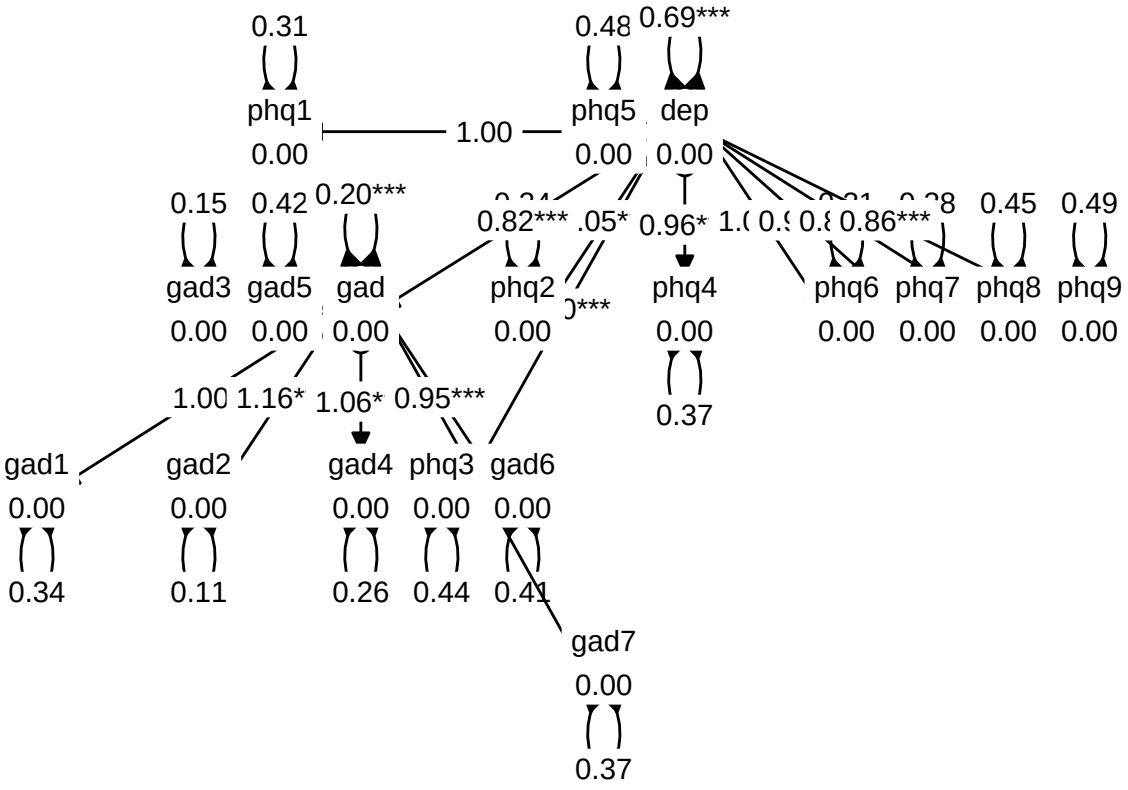
```
##                                                   Standard      Robust
##   Test Statistic                                   443.225     700.051
##   Degrees of freedom                                   103         103
##   P-value (Chi-square)                               0.000       0.000
##   Scaling correction factor                                      0.660
##   Shift parameter                                               28.748
##     simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                                 62067.693   21128.811
##   Degrees of freedom                                   120         120
##   P-value                                            0.000       0.000
##   Scaling correction factor                                      2.949
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                        0.995       0.972
##   Tucker-Lewis Index (TLI)                           0.994       0.967
##
##   Robust Comparative Fit Index (CFI)                                NA
##   Robust Tucker-Lewis Index (TLI)                                   NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                              0.068       0.089
##   90 Percent confidence interval - lower             0.061       0.083
##   90 Percent confidence interval - upper             0.074       0.096
##   P-value RMSEA <= 0.05                              0.000       0.000
##
##   Robust RMSEA                                                      NA
##   90 Percent confidence interval - lower                            NA
##   90 Percent confidence interval - upper                            NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                               0.056       0.056
##
## Parameter Estimates:
##
##   Standard errors                             Robust.sem
##   Information                                   Expected
##   Information saturated (h1) model          Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   gad =~
##     gad1              1.000                               0.814    0.814
##     gad2              1.161    0.022   52.398    0.000    0.946    0.946
##     gad3              1.131    0.022   50.273    0.000    0.921    0.921
##     gad4              1.055    0.023   45.790    0.000    0.859    0.859
##     gad5              0.938    0.030   31.638    0.000    0.764    0.764
##     gad6              0.946    0.029   32.916    0.000    0.770    0.770
##     gad7              0.977    0.029   34.138    0.000    0.796    0.796
```

```
##   dep =~
##     phq1              1.000                                    0.830    0.830
##     phq2              1.054    0.025   41.961    0.000    0.874    0.874
##     phq3              0.898    0.028   32.299    0.000    0.745    0.745
##     phq4              0.957    0.026   36.498    0.000    0.794    0.794
##     phq5              0.866    0.030   28.485    0.000    0.719    0.719
##     phq6              0.998    0.026   38.776    0.000    0.828    0.828
##     phq7              0.948    0.029   32.487    0.000    0.786    0.786
##     phq8              0.892    0.035   25.459    0.000    0.740    0.740
##     phq9              0.862    0.041   20.917    0.000    0.715    0.715
##
## Regressions:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   gad ~
##     dep               0.821    0.027   30.229    0.000    0.836    0.836
##
## Intercepts:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .gad1              0.000                                0.000    0.000
##    .gad2              0.000                                0.000    0.000
##    .gad3              0.000                                0.000    0.000
##    .gad4              0.000                                0.000    0.000
##    .gad5              0.000                                0.000    0.000
##    .gad6              0.000                                0.000    0.000
##    .gad7              0.000                                0.000    0.000
##    .phq1              0.000                                0.000    0.000
##    .phq2              0.000                                0.000    0.000
##    .phq3              0.000                                0.000    0.000
##    .phq4              0.000                                0.000    0.000
##    .phq5              0.000                                0.000    0.000
##    .phq6              0.000                                0.000    0.000
##    .phq7              0.000                                0.000    0.000
##    .phq8              0.000                                0.000    0.000
##    .phq9              0.000                                0.000    0.000
##    .gad               0.000                                0.000    0.000
##     dep               0.000                                0.000    0.000
##
## Thresholds:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     gad1|t1         -0.487    0.049  -10.013    0.000   -0.487   -0.487
##     gad1|t2          0.720    0.051   14.045    0.000    0.720    0.720
##     gad1|t3          1.405    0.068   20.718    0.000    1.405    1.405
##     gad2|t1         -0.113    0.047   -2.412    0.016   -0.113   -0.113
##     gad2|t2          0.779    0.052   14.956    0.000    0.779    0.779
##     gad2|t3          1.414    0.068   20.747    0.000    1.414    1.414
##     gad3|t1         -0.392    0.048   -8.180    0.000   -0.392   -0.392
##     gad3|t2          0.641    0.050   12.765    0.000    0.641    0.641
##     gad3|t3          1.310    0.064   20.322    0.000    1.310    1.310
##     gad4|t1         -0.411    0.048   -8.548    0.000   -0.411   -0.411
##     gad4|t2          0.608    0.050   12.191    0.000    0.608    0.608
##     gad4|t3          1.278    0.063   20.150    0.000    1.278    1.278
##     gad5|t1          0.399    0.048    8.327    0.000    0.399    0.399
##     gad5|t2          1.217    0.062   19.771    0.000    1.217    1.217
##     gad5|t3          1.819    0.089   20.471    0.000    1.819    1.819
```

```
##     gad6|t1          -0.650    0.050  -12.908    0.000   -0.650   -0.650
##     gad6|t2           0.558    0.049   11.323    0.000    0.558    0.558
##     gad6|t3           1.247    0.062   19.966    0.000    1.247    1.247
##     gad7|t1           0.246    0.047    5.228    0.000    0.246    0.246
##     gad7|t2           0.989    0.056   17.696    0.000    0.989    0.989
##     gad7|t3           1.597    0.076   20.984    0.000    1.597    1.597
##     phq1|t1          -0.154    0.047   -3.302    0.001   -0.154   -0.154
##     phq1|t2           0.989    0.056   17.696    0.000    0.989    0.989
##     phq1|t3           1.751    0.085   20.715    0.000    1.751    1.751
##     phq2|t1          -0.211    0.047   -4.488    0.000   -0.211   -0.211
##     phq2|t2           1.053    0.057   18.377    0.000    1.053    1.053
##     phq2|t3           1.676    0.080   20.897    0.000    1.676    1.676
##     phq3|t1          -0.475    0.049   -9.794    0.000   -0.475   -0.475
##     phq3|t2           0.530    0.049   10.815    0.000    0.530    0.530
##     phq3|t3           1.161    0.060   19.355    0.000    1.161    1.161
##     phq4|t1          -0.887    0.054  -16.457    0.000   -0.887   -0.887
##     phq4|t2           0.411    0.048    8.548    0.000    0.411    0.411
##     phq4|t3           1.053    0.057   18.377    0.000    1.053    1.053
##     phq5|t1          -0.243    0.047   -5.154    0.000   -0.243   -0.243
##     phq5|t2           0.571    0.049   11.541    0.000    0.571    0.571
##     phq5|t3           1.270    0.063   20.105    0.000    1.270    1.270
##     phq6|t1          -0.151    0.047   -3.228    0.001   -0.151   -0.151
##     phq6|t2           0.793    0.052   15.164    0.000    0.793    0.793
##     phq6|t3           1.453    0.070   20.850    0.000    1.453    1.453
##     phq7|t1           0.081    0.047    1.744    0.081    0.081    0.081
##     phq7|t2           0.939    0.055   17.117    0.000    0.939    0.939
##     phq7|t3           1.561    0.074   20.986    0.000    1.561    1.561
##     phq8|t1           0.676    0.051   13.337    0.000    0.676    0.676
##     phq8|t2           1.424    0.069   20.775    0.000    1.424    1.424
##     phq8|t3           1.987    0.102   19.570    0.000    1.987    1.987
##     phq9|t1           0.989    0.056   17.696    0.000    0.989    0.989
##     phq9|t2           1.720    0.083   20.802    0.000    1.720    1.720
##     phq9|t3           2.098    0.112   18.775    0.000    2.098    2.098
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .gad1              0.337                               0.337    0.337
##    .gad2              0.106                               0.106    0.106
##    .gad3              0.152                               0.152    0.152
##    .gad4              0.261                               0.261    0.261
##    .gad5              0.417                               0.417    0.417
##    .gad6              0.407                               0.407    0.407
##    .gad7              0.367                               0.367    0.367
##    .phq1              0.312                               0.312    0.312
##    .phq2              0.235                               0.235    0.235
##    .phq3              0.444                               0.444    0.444
##    .phq4              0.370                               0.370    0.370
##    .phq5              0.483                               0.483    0.483
##    .phq6              0.314                               0.314    0.314
##    .phq7              0.382                               0.382    0.382
##    .phq8              0.452                               0.452    0.452
##    .phq9              0.489                               0.489    0.489
##    .gad               0.199    0.018   10.991    0.000    0.300    0.300
##     dep               0.688    0.029   23.826    0.000    1.000    1.000
```

```
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     gad1              1.000                               1.000    1.000
##     gad2              1.000                               1.000    1.000
##     gad3              1.000                               1.000    1.000
##     gad4              1.000                               1.000    1.000
##     gad5              1.000                               1.000    1.000
##     gad6              1.000                               1.000    1.000
##     gad7              1.000                               1.000    1.000
##     phq1              1.000                               1.000    1.000
##     phq2              1.000                               1.000    1.000
##     phq3              1.000                               1.000    1.000
##     phq4              1.000                               1.000    1.000
##     phq5              1.000                               1.000    1.000
##     phq6              1.000                               1.000    1.000
##     phq7              1.000                               1.000    1.000
##     phq8              1.000                               1.000    1.000
##     phq9              1.000                               1.000    1.000
```

```
# Graph
graph_sem(model_fit)
```



It should be noted that there are many different questions and extensions that can be applied with SEM and latent variables, such as modeling longitudinal data or clustering (and a mix of these two).

11.2 Power Analysis Resources

Power analysis for SEM is more complicated than typical statistical tests, and there are various ways to conceptualize and actually determine statistical power in these models (e.g., statistical power for model fit, statistical power for an individual parameter). The following papers may be helpful as an introduction into conducting a power analysis for SEM:

- https://link.springer.com/article/10.3758/s13428-020-01479-0 (includes an online app; power for model fit)
- https://psycnet.apa.org/record/2021-97628-001 (includes an R package)
- https://journals.sagepub.com/doi/full/10.1177/2515245920918253 (includes an online app; power for individual parameters)

12 Multilevel Modeling

Multilevel modeling (otherwise referred to as mixed effects modeling or hierarchical linear modeling in the literature) is a powerful modeling technique that can examine both within-person and between-person effects. It might help to think of a multilevel model as a more powerful and general version of a mixed ANOVA, in the sense that you can look at the effect(s) of a within-person variable (such as time) on some outcome, the effect(s) of a between-person variable (such as diagnosis status) on some outcome, and the interactive effects between these variables. A mixed ANOVA is rather limited such that you can only look at *categorical* within-person and between-person variables, rather than the addition of continuous variables. Multilevel models allow to you incorporate both categorical and continuous variables.

The following examples take data from https://journals.sagepub.com/doi/full/10.1177/21677026211013507 where the authors were interested in looking at daily reports of affect, interpersonal traits (e.g., warmness-coldness), and how these were related to baseline Antagonism scores.

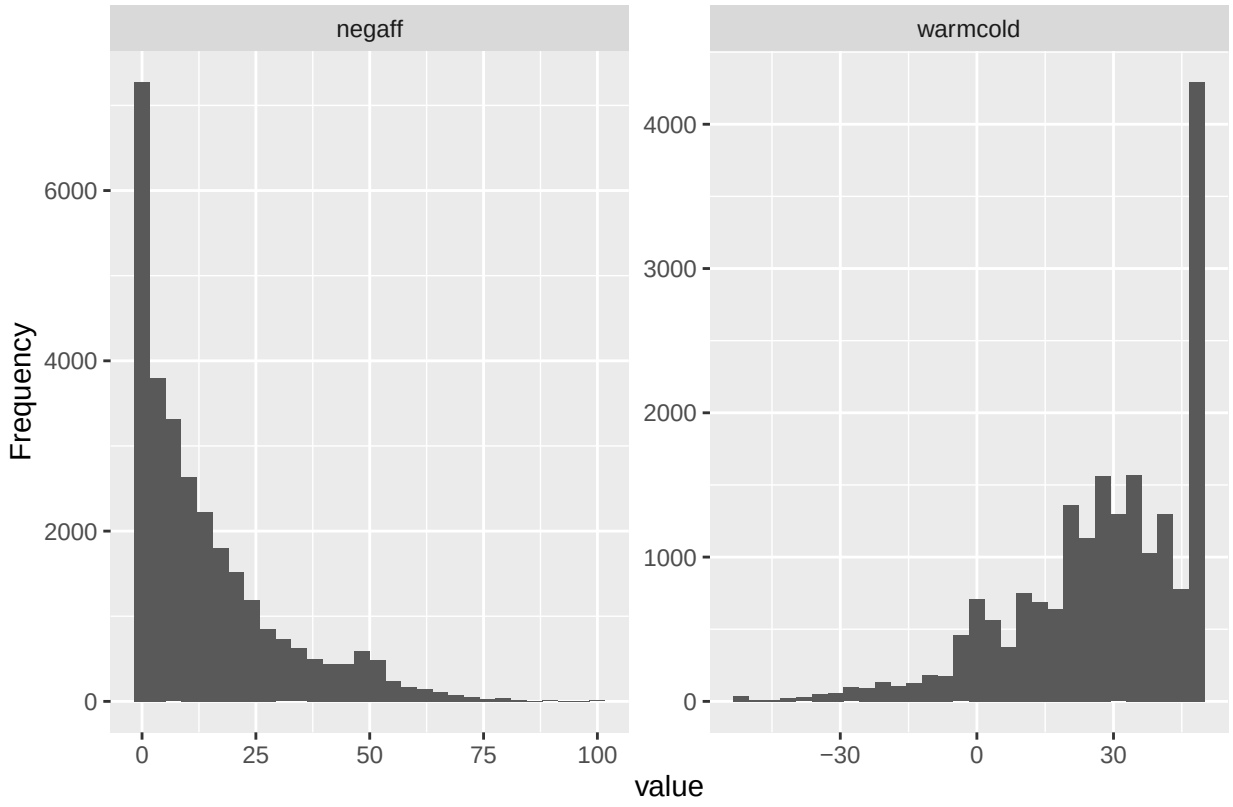
12.1 R

```
library(naniar)
library(tidymodels)

# Importing data
df <- read.csv("S3_S4_pooled.csv", header = F)

# Naming the variables (because columns didn't have variable names)
names <- c(
    "id", "total_ema", "pa_now", "negaff", "domsubother", "warmcoldother", "domsubyou", "warmcold",
    "emp1", "emp2", "emp3", "lonely", "antagonism", "man", "dec", "grand", "att", "call", "hos"
)
names(df) <- names

# Selecting variables of interest and recoding NAs
df <- df %>%
    dplyr::select(id, negaff, warmcold, antagonism) %>%
    replace_with_na_all(condition = ~ .x == 9999)
```

With multilevel models, it helps to do something called 'person mean-centering' on the within-person variables (i.e., the variables that a participant provides multiple daily ratings on). This involves calculating each person's *mean score* on a given variable across all of the time points, and then subtracting this mean score from *each* daily score. The reason this is done is to make the results more interpretable and to aid in disentangling within-person and between-person effects in daily associations. What this means is that, for example, you might be interested in the daily associations between daily negative affect and daily interpersonal warmth. This daily association can be broken down into a within-person effect (i.e., do deviations from your average daily affect predict interpersonal warmth) and a between-person effect (i.e., does average daily affect predict interpersonal warmth). Below is R code for calculating each person's mean on warmth and negative affect, and for centering each time point's score.

```
library(DataExplorer)

# Calculating new variables
df <- df %>%
    group_by(id) %>%
    mutate(
        negaff_mean = mean(negaff, na.rm = T),
        warmcold_mean = mean(warmcold, na.rm = T),
        negaff_c = negaff - mean(negaff, na.rm = T),
```

```
        warmcold_c = warmcold - mean(warmcold, na.rm = T)
    ) %>%
    ungroup()

# Graphs
df %>%
    dplyr::select(negaff, warmcold) %>%
    plot_histogram()
```



Now, we will estimate a few different multilevel models that answer different questions. For all of these models, the outcome of interest is daily interpersonal warmth. The first model is called an intercept-only model, which basically tries to estimate what the **average** daily interpersonal warmth is across the whole sample. The resulting coefficient tells you the average interpersonal warmth score across all time points and across every participant, and whether this average score is significantly different from zero. The second model involves using baseline Antagonism to predict average daily warmth - that is, do those with more Antagonism tend to have more or less daily interpersonal warmth? The third model estimates both within-person effects and between-person effects in the daily associations between daily negative affect and daily interpersonal warmth - that is, do deviations from average negative affect predict daily interpersonal warmth (within-person effect) and do averaged daily negative affect scores predict daily interpersonal warmth (between-person effect)? The last model is similar to what is usually done with a mixed ANOVA, where you wou might be interested in the interaction between a within-person variable (in this case negative affect across time) and a between-person variable (in this case baseline Antagonism). In multilevel modeling jargon, this interaction is called a cross-level interaction.

```
library(jtools)
library(interactions)
library(afex)
library(performance)
library(parameters)
```

```r
# Intercept-only model for warm-cold
model1 <- lmer(warmcold ~ 1 + (1 | id), data = df)
summ(model1)
```

Observations	19649
Dependent variable	warmcold
Type	Mixed effects linear regression

AIC	166498.49
BIC	166522.15
Pseudo-R^2 (fixed effects)	0.00
Pseudo-R^2 (total)	0.34

Fixed Effects

	Est.	S.E.	t val.	d.f.	p
(Intercept)	26.95	0.47	57.55	648.23	0.00

p values calculated using Satterthwaite d.f.

Random Effects

Group	Parameter	Std. Dev.
id	(Intercept)	11.54
Residual		16.00

Grouping Variables

Group	# groups	ICC
id	669	0.34

```r
model_parameters(model1)
```

```
## # Fixed Effects
##
## Parameter   | Coefficient |   SE |         95% CI | t(19646) |      p
## ------------------------------------------------------------------------
## (Intercept) |       26.95 | 0.47 | [26.03, 27.87] |    57.55 | < .001
##
## # Random Effects
##
## Parameter          | Coefficient
## ---------------------------------
## SD (Intercept: id) |       11.54
## SD (Residual)      |       16.00
```

```r
model_performance(model1)
```

```
## # Indices of model performance
##
## AIC       |      AICc |       BIC | R2 (cond.) | R2 (marg.) |   ICC |   RMSE |  Sigma
```

```
## ---------------------------------------------------------------------------------------
## 1.665e+05 | 1.665e+05 | 1.665e+05 |      0.342 |      0.000 | 0.342 | 15.753 | 16.003
```

```
icc(model1)
```

```
## # Intraclass Correlation Coefficient
##
##     Adjusted ICC: 0.342
##   Unadjusted ICC: 0.342
```

```
# Baseline Antagonism predicting daily warm-cold
model2 <- lmer(warmcold ~ 1 + antagonism + (1 | id), data = df)
summ(model2)
```

Observations	19649
Dependent variable	warmcold
Type	Mixed effects linear regression

AIC	166464.43
BIC	166495.97
Pseudo-R^2 (fixed effects)	0.02
Pseudo-R^2 (total)	0.34

Fixed Effects

	Est.	S.E.	t val.	d.f.	p
(Intercept)	29.79	0.66	45.04	642.57	0.00
antagonism	-5.74	0.97	-5.93	649.01	0.00

p values calculated using Satterthwaite d.f.

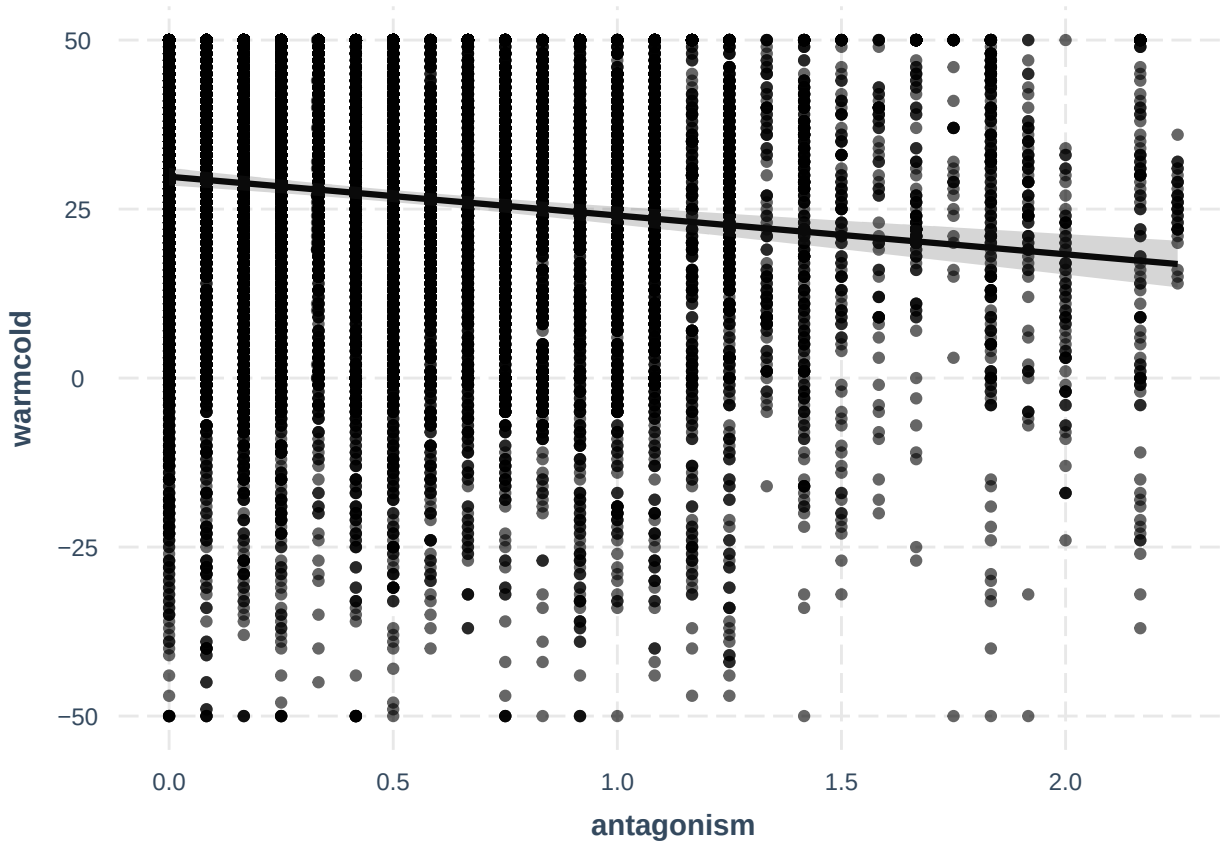
Random Effects

Group	Parameter	Std. Dev.
id	(Intercept)	11.23
Residual		16.00

Grouping Variables

Group	# groups	ICC
id	669	0.33

```
effect_plot(model = model2, pred = antagonism, plot.points = T, interval = T)
```




```
# Within- and between-person associations between negative affect and warm-cold
model3 <- lmer(warmcold ~ 1 + antagonism + negaff_mean + negaff_c + (1 | id), data = df)
model_parameters(model3)
```

```
## # Fixed Effects
##
## Parameter    | Coefficient |   SE |         95% CI | t(19642) |      p
## ------------------------------------------------------------------------
## (Intercept) |       34.28 | 0.74 | [32.82, 35.74] |    46.13 | < .001
## antagonism  |       -3.40 | 0.92 | [-5.20, -1.59] |    -3.69 | < .001
## negaff mean |       -0.38 | 0.04 | [-0.45, -0.31] |   -10.82 | < .001
## negaff c    |       -0.41 | 0.01 | [-0.43, -0.39] |   -39.99 | < .001
##
## # Random Effects
##
## Parameter          | Coefficient
## -------------------------------
## SD (Intercept: id) |       10.31
## SD (Residual)      |       15.37
```

```
summ(model3)
```

Observations	19648
Dependent variable	warmcold
Type	Mixed effects linear regression

AIC	164826.09
BIC	164873.40
Pseudo-R^2 (fixed effects)	0.12
Pseudo-R^2 (total)	0.39

Fixed Effects

	Est.	S.E.	t val.	d.f.	p
(Intercept)	34.28	0.74	46.13	632.75	0.00
antagonism	-3.40	0.92	-3.69	649.12	0.00
negaff_mean	-0.38	0.04	-10.82	657.59	0.00
negaff_c	-0.41	0.01	-39.99	19057.11	0.00

p values calculated using Satterthwaite d.f.

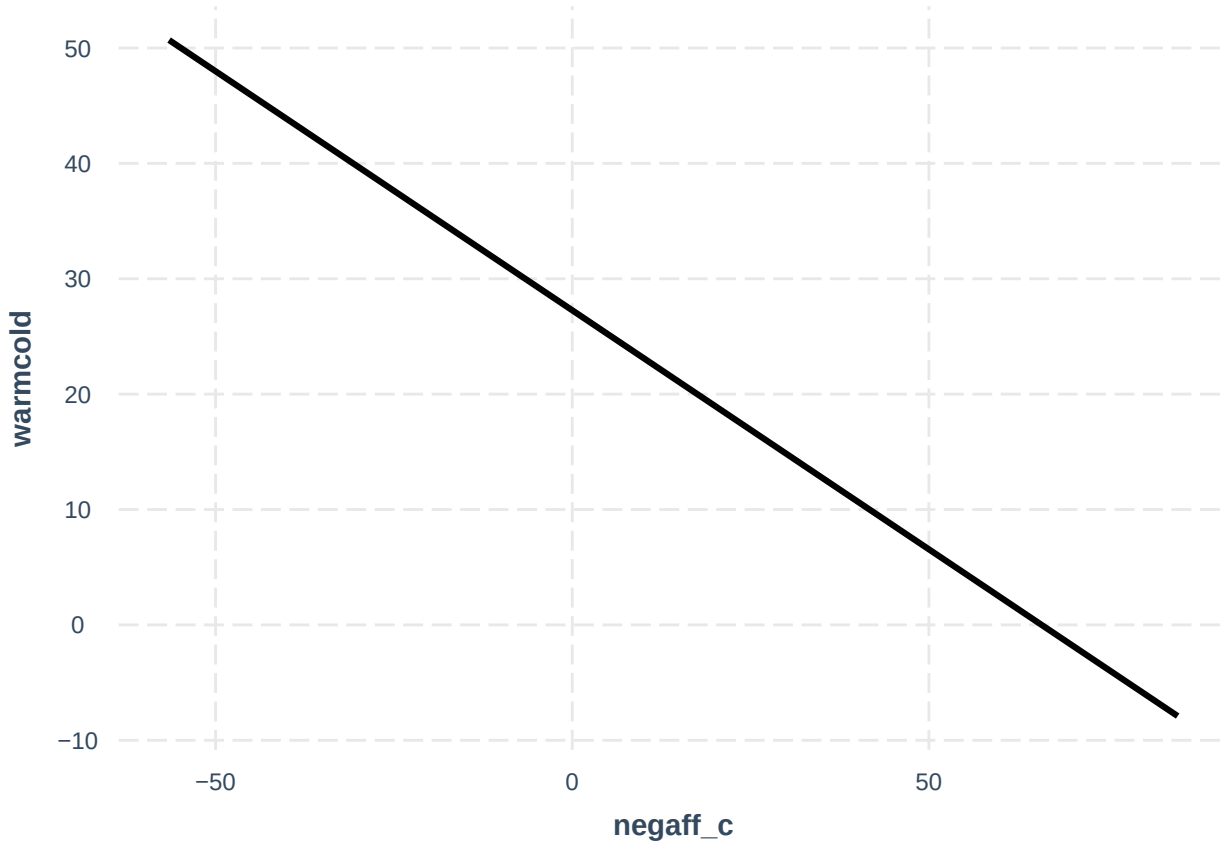
Random Effects

Group	Parameter	Std. Dev.
id	(Intercept)	10.31
Residual		15.37

Grouping Variables

Group	# groups	ICC
id	669	0.31

```
effect_plot(model = model3, pred = negaff_c)
```




```
# Cross-level interaction between antagonism and negative affect predicting warm-cold
model4 <- lmer(warmcold ~ 1 + antagonism + negaff_mean + negaff_c
    + antagonism:negaff_c + antagonism:negaff_mean + (1 | id), data = df)
summ(model4)
```

Observations	19648
Dependent variable	warmcold
Type	Mixed effects linear regression

AIC	164834.30
BIC	164897.39
Pseudo-R^2 (fixed effects)	0.12
Pseudo-R^2 (total)	0.39

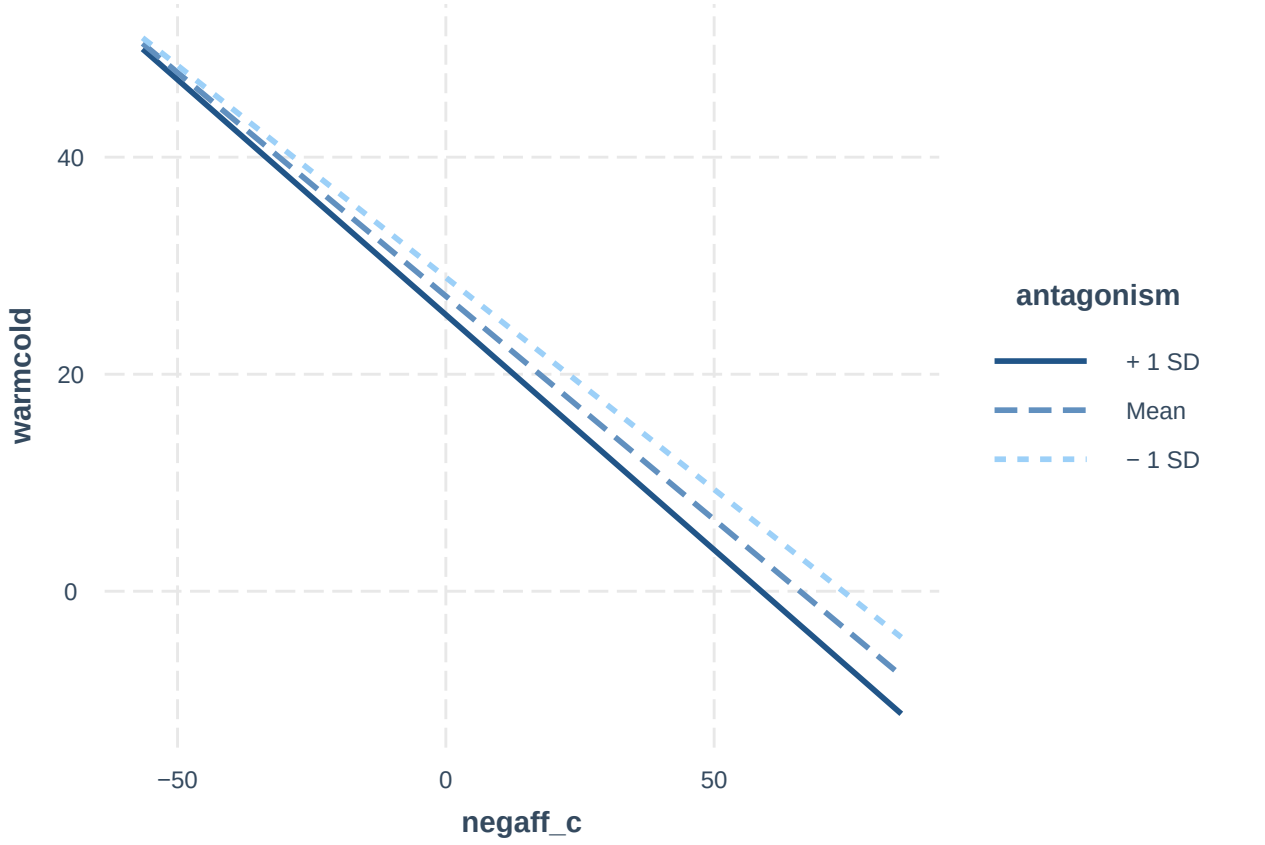
Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	34.76	0.95	36.54	630.81	0.00
antagonism	-4.36	1.49	-2.93	639.91	0.00
negaff_mean	-0.41	0.05	-7.81	659.19	0.00
negaff_c	-0.39	0.02	-25.48	19060.18	0.00
antagonism:negaff_c	-0.04	0.02	-2.15	19060.84	0.03
antagonism:negaff_mean	0.05	0.07	0.80	656.10	0.42

p values calculated using Satterthwaite d.f.

Random Effects		
Group	Parameter	Std. Dev.
id	(Intercept)	10.31
Residual		15.37

Grouping Variables		
Group	# groups	ICC
id	669	0.31

```
interact_plot(model4, pred = negaff_c, modx = antagonism)
```



12.2 Power Analysis Resources

Power analysis for multilevel models is also tricky, but there are a few recent papers that discuss how to conduct a power analysis:

- https://journals.sagepub.com/doi/10.1177/2515245920978738 (includes an app)
- https://psycnet.apa.org/record/2022-22670-001

13 R packages used

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.0.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] parameters_0.19.0  tidySEM_0.2.3      OpenMx_2.20.7      semTools_0.5-5
##  [5] lavaan_0.6-12      coin_1.4-2         survival_3.2-13    mediation_4.5.0
##  [9] sandwich_3.0-2     mvtnorm_1.1-3      MASS_7.3-54        interactions_1.1.5
## [13] afex_1.0-1         lme4_1.1-27.1      Matrix_1.3-4       foreign_0.8-81
## [17] jmv_2.3.4          modelbased_0.7.0.1 ggpubr_0.4.0       effectsize_0.8.1
## [21] pwr_1.3-0          see_0.7.0.1        boot_1.3-28        jtools_2.1.4
## [25] skimr_2.1.3        performance_0.10.0 correlation_0.8.2  simputation_0.2.8
## [29] naniar_0.6.1       car_3.0-12         carData_3.0-4      DataExplorer_0.8.2
## [33] datawizard_0.6.2   psych_2.1.9        yardstick_0.0.9    workflowsets_0.1.0
## [37] workflows_0.2.4    tune_0.1.6         tidyr_1.1.4        tibble_3.1.6
## [41] rsample_0.1.1      recipes_0.1.17     purrr_0.3.4        parsnip_0.1.7
## [45] modeldata_0.1.1    infer_1.0.0        ggplot2_3.3.5      dplyr_1.0.10
## [49] dials_0.0.10       scales_1.2.1       broom_0.7.10       tidymodels_0.1.4
##
## loaded via a namespace (and not attached):
##   [1] estimability_1.3      coda_0.19-4           nonnest2_0.5-5
##   [4] visdat_0.5.3          knitr_1.37            multcomp_1.4-20
##   [7] data.table_1.14.2     rpart_4.1-15          inline_0.3.19
##  [10] hardhat_0.1.6         generics_0.1.1        GPfit_1.0-8
##  [13] callr_3.7.0           TH.data_1.1-1         future_1.23.0
##  [16] webshot_0.5.4         xml2_1.3.3            lubridate_1.8.0
##  [19] StanHeaders_2.21.0-7  gower_0.2.2           xfun_0.30
##  [22] bayesplot_1.9.0       evaluate_0.14         fansi_0.5.0
##  [25] igraph_1.2.10         tmvnsim_1.0-2         htmlwidgets_1.5.4
##  [28] stats4_4.1.2          ellipsis_0.3.2        backports_1.4.1
##  [31] pbivnorm_0.6.0        insight_0.18.8        RcppParallel_5.1.5
##  [34] libcoin_1.0-9         deldir_1.0-6          jmvcore_2.3.12
##  [37] vctrs_0.5.1           abind_1.4-5           withr_2.4.3
##  [40] checkmate_2.1.0       emmeans_1.7.1-1       prettyunits_1.1.1
##  [43] mnormt_2.0.2          svglite_2.1.0         cluster_2.1.2
##  [46] crayon_1.4.2          pkgconfig_2.0.3       labeling_0.4.2
##  [49] nlme_3.1-153          vipor_0.4.5           nnet_7.3-16
##  [52] rlang_1.0.6           globals_0.14.0        lifecycle_1.0.3
##  [55] fastDummies_1.6.3     matrixStats_0.61.0    loo_2.5.1
##  [58] zoo_1.8-10            base64enc_0.1-3       beeswarm_0.4.0
```

```
##  [61] ggridges_0.5.3        processx_3.5.2        png_0.1-7
##  [64] viridisLite_0.4.0     texreg_1.38.6         pROC_1.18.0
##  [67] pander_0.6.4          stringr_1.4.0         parallelly_1.30.0
##  [70] jpeg_0.1-9            rstatix_0.7.0         ggsignif_0.6.3
##  [73] lpSolve_5.6.17        magrittr_2.0.3        plyr_1.8.6
##  [76] compiler_4.1.2        rstantools_2.2.0      kableExtra_1.3.4
##  [79] RColorBrewer_1.1-2    cli_3.4.1             lmerTest_3.1-3
##  [82] DiceDesign_1.9        listenv_0.8.0         patchwork_1.1.1
##  [85] ps_1.6.0              htmlTable_2.4.0       Formula_1.2-4
##  [88] mgcv_1.8-38           tidyselect_1.1.1      stringi_1.7.6
##  [91] highr_0.9             yaml_2.2.1            norm_1.0-9.5
##  [94] latticeExtra_0.6-30   ggrepel_0.9.1         grid_4.1.2
##  [97] tools_4.1.2           future.apply_1.8.1    parallel_4.1.2
## [100] rstudioapi_0.13       foreach_1.5.1         gridExtra_2.3
## [103] prodlim_2019.11.13    farver_2.1.0          digest_0.6.29
## [106] lava_1.6.10           proto_1.0.0           networkD3_0.4
## [109] Rcpp_1.0.7            httr_1.4.2            colorspace_2.0-2
## [112] blavaan_0.4-3         rvest_1.0.2           splines_4.1.2
## [115] systemfonts_1.0.3     xtable_1.8-4          jsonlite_1.7.2
## [118] nloptr_1.2.2.3        timeDate_3043.102     rstan_2.21.5
## [121] UpSetR_1.4.0          modeltools_0.2-23     ipred_0.9-12
## [124] R6_2.5.1              Hmisc_4.7-2           lhs_1.1.3
## [127] gsubfn_0.7            pillar_1.6.4          htmltools_0.5.2
## [130] glue_1.6.2            fastmap_1.1.0         minqa_1.2.4
## [133] class_7.3-19          codetools_0.2-18      pkgbuild_1.3.1
## [136] furrr_0.2.3           utf8_1.2.2            lattice_0.20-45
## [139] pbkrtest_0.5.1        numDeriv_2016.8-1.1   ggbeeswarm_0.6.0
## [142] MplusAutomation_1.1.0 interp_1.1-3          CompQuadForm_1.4.3
## [145] rmarkdown_2.13        repr_1.1.3            munsell_0.5.0
## [148] iterators_1.0.13      reshape2_1.4.4        gtable_0.3.0
## [151] bayestestR_0.13.0
```