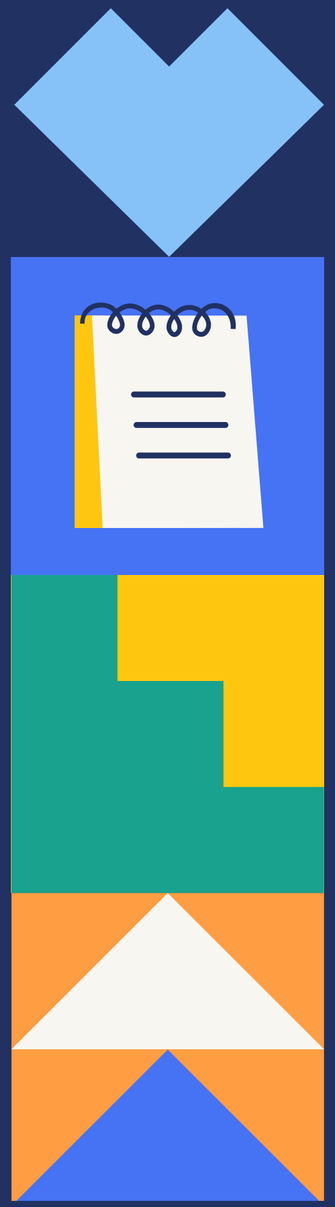
Diagnosing and Curing Data Problems (Using R Statistical Software!)

Alexia Samiotis
PhD Candidate (Clinical Neuropsychology)
Monash Epworth Rehabilitation Research Centre

EPWORTH RESEARCH WEEK 2025 | 6.8.25

https://github.com/AlexiaSam/R-Intro/blob/main_Epworth_Research_Week_Intro.md



Outline

R Crash Course

Link to code

Diagnosing Data Problems

- Visualising
- Descriptives
- Missing Data
- Outliers

Curing Data Problems

- Handling missing data
- Handling Outliers
- Visualising relationships

R Crash Course

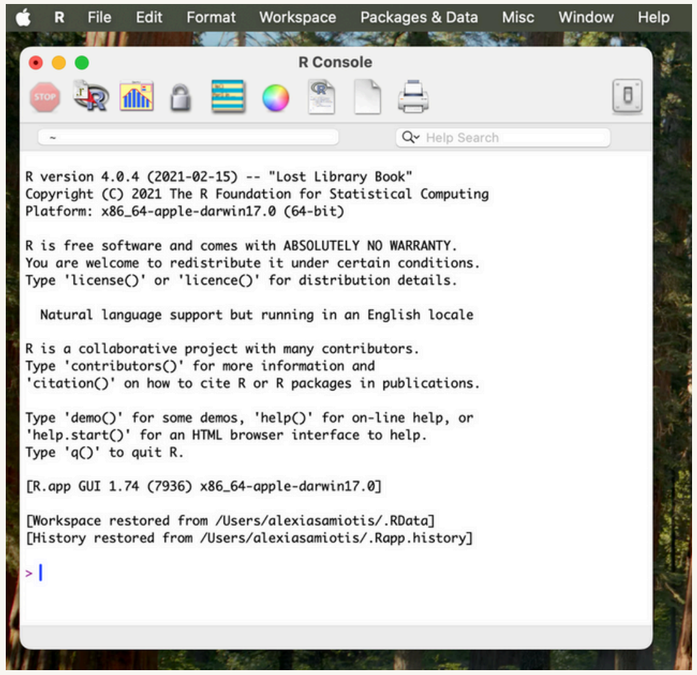
Downloading R and R Studio to Mac if needed

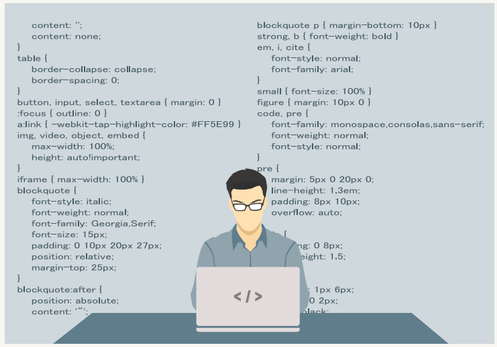
How to Install R on Windows, Mac OS X, and Ubuntu Tutorial

This is a beginner guide that is designed to save yourself a headache and valuable time if you deci...

DataCamp / Mar 11, 2020

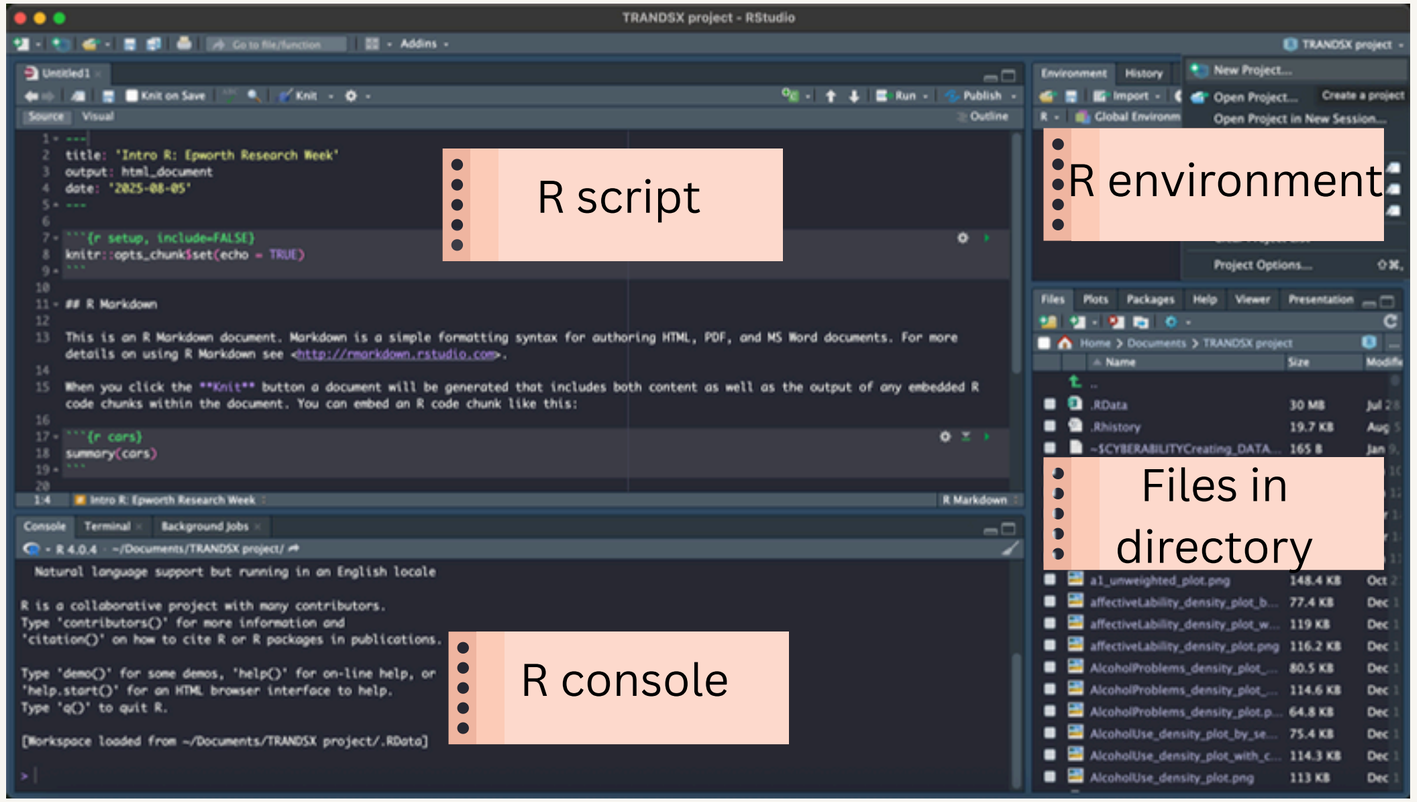
R - computer programming language





R Crash Course

R Studio - An application that provides an interface that looks the same on Mac/Windows and is more user friendly



Downloading to R Studio to Mac if needed

How to Install R on Windows, Mac OS X, and Ubuntu Tutorial

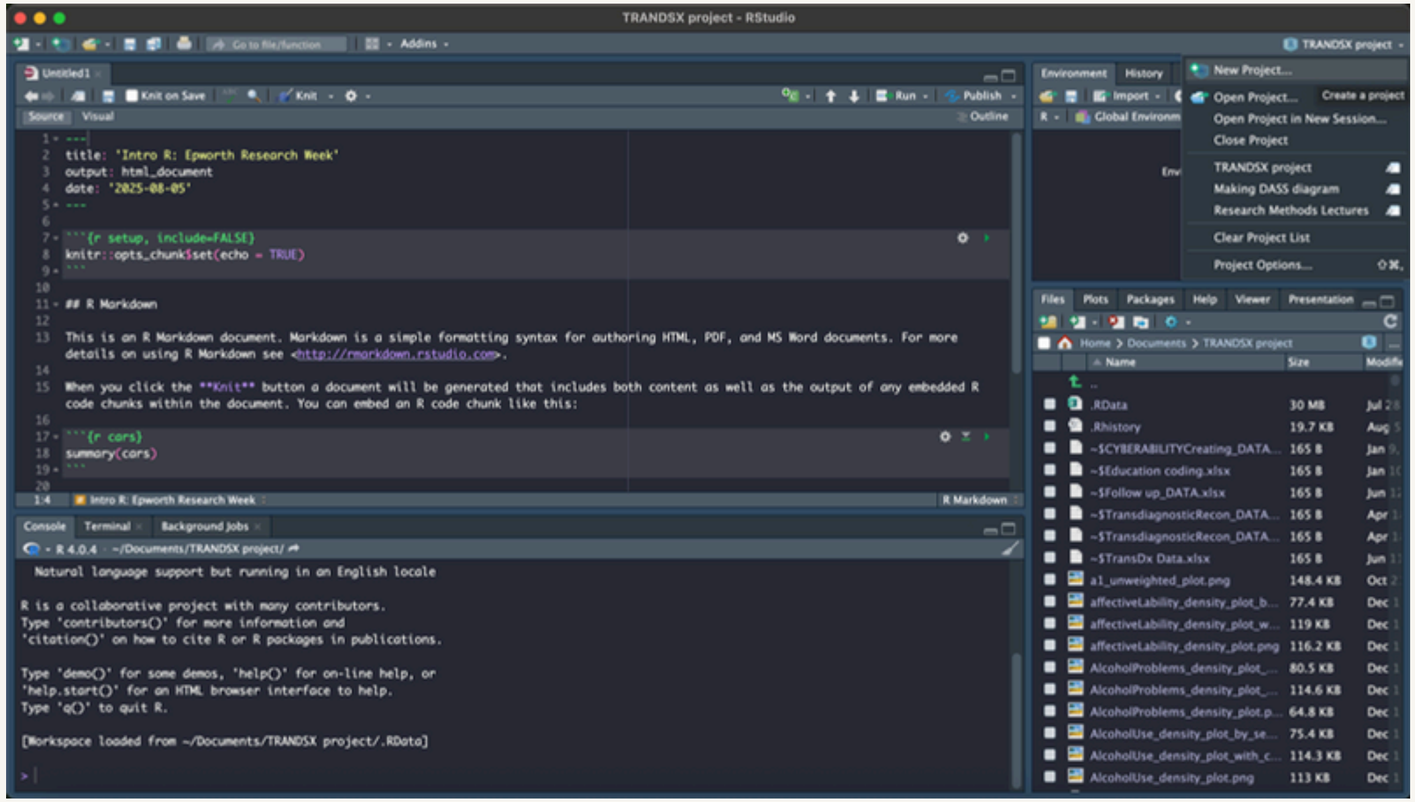
This is a beginner guide that is designed to save yourself a headache and valuable time if you deci...

DataCamp / Mar 11, 2020

R Crash Course

Setting Up

1. New Project



Downloading to R Studio to Mac if needed

How to Install R on Windows, Mac OS X, and Ubuntu Tutorial

This is a beginner guide that is designed to save yourself a headache and valuable time if you deci...

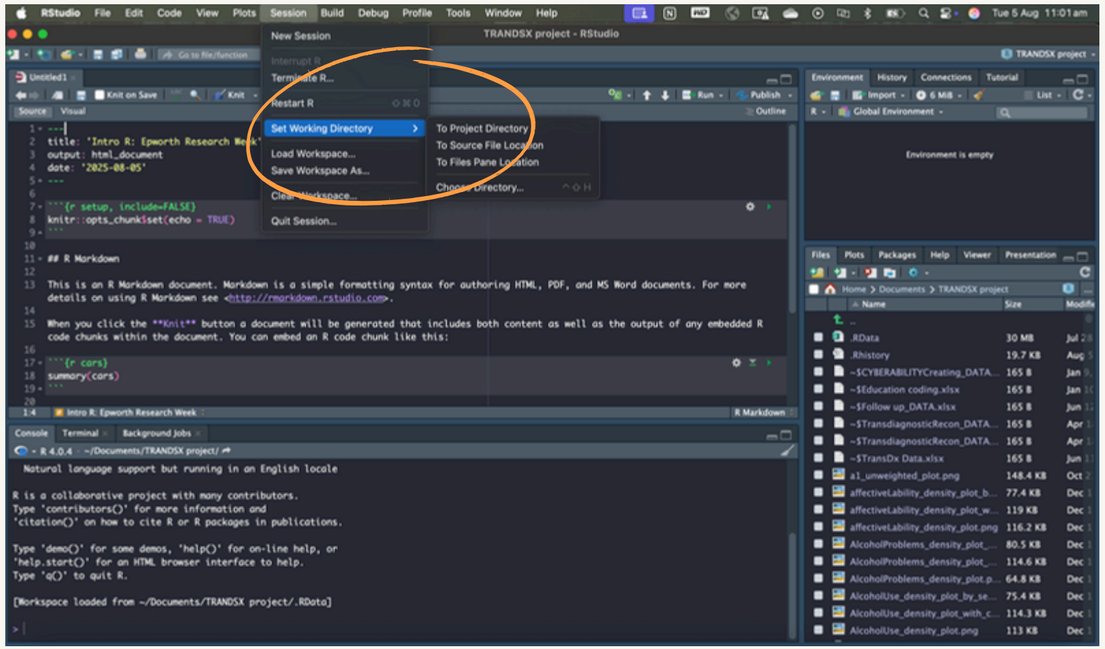
DataCamp / Mar 11, 2020

R Crash Course

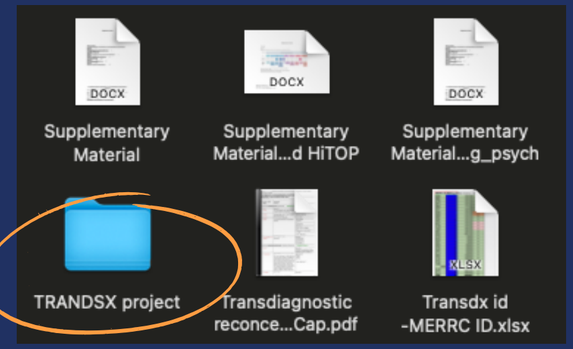
Setting Up

1. New Project
2. Set Working Directory

Macintosh HD › Users › alexiasamiotis › Documents › TRANDSX project



Create a new folder on your computer itself (NOT OneDrive or iCloud)

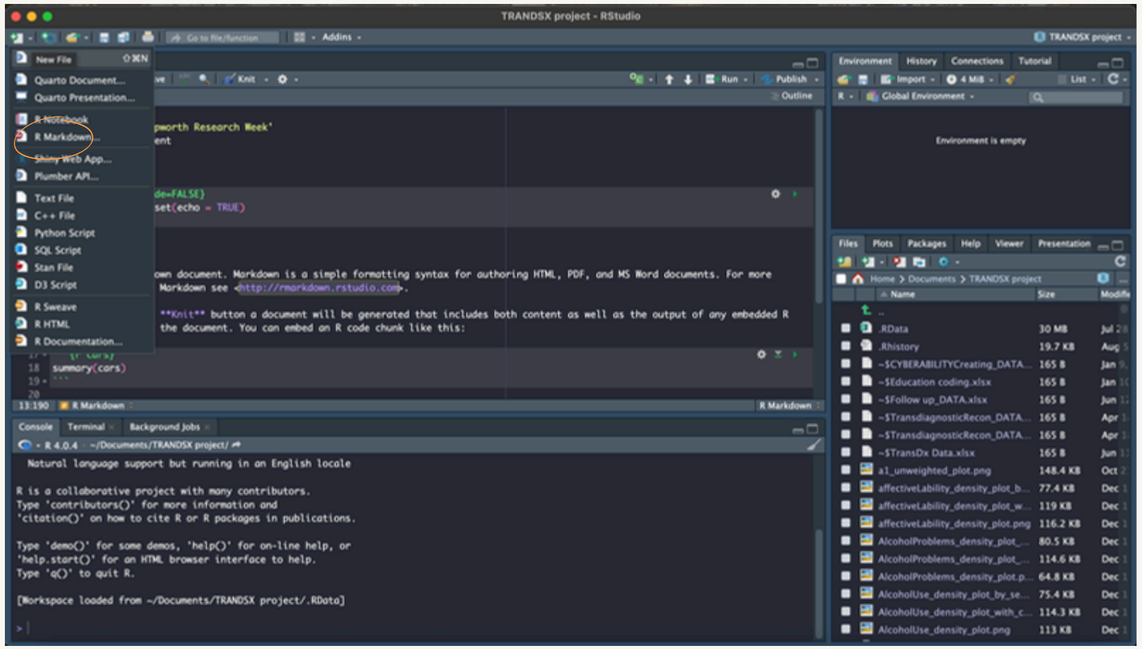


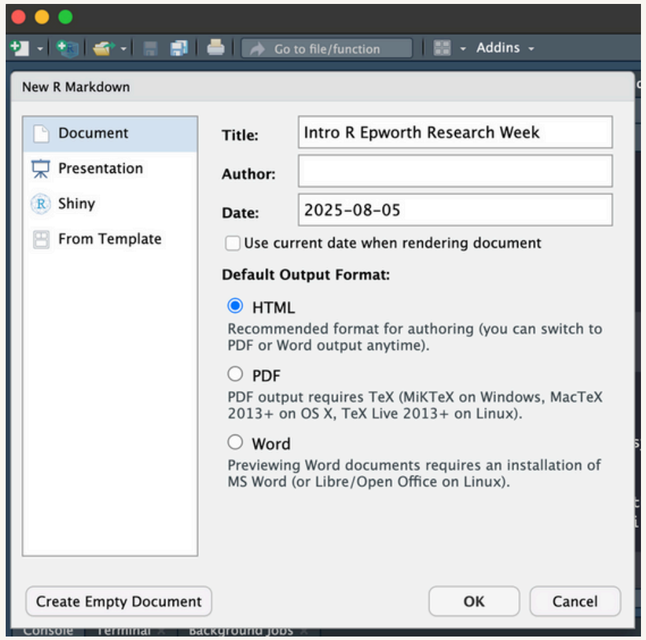
Set this as your project directory. Mine is in my Documents folder.

R Crash Course

Setting Up

1. New Project
2. Set Working Directory
3. Open new Markdown file





Now we create a new file within R. I prefer Markdown.

R Markdown

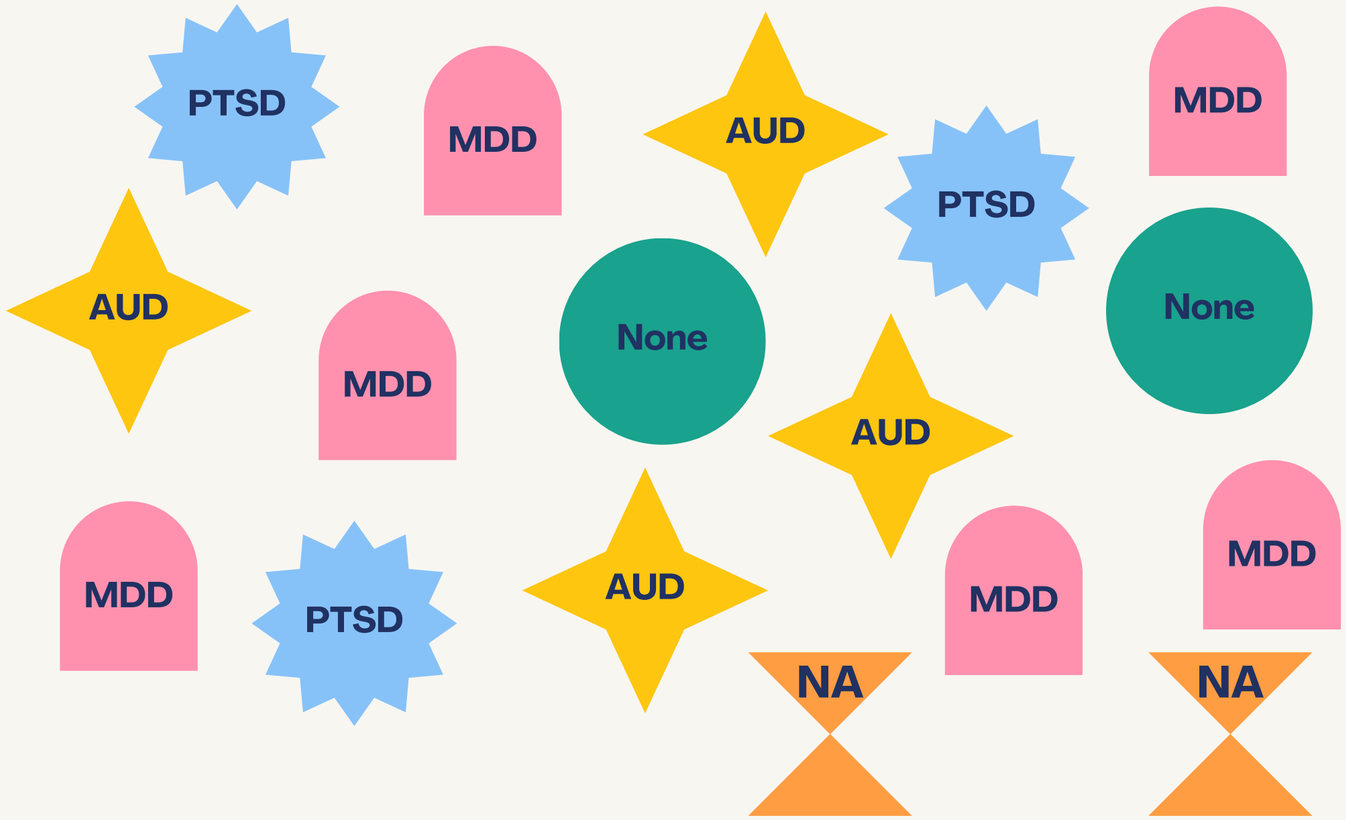
Turn your analyses into high quality documents, reports, presentations and dashboards with R Markdown. Use a productive notebook interface to weave together narrative text and code to produce elegantly formatted...

rstudio.com

You can knit it at the end to a nice looking file, like the one I made for today.

Diagnosing Problems: Discreet Data

Psychologists interviewed individuals with TBI and these are the diagnoses that were provided:

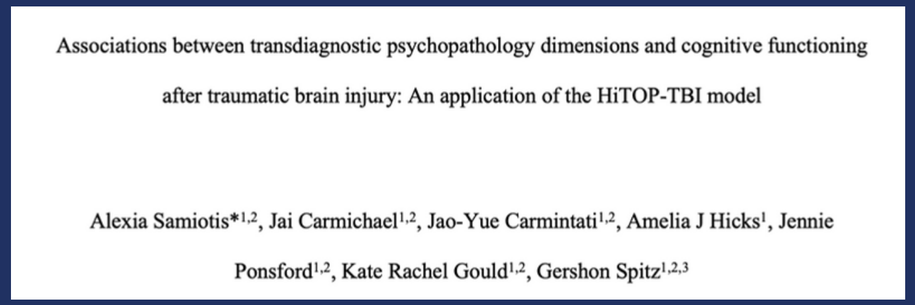


How can we first organise and present the data in R?

Associations between transdiagnostic psychopathology dimensions and cognitive functioning after traumatic brain injury: An application of the HiTOP-TBI model

Alexia Samiotis*[1,2], Jai Carmichael[1,2], Jao-Yue Carmintati[1,2], Amelia J Hicks[1], Jennie Ponsford[1,2], Kate Rachel Gould[1,2], Gershon Spitz[1,2,3]

Importing data



Associations between transdiagnostic psychopathology dimensions and cognitive functioning after traumatic brain injury: An application of the HiTOP-TBI model

Alexia Samiotis*[1,2], Jai Carmichael[1,2], Jao-Yue Carmintati[1,2], Amelia J Hicks[1], Jennie Ponsford[1,2], Kate Rachel Gould[1,2], Gershon Spitz[1,2,3]

Discrete Data

Represent the frequency of different results in the data.

So how to we get this to R Studio?

You should first have a worksheet where you entered the data

result or data value

frequency

DSM Diagnosis	Number of Individuals with TBI
MDD	5
AUD	12
⋮	⋮

1.Installing packages

```
## Installing Packages
These include functions that you will need to execute basic commands like importing you data or creating basic plots, exporting data.You only have to install these ONCE, not every time you open R (thankfully!). In fact, if you try and install them more than once you will effectively just be updating the package (which is fine just not necessary).

```{r packages}
install.packages("readxl") # needed to import data from excel sheets
install.packages("tidyverse") # meta-package with other packages I often use
install.packages("tidyr") # for data tidying
install.packages("lubridate") # for data wrangling
install.packages("writexl") # for exporting to excel
install.packages("ggplot2") # visualisation, plots
install.packages("psych") # psych stats stuff :)
install.packages("openxlsx") # used sometimes when readxl doesn't work
```
```

I like my R Studio interface with these colours, you can change yours to have a white background if you like

Tools > Options > Environment > General and choose a colour theme from the "Colour theme" dropdown menu

2.Opening packages

```
## Load Relevant Libraries
You can think of intalling packages like downloading an app, once it's done it will be there. This doesn't automatically mean the app will open, you still have to click on it. In R, you have to tell the package to open by loading it in the environment. You do have to load packages EVERY time you use R Studio. If you stay in the same session and don't exit R, you don't have to load them.

```{r libraries, echo=FALSE}
library(readxl)
library(tidyverse)
library(tidyr)
library(lubridate)
library(writexl)
library(ggplot2)
library(psych)
library(openxlsx)
```
```

Link to code

3.Importing and naming your data

I like my R Studio interface with these colours, you can change yours to have a white background if you like

```
45 ## Import the dataset
46 This is unpublished data that I have collected as part of my PhD looking at mental health after Traumatic Brain Injury. As you can
   see there >1000 variables, many of which will be included in the analyses for my studies. Today we will only be focusing on a few of
   these. I will show you how to call on the variables you would like to use.
47
48 ```{r transdiagnostic data}
49 df <- read_excel("HiTOP_followup_data.xlsx") # I called this df but can call it something more meaningful if you like.
50 view(df) # lets have a look at what our imported dataset looks like
51 ```
```

Naming: what would you like to name the dataframe? name <-

Tools > Options > Environment > General and choose a colour theme from the "Colour theme" dropdown menu

Link to code

https://github.com/AlexiaSam/R-Intro/blob/main_Epworth_Research_Week_Intro.md

3.Importing and naming your data

This is the function to import the data (found in the readxl package we loaded earlier)

Type the name of your excel file (must be verbatim)

```
45 ## Import the dataset
46 This is unpublished data that I have collected as part of my PhD looking at mental health after Traumatic Brain Injury. As you can
   see there >1000 variables, many of which will be included in the analyses for my studies. Today we will only be focusing on a few of
   these. I will show you how to call on the variables you would like to use.
47
48 ```{r transdiagnostic data}
49 df <- read_excel("HiTOP_followup_data.xlsx") # I called this df but can call it something more meaningful if you like.
50 view(df) # lets have a look at what our imported dataset looks like
51 ```
```

Naming: what would you like to name the dataframe? name <-

I like my R Studio interface with these colours, you can change yours to have a white background if you like

Tools > Options > Environment > General and choose a colour theme from the "Colour theme" dropdown menu

Link to code

https://github.com/AlexiaSam/R-Intro/blob/main_Epworth_Research_Week_Intro.md

3.Importing and naming your data

I like my R Studio interface with these colours, you can change yours to have a white background if you like

This is the function to import the data (found in the readxl package we loaded earlier)

Type the name of your excel file (must be verbatim)

```
## Import the dataset
This is unpublished data that I have collected as part of my PhD looking at mental health after Traumatic Brain Injury. As you can see there >1000 variables, many of which will be included in the analyses for my studies. Today we will only be focusing on a few of these. I will show you how to call on the variables you would like to use.

```{r transdiagnostic data}
df <- read_excel("HiTOP_followup_data.xlsx") # I called this df but can call it something more meaningful if you like.
view(df) # lets have a look at what our imported dataset looks like
```
```

Naming: what would you like to name the dataframe? name <-

Let's have a look at our dataframe

Tools > Options > Environment > General and choose a colour theme from the "Colour theme" dropdown menu

Untitled1* | df

Filter | Cols: « < 1 - 50 > »

	id	mode	participant_id_complete	eligibility	explanatory_statement_complete	background_information_timestamp	no_head_injuries	accomodat
1	1	1	2	1	2	2024-01-10 13:04:22	1	
2	2	1	2	1	2	2024-01-19 20:38:13	1	
3	3	1	2	1	2	2024-01-13 08:19:14	1	
4	4	1	2	1	2	2024-01-17 10:00:01	2	
5	5	1	2	1	2	2024-01-11 22:18:24	1	
6	6	1	2	1	2	2024-01-15 07:07:37	1	
7	7	1	2	1	2	2024-01-16 12:47:01	2	
8	8	1	2	1	2	2024-01-17 11:03:49	1	
9	9	1	2	1	2	2024-01-17 14:39:52	1	
10	10	1	2	1	2	2024-01-23 16:03:03	5	
11	11	1	2	1	2	2024-01-17 12:55:29	4	
12	12	1	2	1	2	2024-01-16 09:50:17	1	
13	13	1	2	1	2	2024-01-21 11:22:38	1	
14	14	1	2	1	2	2024-01-24 09:49:11	1	
15	15	1	2	1	2	2024-01-29 11:25:55	1	
16	16	1	2	1	2	2024-02-04 22:55:38	1	

Showing 1 to 16 of 99 entries, 1591 total columns

4.Calling on specific variables

$ is used to call the variable

```
## Calling on specific variables
I'm sure you don't want to go searching through 1000 variables, so instead you can call on the variable you need and then work with it. You can also perform functions on a vector of variables you call on (1). Alternatively, you can filter a subset of variables that you would like to work with (2). I'll show you both.

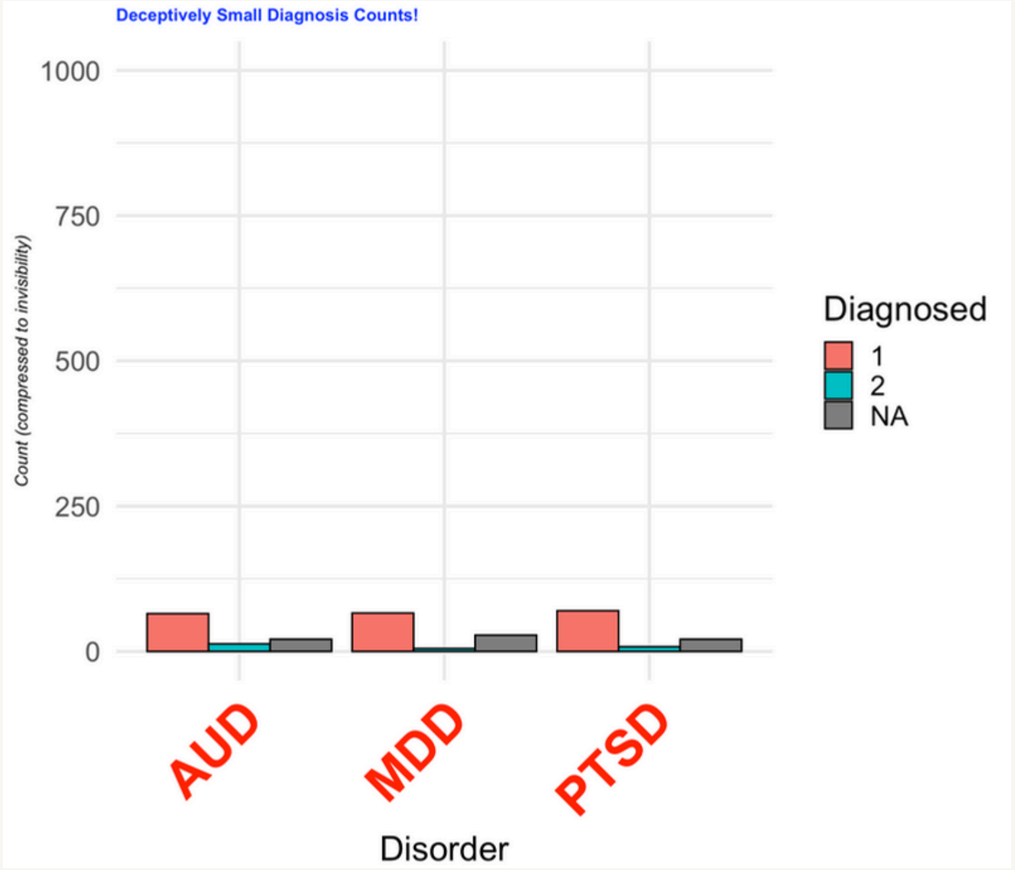
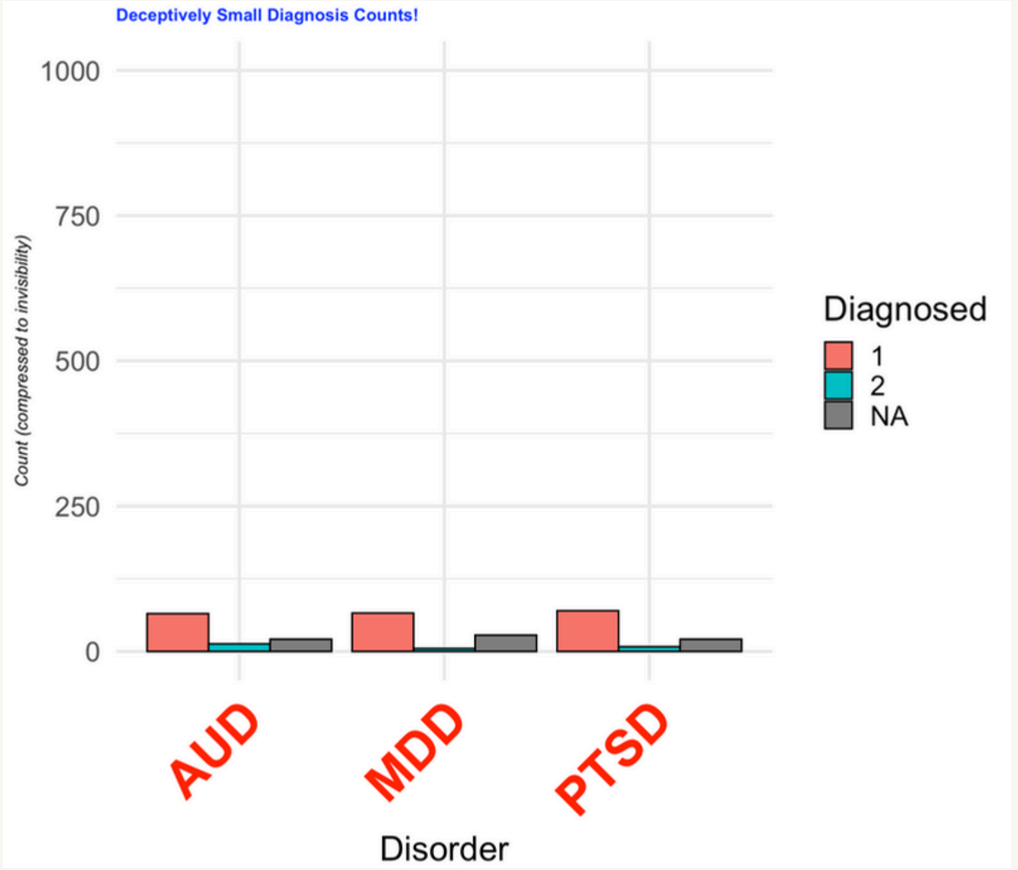
```{r calling on variables}
df$mini_mdd_current # what does our Major Depressive Disorder variable contain? Let's have a quick look by printing it here.
df$mini_alcohol_current # AUD diagnoses
df$mini_ptsd_current #PTSD diagnoses
```
```

Which dataframe contains the variable you would like to call on? Mine is df

Which variable do you want to call? You must type verbatim the name that is in your dataframe.

What does the output look like?

```
Console  Terminal  Background Jobs
R 4.0.4 · ~/Documents/TRANDSX project/
> view(df) # lets have a look at what our imported dataset looks like
> df$mini_mdd_current
 [1]  1 NA NA  1  1 NA  1  1  1  1  1  1  1  1 NA  2  1  1  1  1  1  2  1  1  1  1  1  1  1  1  1  1  2 NA  1  1 NA  1  1  1  1  1
[45]  1  1  1  1 NA  1  1  2 NA  1 NA NA  1  1 NA  1  1  1  1  1 NA  1  1 NA NA  1 NA  1 NA  2  1 NA  1  1 NA  1  1  1 NA  1 NA  1  1  1
[89] NA NA  1 NA NA NA  1  1 NA NA NA
> df$mini_alcohol_current # AUD
 [1]  1 NA NA  1  1 NA  1  1  1  1  1  1  1  1 NA  2  2  1  1  1  1  2  1  1  1  1  1  2  1  1  1  1  1  1 NA  1  1 NA  2  2  1  1  1
[45]  1  1  1  1 NA  1  1  1 NA  1 NA  1  1  1  1  1  1  1  2 NA  2  1 NA  1  2 NA  1  1  2  1  1  1  1  1  1  2  2 NA  1  1  2  1  1
[89] NA NA  1 NA NA NA  1  1 NA NA NA
> df$mini_ptsd_current #PTSD diagnoses
 [1]  1 NA NA  1  1 NA  1  1  1  1  1  1  1  1 NA  2  1  1  1  1  2  1  1  1  2  1  1  1  1  1  1  1  1  1 NA  1  1 NA  1  1  1  1  1
[45]  1  1  1  1 NA  1  1  2 NA  1 NA  1  1  1  1  1  2  1  1  1 NA  1  1 NA  1  1 NA  1  1  2  1  1  1  1  1  2  1  1 NA  1  1  1  2  1
[89] NA NA  1 NA NA NA  1  1 NA NA NA
>
```

How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data
2. Induce the reader to think about the data being presented
3. Avoid distorting the data
4. Present many numbers with minimum ink
5. Make large datasets coherent
6. Encourage the reader to compare different bits of data

5 people have MDD
13 people have AUD
8 people have PTSD

```
### 1. Show the data

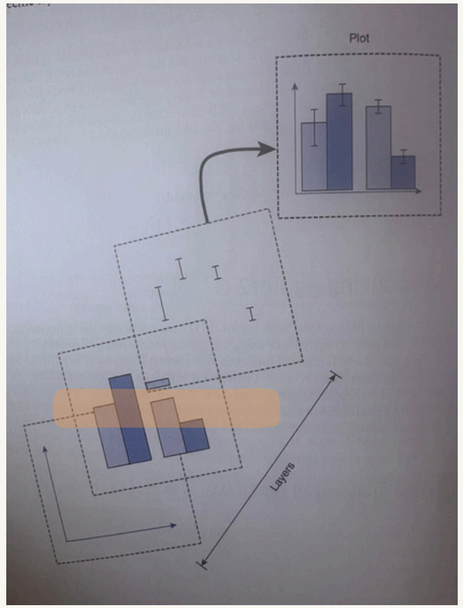
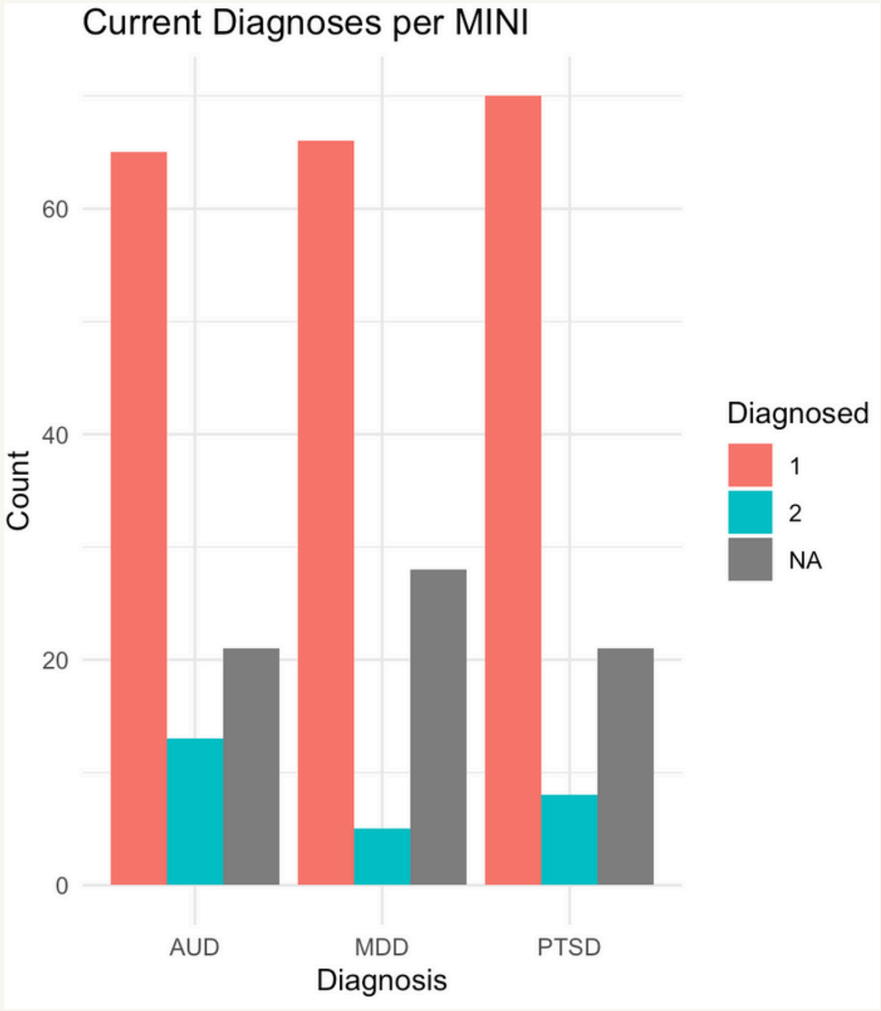
Check what they contain by calling on each variable individually (option 1)
This gives you the frequency of each diagnosis. For this dataset, 1 = NO diagnosis, 2 = YES diagnosis
```{r}
table(df$mini_mdd_current)
table(df$mini_alcohol_current)
table(df$mini_ptsd_current)
```

OR Calling on variables altogether as a vector (option 2)
```{r}
sapply(df[c("mini_mdd_current", "mini_alcohol_current", "mini_ptsd_current")], table)
```
```

```
Console  Terminal  Background Jobs
R 4.0.4 · ~/Documents/TRANDSX project/
> table(df$mini_mdd_current)

 1  2
66  5
> table(df$mini_alcohol_current)

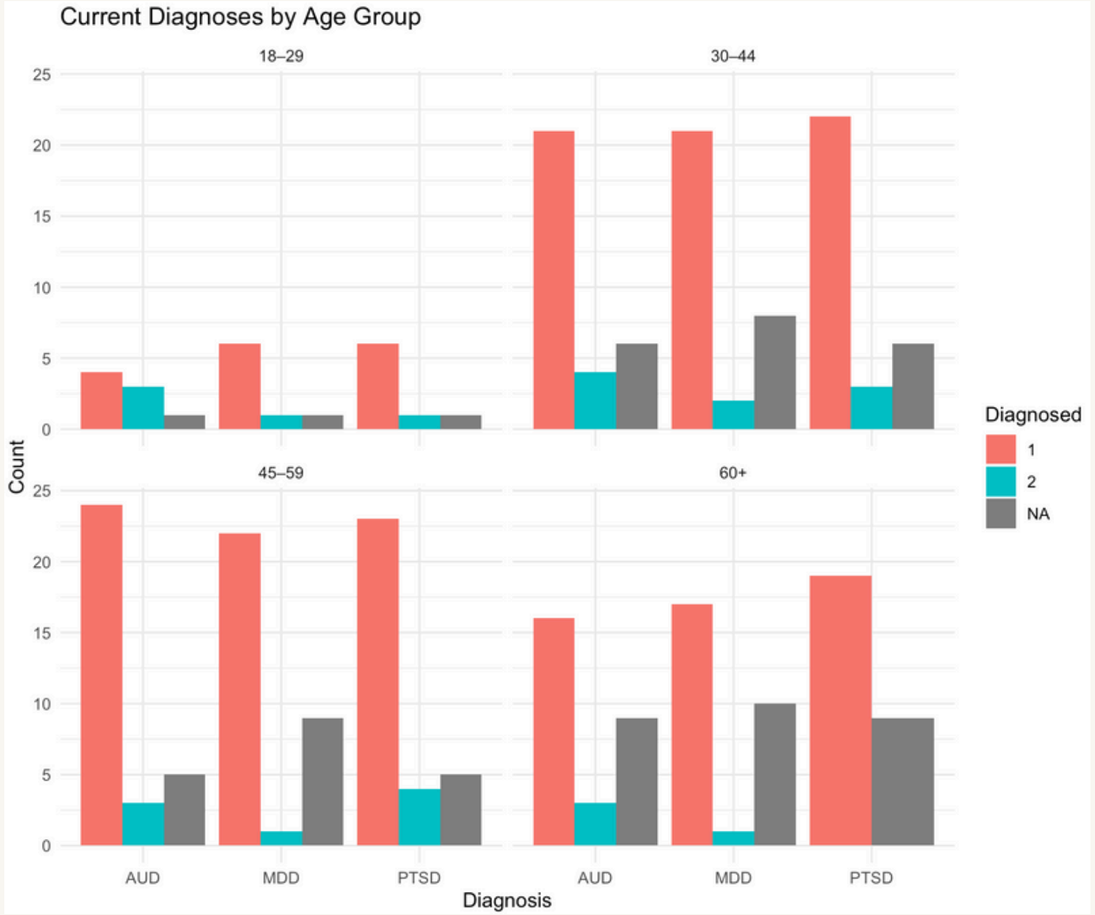
 1  2
65 13
> table(df$mini_ptsd_current)

 1  2
70  8
> sapply(df[c("mini_mdd_current", "mini_alcohol_current", "mini_ptsd_current")], table)
  mini_mdd_current mini_alcohol_current mini_ptsd_current
1               66                  65                70
2                5                  13                 8
>
```

Variables called on in vector = c("variable", "variable)

Apply the function to all variables in the vector

Same function from before, now just calling it all at once

How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data
2. Induce the reader to think about the data being presented
3. Avoid distorting the data
4. Present many numbers with minimum ink
5. Make large datasets coherent
6. Encourage the reader to compare different bits of data

Link to code

How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data
2. Induce the reader to think about the data being presented
3. **Avoid distorting the data**
4. Present many numbers with minimum ink
5. Make large datasets coherent
6. Encourage the reader to compare different bits of data

Anything wrong with this?



How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data
2. Induce the reader to think about the data being presented
3. **Avoid distorting the data**
4. Present many numbers with minimum ink
5. Make large datasets coherent
6. Encourage the reader to compare different bits of data

Anything wrong with this?

Y-axis inflation	- By setting the y-limit to 10,00 (likely way above actual values), you minimise visual differences, making everything look tiny.
Perception distortion	- Viewers may falsely assume diagnoses are rare or not worth attention.
Undermines comparison	- Tiny differences become invisible, reducing insight.



How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data
2. Induce the reader to think about the data being presented
3. Avoid distorting the data
4. Present many numbers with minimum ink
5. Make large datasets coherent
6. Encourage the reader to compare different bits of data

What does this look like in R?

```
2. Induce the reader to think about the frequency of diagnoses across each category using ggplot to create a bar chart.

```{r}
df %>% # calling on our df
  select(MDD = mini_mdd_current, # elect the variables to plot
         AUD = mini_alcohol_current,
         PTSD = mini_ptsd_current) %>%
  pivot_longer(everything(), names_to = "Diagnosis", values_to = "Status") %>% # changing our data to long format as is bettter for plotting
  group_by(Diagnosis, Status) %>% # specify which groups to plot and how
  summarise(Count = n(), .groups = 'drop') %>%
  ggplot(aes(x = Diagnosis, y = Count, fill = factor(Status))) + # layer to label things nicely and tell the graph where to overlay the data
  geom_bar(stat = "identity", position = "dodge") +
  labs(fill = "Diagnosed", title = "Current Diagnoses per MINI") + # add a title layer
  theme_minimal()

```
```

Layer to tell ggplot what to fill the axes with

Layer for the axis labels and titles

geom_bar - tells ggplot to add a bar plot layer



How should we present our diagnostic data?

There are some things to consider (Tufte, 2001):

1. Show the data ✓
2. Induce the reader to think about the data being presented ✓
3. Avoid distorting the data ✓
4. Present many numbers with minimum ink ✓
5. Make large datasets coherent ✓
6. Encourage the reader to compare different bits of data ✓

What does this look like in R?
Pretty!



Bonus: stratify by other variables

There are some things to consider (Tufte, 2001):

1. Show the data ✓
2. Induce the reader to think about the data being presented ✓
3. Avoid distorting the data ✓
4. Present many numbers with minimum ink ✓
5. Make large datasets coherent ✓
6. Encourage the reader to compare different bits of data ✓

```
### BONUS: stratify the diagnoses by age bracket and make look pretty :)

```{r}
# first need to specify the age groups
df$age_group <- cut(df$age, breaks = c(18, 30, 45, 60, 100),
                    labels = c("18-29", "30-44", "45-59", "60+"),
                    right = FALSE)
df %>%
  select(age_group,
         MDD = mini_mdd_current,
         AUD = mini_alcohol_current,
         PTSD = mini_ptsd_current) %>%
  pivot_longer(cols = c(MDD, AUD, PTSD), names_to = "Diagnosis", values_to = "Status") %>%
  group_by(age_group, Diagnosis, Status) %>%
  summarise(Count = n(), .groups = "drop") %>%
  ggplot(aes(x = Diagnosis, y = Count, fill = factor(Status))) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~age_group) +
  labs(title = "Current Diagnoses by Age Group", fill = "Diagnosed") +
  theme_minimal()
```
```

Pretty!



Problem 1: Missing Data

Frequency

we have plotted already (NAs)

Solution

different options - for diagnostic data, we will be transparent about missingness and filter missing data for analysis

Problem 1: Missing Data

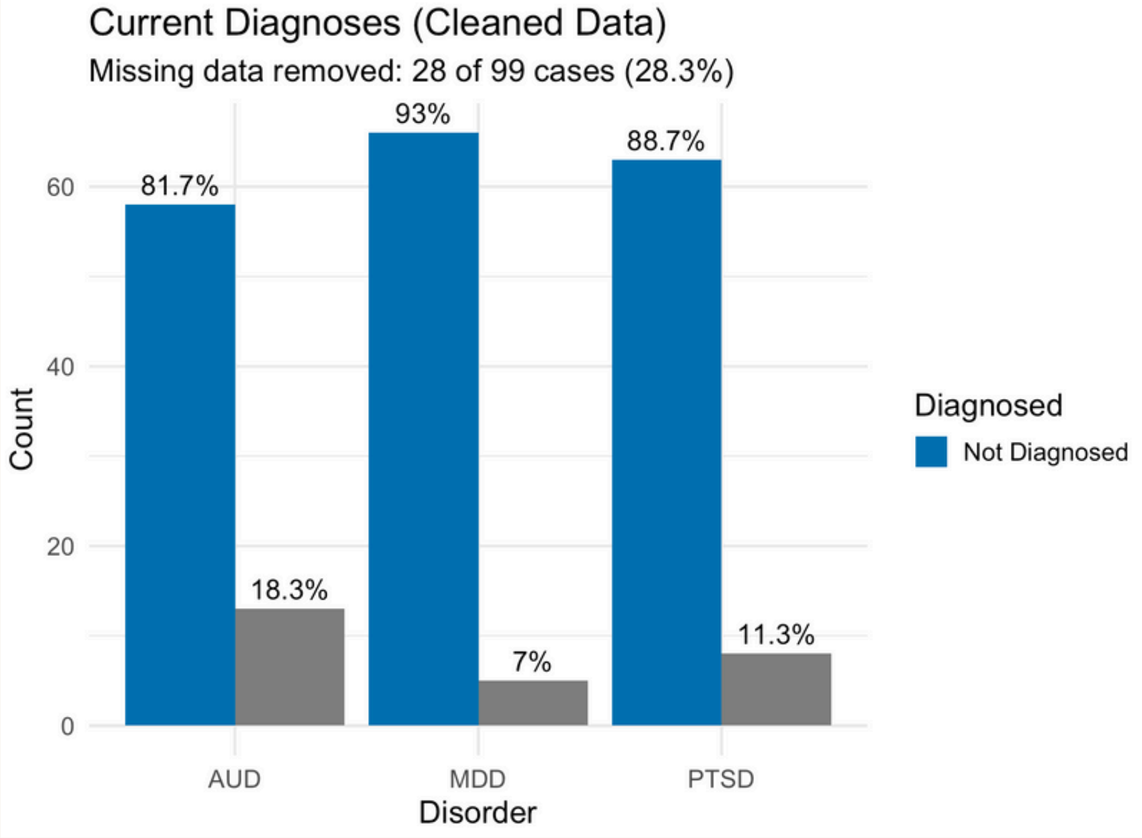
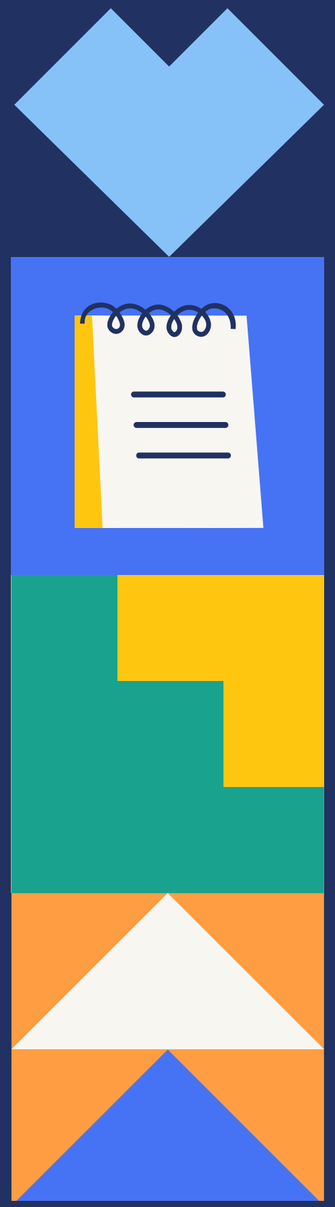
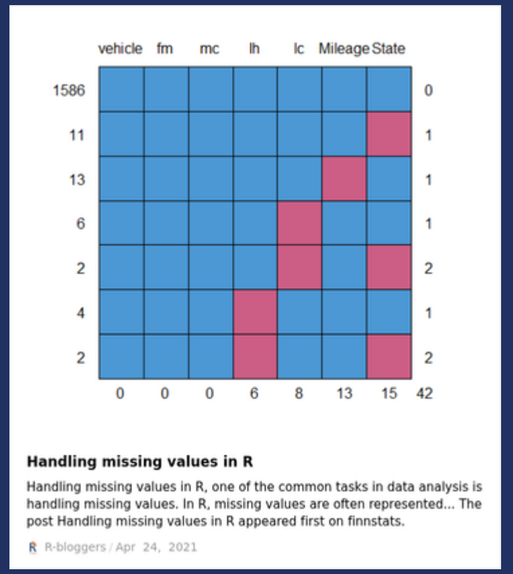
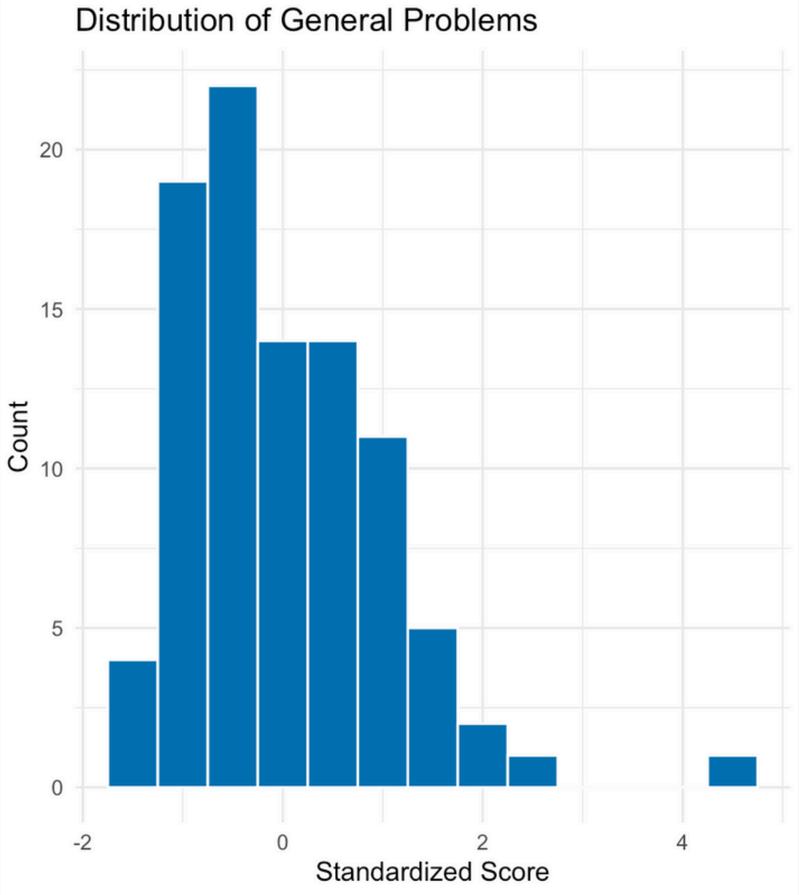
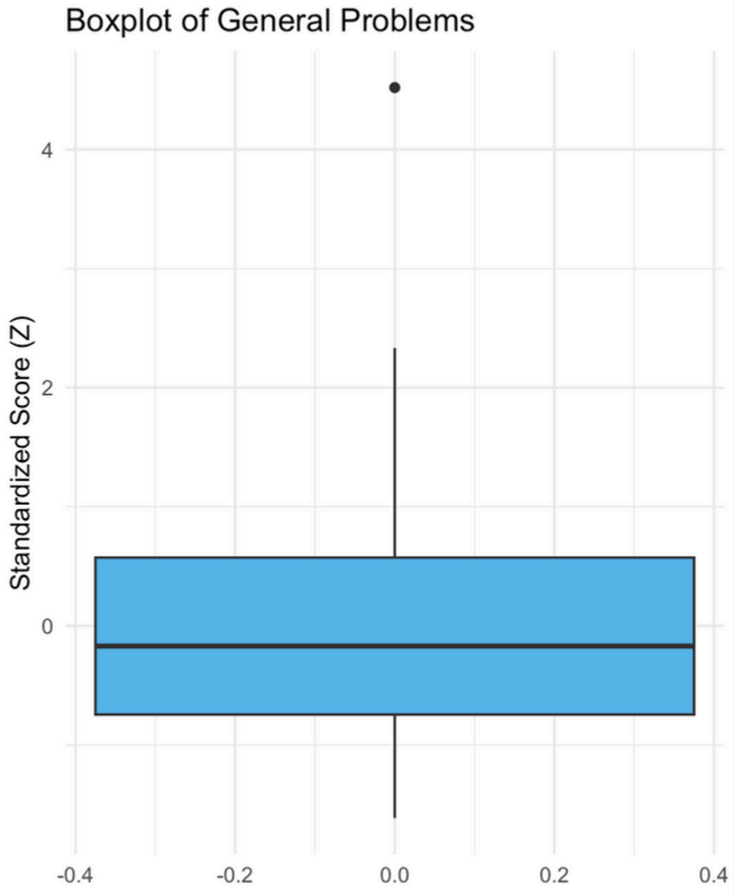
filter the data
! = NOT in R
is.na = anything with an NA value

Filter the data to only include individuals without NAs across any of our diagnostic variables

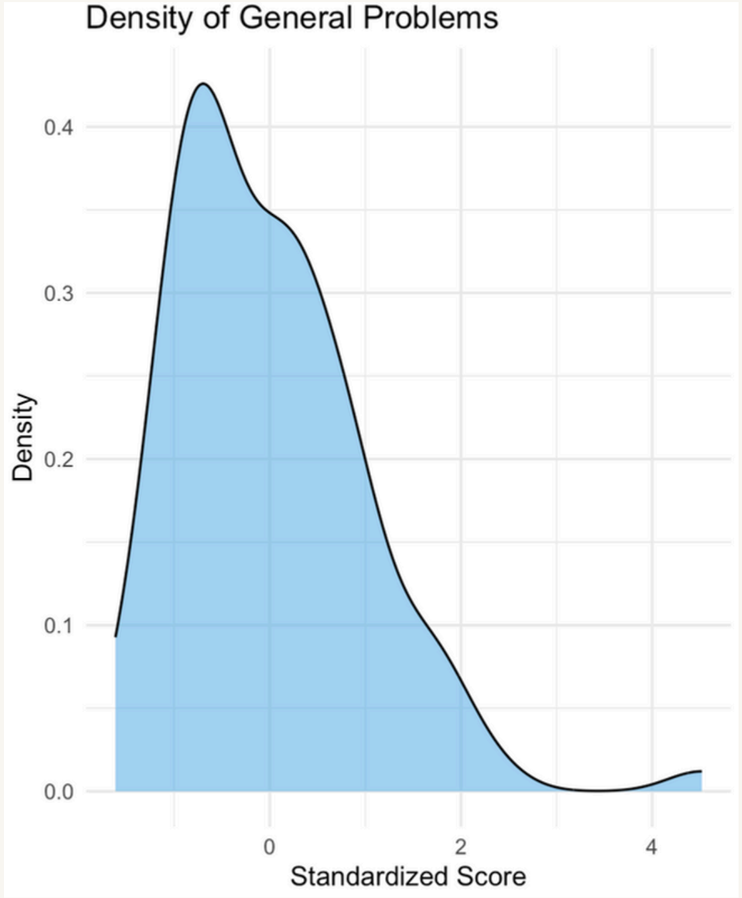
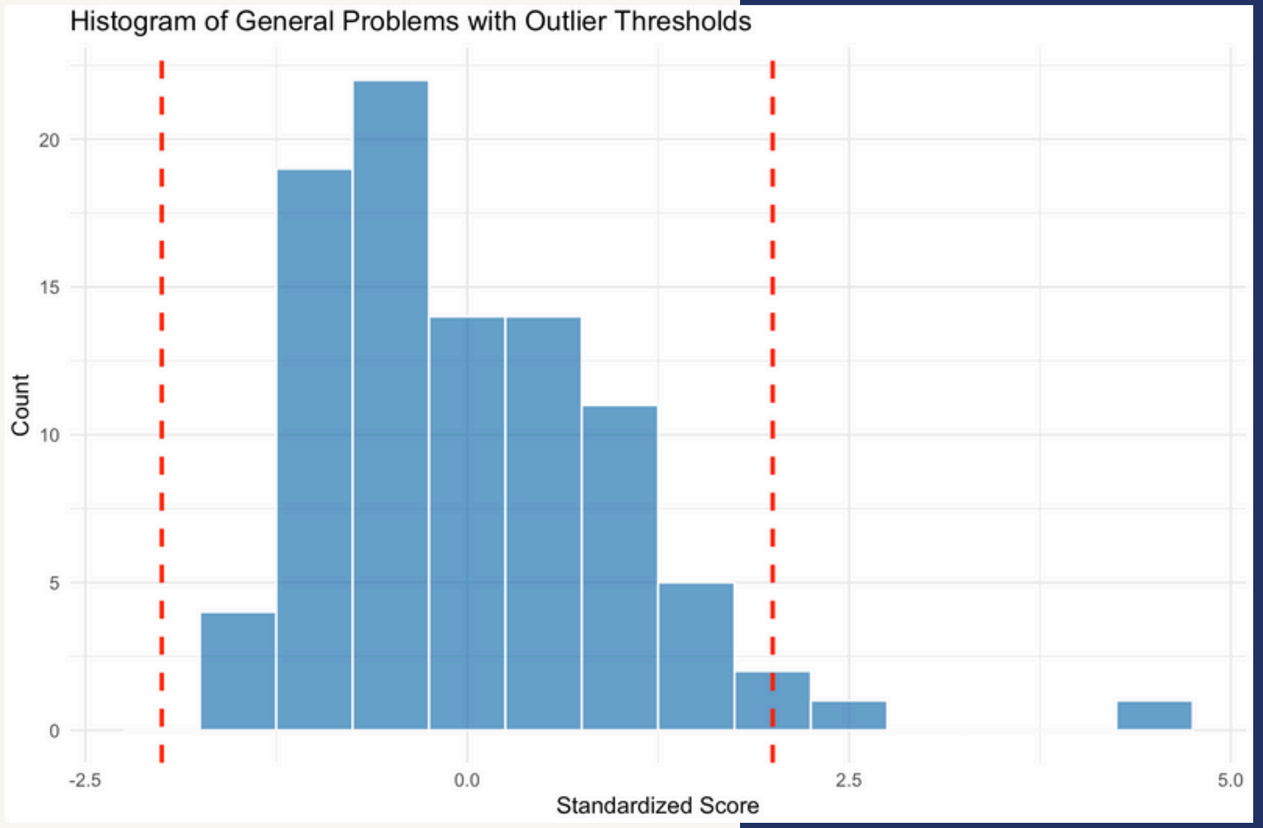
```r
## Problem 1: Handling Missing Diagnoses in the data
WHile there are many methods to handle missing data, for today's purpose we will be transparent about the freqwuency of missingness (achieved above).

### Solution: We will then filter the data so it is clean prior to analysis.

```{r}
df_clean <- df %>%
  filter(!is.na(mini_mdd_current) & # filter the dataset to only include values WITHOUT NAs
           !is.na(mini_alcohol_current) &
           !is.na(mini_ptsd_current))

# Total rows before cleaning
total_rows <- nrow(df)

# Total rows after removing missing
clean_rows <- nrow(df_clean)

# Rows with any missing in key diagnosis variables
missing_rows <- total_rows - clean_rows
```
```

Define a variable for the missingness. Simple maths formula: Total rows - clean rows (which you calculate in the two steps above!)

Solution

different options - for diagnostic data, we will be transparent about missingness and filter missing data for analysis

Problem 1: Missing Data

Solution

Plot data without NAs and add footnote about missingness

```
## Plot clean dataset

```{r}
ggplot(plot_data, aes(x = Diagnosis, y = Count, fill = factor(Status))) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = Label),
            position = position_dodge(width = 0.9),
            vjust = -0.5, size = 5) +
  scale_fill_manual(values = c("0" = "#999999", "1" = "#0072B2"),
                    labels = c("Not Diagnosed", "Diagnosed")) +
  labs(
    title = "Current Diagnoses (Cleaned Data)",
    subtitle = paste("Missing data removed: ", missing_rows, " of ", total_rows, " cases (",
                     round(100 * missing_rows / total_rows, 1), "%)", sep = ""),
    x = "Disorder",
    y = "Count",
    fill = "Diagnosed"
  ) +
  theme_minimal(base_size = 16)

```
```

Add a subtitle to explain missingness

Presenting Frequencies

Graph of our diagnostic data with our clean dataset while being transparent about missingness



Solution

Plot data without NAs and add footnote about missingness



Outline

R Crash Course

Link to code

Diagnosing Data Problems

- Visualising
- Descriptives
- Missing Data
- Outliers

Curing Data Problems

- Handling missing data
- Handling Outliers
- Visualising relationships

Diagnosing Problems: Continuous Data

Psychologists provided individuals with TBI a survey and then standardised their total score (we are neuropsychs so we love z scores).

We will use this data to explore outliers now. This will be done visually.

Associations between transdiagnostic psychopathology dimensions and cognitive functioning after traumatic brain injury: An application of the HiTOP-TBI model

Alexia Samiotis*[1,2], Jai Carmichael[1,2], Jao-Yue Carmintati[1,2], Amelia J Hicks[1], Jennie Ponsford[1,2], Kate Rachel Gould[1,2], Gershon Spitz[1,2,3]

Continuous raw data

ID	z-score on psych scale
1	0.50
2	1.53
3	3.63
4	0.10



This is not the method I use to handle missing data, see other methods

Follow the steps outlined previously to import and clean the data by filtering NA values

Visualisation for Outlier Identification





Values beyond ±2 or ±3 SD are often flagged as potential outliers

```
### Histogram – Distribution Overview

```{r}
ggplot(df, aes(x = General_Problems)) +
  geom_histogram(binwidth = 0.5, fill = "#0072B2", color = "white") +
  labs(title = "Distribution of General Problems",
       x = "Standardized Score",
       y = "Count") +
  theme_minimal()
```
```

```
### Boxplot – Classic Outlier Detection

```{r}
ggplot(df, aes(y = General_Problems)) +
  geom_boxplot(fill = "#56B4E9") +
  labs(title = "Boxplot of General Problems",
       y = "Standardized Score (Z)") +
  theme_minimal()
```
```

Visualisation for Outlier Identification

Values beyond ±2 or ±3 SD are often flagged as potential outliers






```
### Density Plot – Smoothed Distribution

```{r}
ggplot(df, aes(x = General_Problems)) +
  geom_density(fill = "#56B4E9", alpha = 0.6) +
  labs(title = "Density of General Problems",
       x = "Standardized Score",
       y = "Density") +
  theme_minimal()

```
```

```
### Histogram or Density Plot with Threshold Lines
Shows where outliers fall relative to the main distribution.

```{r}
ggplot(df, aes(x = General_Problems)) +
  geom_histogram(binwidth = 0.5, fill = "#0072B2", color = "white", alpha = 0.7) +
  geom_vline(xintercept = c(-2, 2), linetype = "dashed", color = "red", size = 1) +
  labs(title = "Histogram of General Problems with Outlier Thresholds",
       x = "Standardized Score",
       y = "Count") +
  theme_minimal()

```
```

Two potential outliers were identified

Problem 2: Outliers

There are many ways to handle outliers once identified.

Outliers

Can be anything > 2SD from the mean

As identified on boxplots or other visuals

Solution

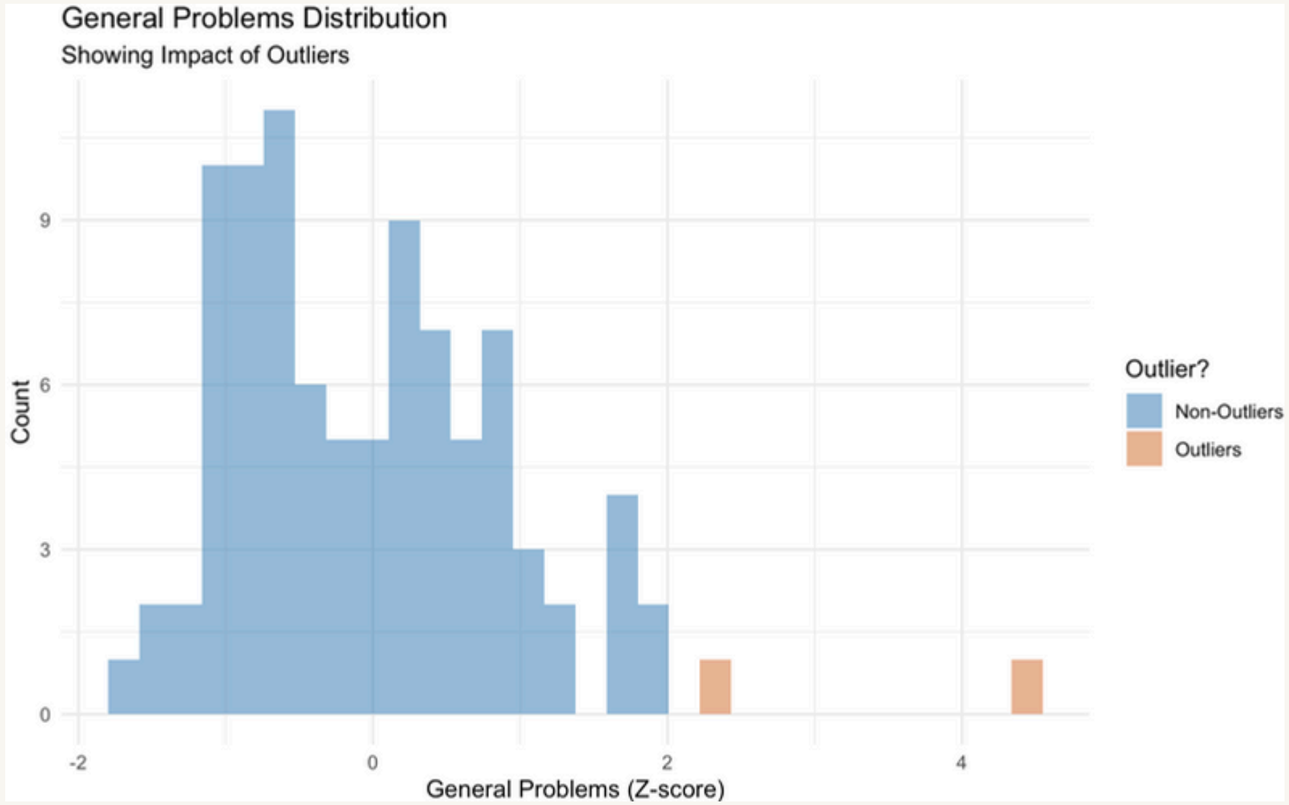
Leave them

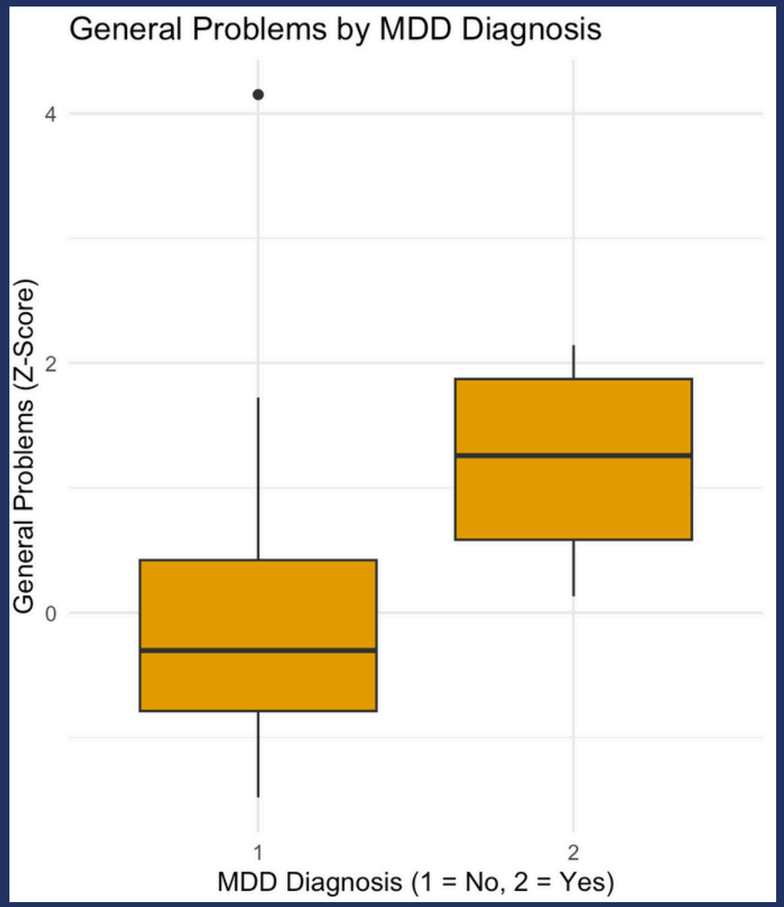
Remove

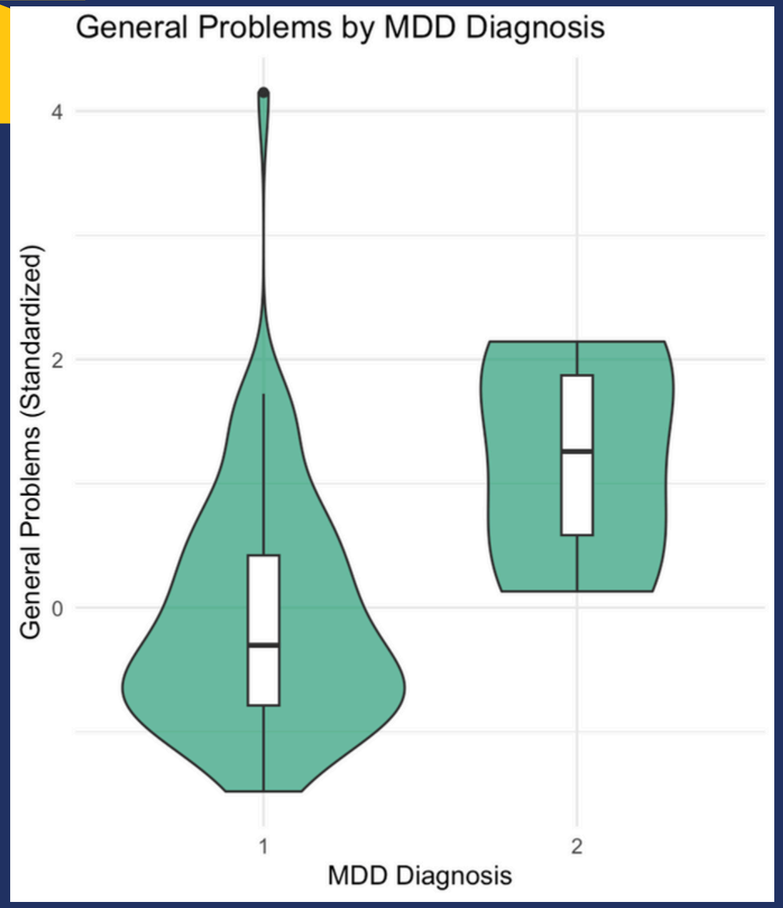
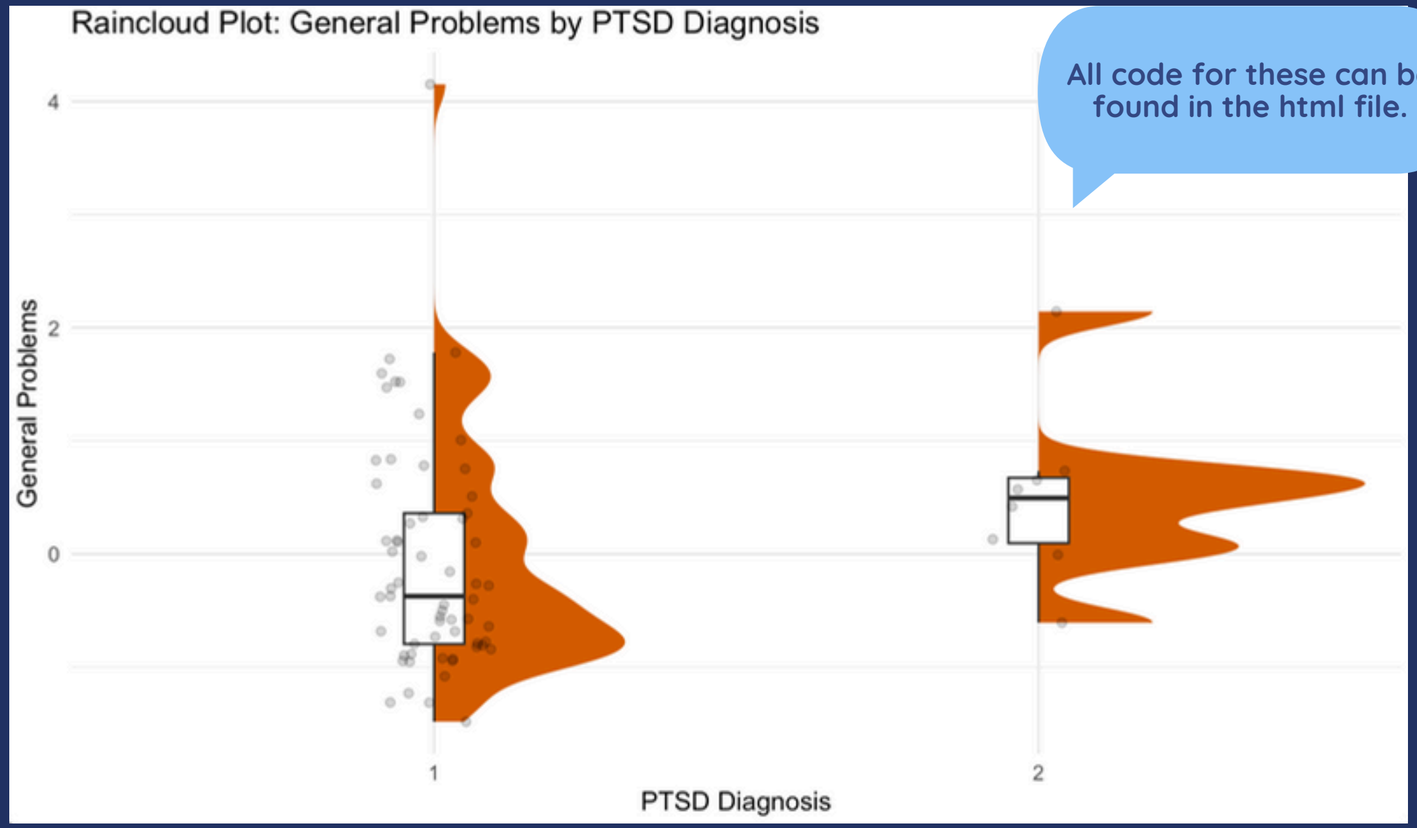
Winsorise (not covering)

Removing Outliers

We can compare the data with and without outliers before deciding what to do.




```r
### Option 3: Run analyses with and without outliers and see if it affects it.
```{r}
df <- df %>%
  mutate(outlier_flag = abs(General_Problems) > 2)


df %>%
  group_by(outlier_flag) %>%
  summarise(
    n = n(),
    mean_gp = mean(General_Problems, na.rm = TRUE),
    sd_gp = sd(General_Problems, na.rm = TRUE)
  )


ggplot(df, aes(x = General_Problems, fill = outlier_flag)) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  scale_fill_manual(values = c("FALSE" = "#0072B2", "TRUE" = "#D55E00"),
                    labels = c("Non-Outliers", "Outliers")) +
  labs(title = "General Problems Distribution",
       subtitle = "Showing Impact of Outliers",
       fill = "Outlier?",
       x = "General Problems (Z-score)",
       y = "Count") +
  theme_minimal()
```
```

Bonus: visualising relationships

We can do all the same techniques discussed while considering other imporant variables.

This is consistent with some of the findings from my PhD - dimensional/continuuous scales better capture psych problems than DSM diagnoses.

Interesting to present data this way as it shows how diagnostic labels don't neatly fit around all individuals with severe psychopathology after TBI.



There is one outlier with an extreme (high psychopathology severity) score in the no diagnosis group

```
## Synergystic Visualisatio: Using Two or More Variables
###  Boxplots by Group – Compare Across Diagnoses or Gender
Example: By MDD diagnosis

```{r}
df_clean$General_Problems <- scale(df_clean$a1_composite_score)
ggplot(df_clean, aes(x = factor(mini_mdd_current), y = General_Problems)) +
  geom_boxplot(fill = "#E69F00") +
  labs(title = "General Problems by MDD Diagnosis",
       x = "MDD Diagnosis (1 = No, 2 = Yes)",
       y = "General Problems (Z-Score)") +
  theme_minimal()
```
```

Bonus: visualising relationships

We can do all the same techniques discussed while considering other imporant variables.





All code for these can be found in the html file.

```
### Violin Plot – Combine Density + Boxplot

```{r}
ggplot(df_clean, aes(x = factor(mini_mdd_current), y = General_Problems)) +
  geom_violin(fill = "#009E73", alpha = 0.7) +
  geom_boxplot(width = 0.1, fill = "white") +
  labs(title = "General Problems by MDD Diagnosis",
       x = "MDD Diagnosis",
       y = "General Problems (Standardized)") +
  theme_minimal()
```
```

```
### Raincloud Plot (Optional, Modern & Informative)

```{r}
library(ggdist)
ggplot(df_clean, aes(x = factor(mini_ptsd_current), y = General_Problems)) +
  stat_halfeye(adjust = 0.5, width = 0.6, .width = 0, fill = "#D55E00") +
  geom_boxplot(width = 0.1, outlier.shape = NA) +
  geom_jitter(width = 0.1, alpha = 0.2) +
  labs(title = "Raincloud Plot: General Problems by PTSD Diagnosis",
       x = "PTSD Diagnosis",
       y = "General Problems") +
  theme_minimal()
```
```

Thank you!

Associations between transdiagnostic psychopathology dimensions and cognitive functioning after traumatic brain injury: An application of the HiTOP-TBI model

Alexia Samiotis*[1,2], Jai Carmichael[1,2], Jao-Yue Carmintati[1,2], Amelia J Hicks[1], Jennie Ponsford[1,2], Kate Rachel Gould[1,2], Gershon Spitz[1,2,3]

Link to code

Supervisors

- Prof Jennie Ponsford
- Dr Gershon Spitz
- Dr Jai Carmichael

Questions or need any code?
alexia.samiotis1@monash.edu
or find me at the desks at MERRC Thursdays/Fridays 🙂