Network Analysis in R

2019-12-14

Contents

Overview

Materials for the 4 day Network Analysis course.

This course covers skills such as installing R, opening fies, working with data using tidyverse, and making graphs. It also introduces network analysis as a statistical concept.

Chapter 1

Starting with R

Welcome to the Course!

1.1 Overview

Installing R and opening files

In this session you will learn:

1. What is R?
2. How to install R
3. Where to get help

1.1.1 What is R?

For network analysis, you need two different bits of software, R and RStudio. R is a programming language that you will write code in and R Studio is an Integrated Development Environment (IDE) which makes working with R easier.

1.1.2 How To Install Base R

Install base R from https://cran.rstudio.com/. Choose the download link for your operating system (Linux, Mac OS X, or Windows).

1.1.3 How To Install R Studio

Go to https://rstudio.com and download the RStudio Desktop (Open Source License) version for your operating system under the list titled **Installers for Supported Platforms.**

If you are using a Mac, please install XQuartz

1.1.4 Quiz

Quickfire Questions

We have put questions throughout to help you test your knowledge. When you type in or choose the correct answer, the dashed box will change color and become solid green.

- From the following options, how do you get R for this course? Installing Base R & R Studio Installing R Studio Installing Base R

Explain This Answer!

R is the basic package. R Studio is an add-on that make R much easier to use.

1.2 Where to Get Help

Learning R can take time. Please do not feel that you have to grasp everything right away and it is totally ok to copy and paste.

1.2.1 Google

"Never commit to memory what can be easily looked up in books"

Often people simply google what they want to achieve in R. For example, in preparing this tutorial I googled "how to run a linear mixed effects model in R" (as I did not know) and found the following helpful tutorial which goes through all the steps. This is just one of many examples of resources that are available.

1.2.2 Stackoverflow

Stackoverflow is a website where you can ask for help on bits of coding that you are struggling with. You can search for commonly asked questions and responses.

Feel free to google any time.

Chapter 2

Working With Data in R

2.1 Overview

Working With Data

In this session you will learn:

1. How to open files
2. How to write code
3. How to install packages

This is a basic introduction to R. The material is based on the data skills course for MSc students at the University of Glasgow. Find lots of useful resources here: https://gupsych.github.io/data_skills/01_intro.html

Please take a look at these free open resources in your own time.

2.1.1 Setting Working Directory

First things first, we will set the working directory. What this means is that we need to tell R where the files we need are located. Think of it just like when you have different projects, and you have seperate folders for each project e.g. research conducted in schools, research conducted in the community and so on. When working on R, it's useful to have all the data sets and files you need in one folder.

To set the working directory press session -> set working directory -> choose directory and then select the folder where the data sets we are working on are saved, and save this file in the same folder as well. In other words- make sure your data sets and scripts are all in the same folder.

2.1.2 Code

RStudio generally has four panels: Current file, Console, Environment, and Viewer. You can think of the console as a place to try things out, and the file to write down ideas you want to stick around. Go to the console and type

```
x <- 1 + 5
x
```

Notice how now the environment shows we have a Value x that is 6. We have just created a variable. In the above, we would say "the variable x is assigned to 1 + 5" or "x gets 1 + 5"

2.1.3 Functions & Arguments

We have already created some code. But what does it all mean?

Functions in R execute specific tasks and normally take a number of arguments (if you're into linguistics you might want to think as these as verbs that require a subject and an object). You can look up all the arguments that a function takes by using the help documentation by using the format `?function`. Some arguments are required, and some are optional. Optional arguments will often use a default (normally specified in the help documentation) if you do not enter any value.

As an example, let's look at the help documentation for the function rnorm() which randomly generates a set of numbers with a normal distribution. Just like the numbers in the graph below.

Open up R Studio and in the console, type the following code:

```
?rnorm
```

The help documentation for `rnorm()` should appear in the bottom right help panel. In the usage section, we see that `rnorm()` takes the following form:

```
rnorm(n, mean = 0, sd = 1)
```

In the arguments section, there are explanations for each of the arguments. `n` is the number of observations we want to create, `mean` is the mean of the data points we will create and `sd` is the standard deviation of the set. In the details section it notes that if no values are entered for mean and sd it will use a default of 0 and 1 for these values. Because there is no default value for n it must be specified otherwise the code won't run.

Now, try running the above code for 50 participants with a mean test score of 3 and a standard deviation of 1.

Remember we are asking R to create **random** numbers here, so do not worry if someone else has slightly different values.

I need a hint!

`n` in this case would be changed to 50.

`mean` should be changed to 3

```
rnorm(50, mean = 3, sd = 1)
```

now, try running the above code and see what happens.

2.2 Scripts

So far we have been typing into the console but that means all work is lost when re-start R. We can save our analysis code in scripts. To create a new script select the option from the File menu.

2.2.1 Tidyverse

However, we do not always want to use R to create random numbers, we want to use it to analyse our own data which commonly resides in .csv or .sav files. People have developed many different libraries to help us work with such data. One of the most popular packages is tidyverse.

The Tidyverse is a collection of R packages with a common design , grammar, and data structure that makes analysis faster and easier.

The first time you want to use a package, you must first install the package. `tidyverse` can be installed as follows.

```
install.packages("tidyverse")
```

Once the package has been installed, any time you want to use the package you use the following code. If you want to open a package other than `tidyverse`, simply substitute the package name.

One of the key underlying structures of the tidyverse is that data structures follow a tidy format:

- Each variable is in a column
- Each observation is a row
- Each value is a cell

2.2.2 Opening SPSS Files

The `foreign` package allows easy opening of .sav files which are associated with SPSS.

You can either set a path to open up an SPSS file or run `file.choose()` and pick the file manually.

To use foreign, we need to first install the package, same as we did with tidyverse. Follow the instructions for installing tidyverse and subsitute the package name.

Click to see solution...

```
install.packages("foreign")
```

Once you have installed packages, .sav files can be opened using (editing in your actual file name) with the following code:

```
dataset = read.spss("YOURFILENAMEHERE.sav", to.data.frame=TRUE)
```

While we are not using spss files during this training, it is useful to know how to do this.

2.3 iris Task

R comes with a few built in data sets. One of which is called `iris` which is about the plant . Now that we have installed `tidyverse` let us see what it can do.

Create a new script and call the script iris. We are going to look at the top ten values of the in built data set. Once you are done, feel free to try looking at the top 3,7 and 31 values.

```
library("tidyverse")
head(iris, 10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

What do you think `tail(iris, 10)` will do?

2.4 Calculate a Mean

From looking at the dataset, we can see it has information on the sepal and petal lengths and widths for different species of iris. We are interested in calculating **mean** petal width and we will use the `summarise` function to do it.

`summarise` is a useful function which works for means, medians and interquartile ranges and gives results in a single row.

You can run `?summarise` to bring up the help function for more information.

Now, try running the following codes within your iris script.

the %>% is the pipe operator, it simply means **"and then"**. So, we are bringing up the iris database **"and then"** we are summarising.

```
iris %>% summarise(mean_petal_length = mean(Petal.Length))
```

```
mean_iris <- iris %>% summarise(mean_petal_length = mean(Petal.Length))
```

- From the following options, what is the difference in the code? one is assigned to a variable one actually calculates the median both are the same

Explain This Answer!

`mean_iris <- iris %>% summarise(mean_petal_length = mean(Petal.Length))` calculates the mean and creates a new dataframe (which can be seen in the environment) while `iris %>% summarise(mean_petal_length = mean(Petal.Length))` simply calculates the mean.,

2.5 Calculate a Median

A lot of data in psychology can be skewed, so sometimes the median is better (as numbers are equally likely to fall above or below it). How would you adapt the code for calculating the mean to calculate a median?

Calculate median petal width and save to your script.

I need a hint!

`mean(Petal.Width)` is the function within the code block that calculates the mean, try changing to median(Petal.Width). mean_petal_width is a column name we made up so it might be a good idea to rename it something meaningful.

2.6 Groups

What about if we wanted to calculate a mean for each different species of orchid? `Tidyverse` has a helpful option which lets us group by factors. This is helpful if we are looking at differences between females and males, for example

```
grouped <- iris %>% group_by(Species) %>% summarise(mean_petal_length = mean(Petal.Leng
grouped
```

```
## # A tibble: 3 x 2
##   Species    mean_petal_length
##   <fct>                  <dbl>
## 1 setosa                  1.46
## 2 versicolor              4.26
## 3 virginica               5.55
```

What has the smallest mean petal length?

versicolor sertosa virginca

2.6.1 Keeping Environment Clean

As you work, you will notice your environment will (quickly) fill up with lots of variables that you have assigned.

Before beginning any new analyses it is important to clear the environment which can be done with the following code. If you run this code R will forget any libraries that you have loaded, such as `tidyverse`.

```
rm(list = ls())
```

Chapter 3

Network Analysis

3.1 Overview

In this session you will learn:

1. Basics of Network Approach

Check the references section for more advanced topics!

3.2 Background

Recent thinking conceptulises mental wellbeing as comprising of environmental, psychological and social factors, a challenge to static factor models. Psychologists wishing to study the dynamic interaction of many factors may wish to consider a complexity science perspective such as network analysis.

The potential utility of network analysis was noted over ten years ago in intelligence research. The frequently reported patterns of positive correlations between various cognitive tasks (e.g. verbal comprehension and working memory) are typically explained in terms of a dominant latent factor, i.e. the correlations reflect a hypothesised common factor of general intelligence (g). However, van der Maas (2006) and colleagues argued that this empirical pattern can also be accounted for by means of a network approach (see image below), wherein the patterns of positive relationships can be explained using a mutualism model, i.e. the variables have mutual, reinforcing, relationships (Hevey, 2018)

The image above from van der Maas (2006) demonstrates the difference between a latent factor and a network approach, similiar to what we have seen in the slides.

3.3 Why Use Network Analysis in Mental Health?

"Analysis of mental health data is usually based on sum-scores of symptoms or the estimation of factor models. Both types of analyses disregard direct associations among symptoms that are well-understood in clinical practice: mental health problems can be conceptualized as vicious circles of problems that are hard to escape. A novel research framework, the network perspective on psychopathology, understands mental disorders as complex networks of interacting symptoms" (Fried, 2018)

However, mental health networks can contain more than just symptoms. In a study looking at patients with schizophrenia the researchers wanted to understand the extent to which variables belonging to the same construct were connected and how different constructs mutually interacted and reinforced one another. In addition to modelling symptoms, the researchers also looked at engagement with mental health services and interpersonal relationships. This is important when we want to explore what wider influences might impact upon behaviour we are interested in.

3.4 What is a network?

A network is a set of nodes connected by a set of edges. A network (as collection of various nodes and edges) is also sometimes known as a graph.

A node (in psychological research) can represent a single item from a scale, a subscale, or a composite scale: the choice of node depends upon the type of data that provide the most appropriate and useful understanding of the questions to be addressed. Edges can represent different types of relationships, e.g. comorbidity of psychological symptoms.

Several packages are used in the network analysis, including `network`, `statnet`, `igraph` and `qgraph`.

qgraph was developed in the context of psychometrics approach by Dr. Sacha Epskamp and colleagues in 2012. We will be be working with `qgraph` and `bootnet`. Please `install.packages("qgraph")` (which we will use now) and `install.packages("bootnet")` (which we will use later) so we can get started.

Now, let us load up `qgraph` - remember and save all of your code into a script.

```
library(qgraph)
```

3.5 Drawing Matrices

Tasks on section 3 are based on that of Derek De Beurs who uses network analysis to understand suicide ideation. Check out his work.

First, we will create a blank matrix with the following code. Here, we are asking R to create a matrix with 3 rows and 3 columns.

The code `colnames(Matrix) <- c("Node1","Node2","Node3")` is simply providing column names in the order they are created.

We can see that we have created a 3x3 matrix with no data.

```
Matrix <- matrix(0,nrow=3,ncol=3)
colnames(Matrix) <- c("Node1","Node2","Node3")
Matrix
```

```
##       Node1 Node2 Node3
## [1,]     0     0     0
## [2,]     0     0     0
## [3,]     0     0     0
```

Now, we would like to assign some data to the matrix. We are mapping edges between nodes. For example, in the first line of code below "3" is the weighted edge between node 1 and 2.

```
Matrix[1,2] <- 3
Matrix[2,1] <- 3
Matrix[2,3] <- 3
Matrix[3,2] <- 3
Matrix
```

```
##       Node1 Node2 Node3
## [1,]     0     3     0
## [2,]     3     0     3
## [3,]     0     3     0
```

```
qgraph(Matrix)
```

3.6 Understanding Network plots

Change the Matrix values and visualise the result using qgraph. Feel free to try negative numbers too.

Click to see one possible example…

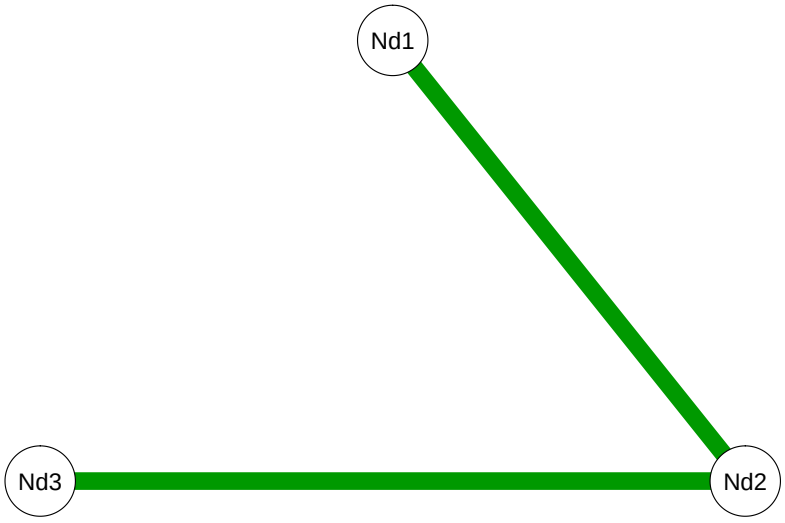

Figure 3.1: Matrix Visualisation

```
Matrix[1,2] <- 3
Matrix[2,1] <- 3
Matrix[2,3] <- -3
Matrix[3,2] <- 3

qgraph(Matrix)
```

3.7 Task

Please run the following code and look at *both* the visualisation of the nodes and the strength plot, what do you think this tells us about node 2?

```
Matrix[1,2] <- 3
Matrix[2,1] <- 3
Matrix[2,3] <- 3
Matrix[3,2] <- 3

data_frame <- qgraph(Matrix)
```

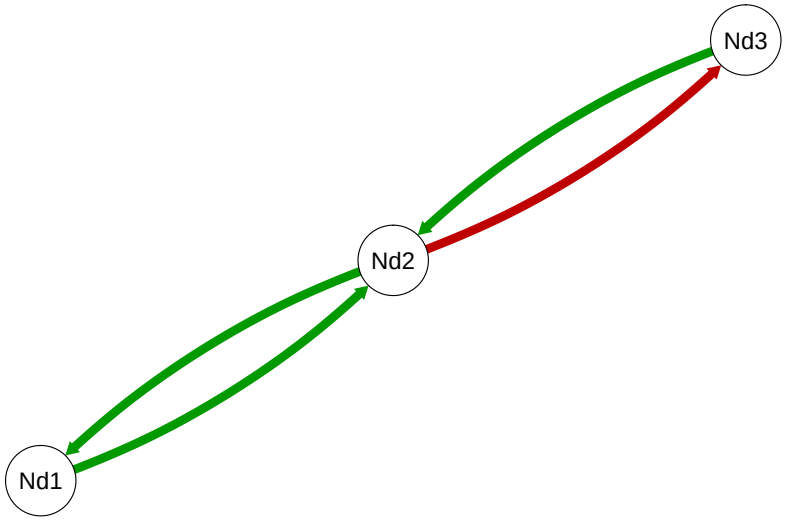

Figure 3.2: One Example - Change Numbers to see what happens

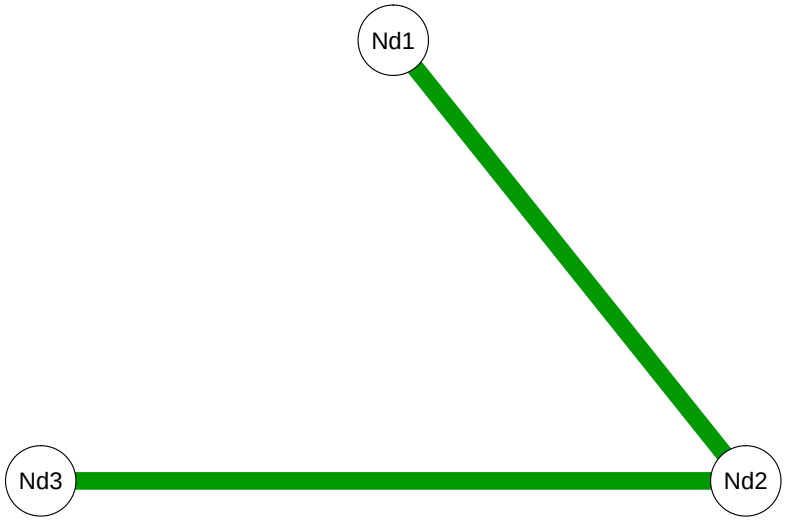

Figure 3.3: Centrality Plot

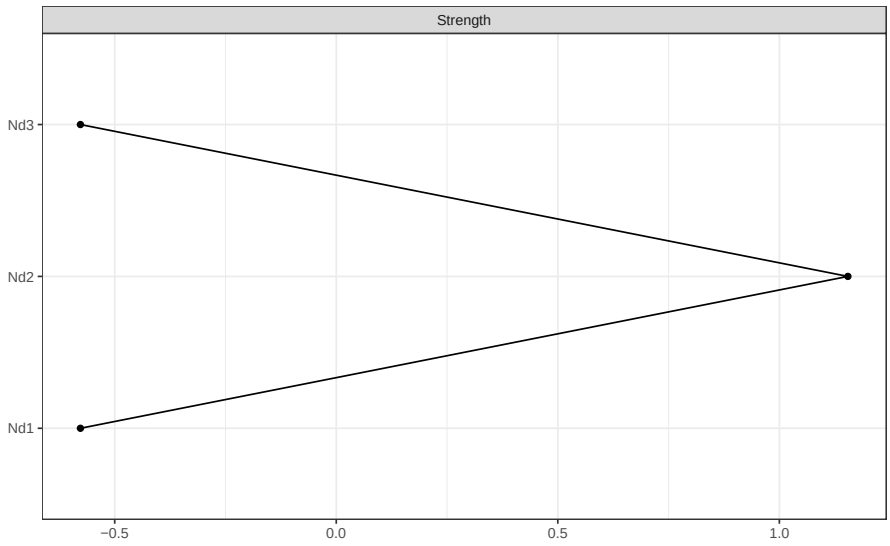

Figure 3.4: Centrality Plot

```
centralityPlot(data_frame)
```

```
## Note: z-scores are shown on x-axis rather than raw centrality indices.
```

Chapter 4

Practical

4.1 Overview

In this session you will learn:

1. How to work with original data
2. How to define factors
3. How to wrangle data
4. How to visualise data
5. How to run a simple network analysis.

4.2 Description of Data

We will work with a local dataset gathered from high school age children. Please download the .csv file HERE (you might need to right click and choose save as).

First, we need to make sure we have `tidyverse` loaded and then open the file. We are creating a data frame called which we will assign to our .csv file.

Remember, this needs to be saved within the working directory that you set so that R knows where to find it. We covered this before

We will be estimating networks in this session so we need to install `bootnet` - please install in the same way as we have for other packages.

I need a Hint…

```
install.packages("bootnet")
```

Once we have installed bootnet, we can load the packages we need and open up the file.

```
library(tidyverse)
library(qgraph)
library(bootnet)
data <- read_csv("networkdataset.csv")
```

R has the `str()` function which lets us look at the structure of data. Look at `str(data)`

As we can see, Gender, Place and Grade are listed as character data (words), this is not accurate and it would be better to let R know these data are factors.

```
data_factors <- data %>%
  mutate(Gender = as.factor(Gender),
         Place = as.factor(Place),
         Grade = as.factor(Grade),
         Familyarrangement = as.factor(Familyarrangement),
         Average = as.factor(Average),
         educationFather = as.factor(educationFather),
         educationMather = as.factor(educationMather))
```

Now, if we run `str(data_factors)` we can see that this dataframe is recognised as factors.

4.3 Visualising Data

R can make all sorts of graphs. We have already seen some and we can now make some ourselves.

4.4 Visualising Data - Practical

How many females and males?

The # symbol in the code is a comment. Comments do not add anything to the code but are there for human understanding to explain what is happening in the code. Look at the comments in the code below to understand what is going on.

```
# we are telling R we want to plot dat, we are letting R know Gender
# is a factor on our x axis and we want to fill with colour
ggplot(data_factors, aes(x = Gender, fill = Gender)) +
```

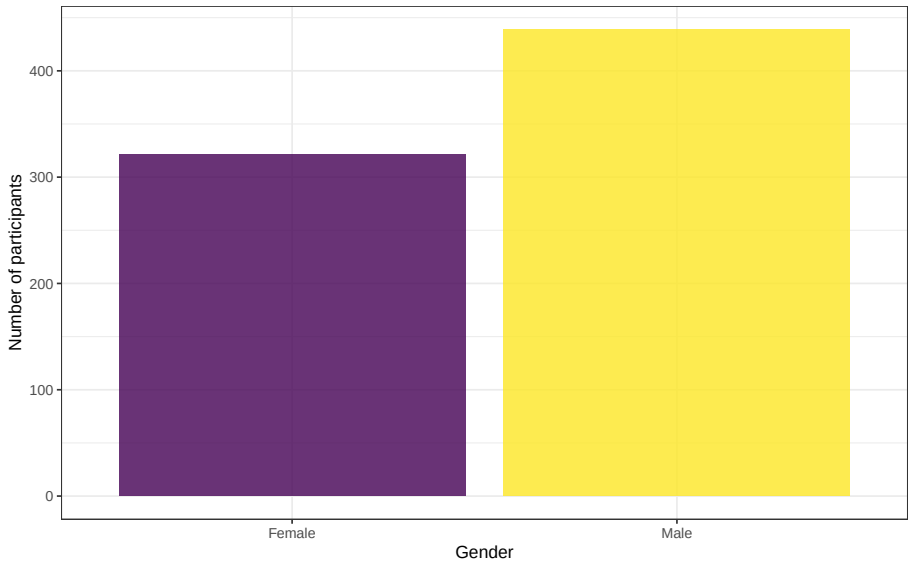

Figure 4.1: Participant Gender

```
  # we are choosing a bar plot
  geom_bar(show.legend = FALSE, alpha = .8) +
  # we are naming the x scale
  scale_x_discrete(name = "Gender") +
  # we are chosing a colour scheme from pre-exising options
  scale_fill_viridis_d(option = "D") +
  # we are naming the y scale
  scale_y_continuous(name = "Number of participants")
```

4.5 Working With Data

Participants rated themselves on 23 items of the TEQ questionnaire which asked about exposure to different traumatic events. We would like to create a plot to visualise the total TEQ and where participants live. However, we need to clean the data a little bit first. We will explain each line in the code below.

We will use the pipe %>% a lot to tell R to do something **"and then"** do something else.

1. we create a new dataframe called `total_TEQ` which we assign to our new code

2. we then use `select`, the . lets R know we are still working with the same dataframe. We then select all TEQ variables, Pcode and Place `select(., TEQ1:TEQ23, Pcode, Place`
3. `gather` tranforms the data from wide to long. We only want it to do this to TEQ variables so we can drop `-Pcode` and `-Place` using the minus sign.
4. We then use `group_by(Pcode, Place)` as we want to look at scores for each participant and where they live
5. Finally `summarise(tot_TEQ = sum(score))` creates a column named tot_TEQ which has the total TEQ score for each participant.

```
total_TEQ <- data_factors %>%
select(., TEQ1:TEQ23, Pcode, Place) %>%
gather("var", "score", -Pcode, -Place) %>%
group_by(Pcode, Place) %>%
summarise(tot_TEQ = sum(score))
```

4.6 Plotting Data

Now we can visualise `total_TEQ`.

When visualising differences between groups in numerical values, people have historically used bar graphs. There is currently a shift taking place, with people moving away from bar graphs, and towards more informative visualisations.

Before we used `geom_bar()` to create a barplot and this time we will use `geom_violin()` to create a violin plot. In the violin plot, the width of the area indicates the density of the data at that point on the y axis. We have also included a box plot using `geom_boxplot` within this graph to show another option.

```
ggplot(total_TEQ, aes(x = Place, fill = Place, y = tot_TEQ)) + geom_violin(show.legend
  geom_boxplot(width = 0.2, show.legend = FALSE) +
  scale_y_continuous(name = "Total TEQ") +
  scale_x_discrete(name = "Place")
```

From looking at the graph we just created, what location tends to have highest TEQ?

Village Camp City

4.7 Recoding Factors

The original file has Camp, city, Village. It might be neater to have Camp, City, Village. We can easily recode the city level of the Place factor using the following code.

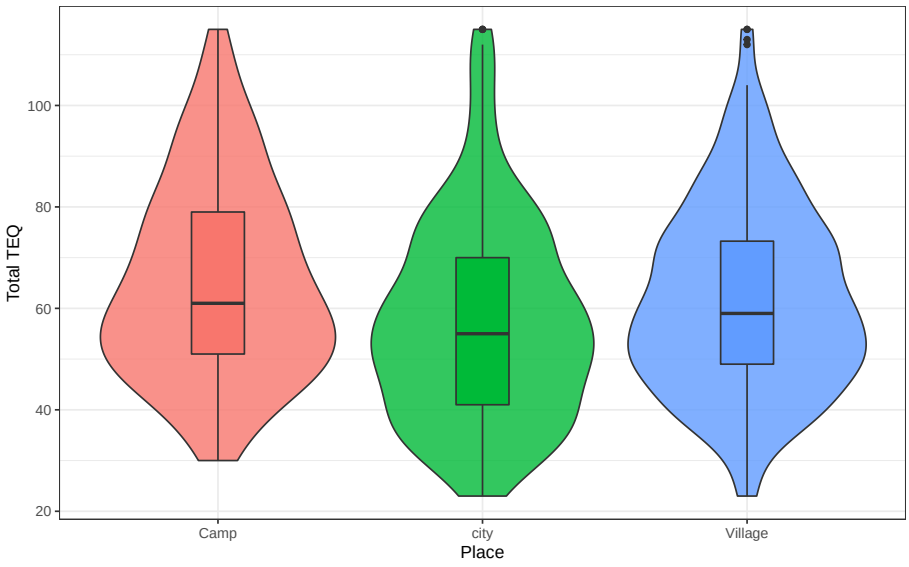

Figure 4.2: TEQ Scores

If you re-run the code for the graph now, this should be updated.

```
total_TEQ <- total_TEQ %>%
    mutate(Place = recode(Place, city = "City"))
```

4.8 Simple Network

When creating a network, it is important to think about how complex it will be. The higher the number of nodes being examined, then the higher the number of edges have to be estimated. Too many nodes results in networks that are unstable.

In this example, we will create a simple network from the single question items of the PAQ (Parental Authority Questionnaire).

First, we will select the PAQ items from data_factors:

```
PAQ <- data_factors %>%
  select(PAQ1:PAQ11)
```

Like all statistical techniques that use sample data to estimate parameters, the correlation and partial correlations values will be influenced by sampling variations and exact zeros will be rarely observed in the matrices. Consequently,

correlation networks will nearly always be fully connected networks, possibly with small weights on many of the edges that reflect weak and potentially spurious partial correlations.

Spurious relationships will be problematic in terms of the network interpretation and will compromise the potential for network replication. In order to limit the number of such spurious relationships, a statistical regularisation technique, which takes into account the model complexity, is frequently used.

The LASSO regularisation yields a more parsimonious network (fewer connections between nodes), assumed to reflect only the most "important" empirical relationships in the data. The LASSO utilizes a tuning parameter (lambda) that controls the level of sparsity. The tuning parameter directly controls how much the likelihood is penalized for the sum of absolute parameter values. When the tuning parameter is low, only a few edges are removed, likely resulting in the retention of spurious edges. When the tuning parameter is high, many edges are removed, likely resulting in the removal of true edges in addition to the removal of spurious edges. EBICglasso sets a tuning parameter of 0.5 by default.

```
Network <- estimateNetwork(PAQ, default = "EBICglasso", corMethod = "cor_auto")
```

The size and density of the edges between the nodes respresent the strength of connectedness. Positively related nodes are shown in blue and negatively related nodes are shown in red.

```
plot(Network, layout="spring", labels=colnames(PAQ))
```

4.9 Interpretation

Parental Authority Questionnaire (PAQ)

1. My family Knows what I do in my free time
2. My family knows friends I have during my free time
3. My family knows how I spend my money
4. My family knows the time of my exams
5. My family knows what I do at school
6. During the past month, my family has no idea if I was at home in the evenings
7. Usually I tell my family about school
8. Usually I tell my family about exams
9. Usually I tell my family about my relationships with teachers
10. Usually I conceal from family a lot of things which I do in the evening
11. When I am out, I tell my family what I did after coming home

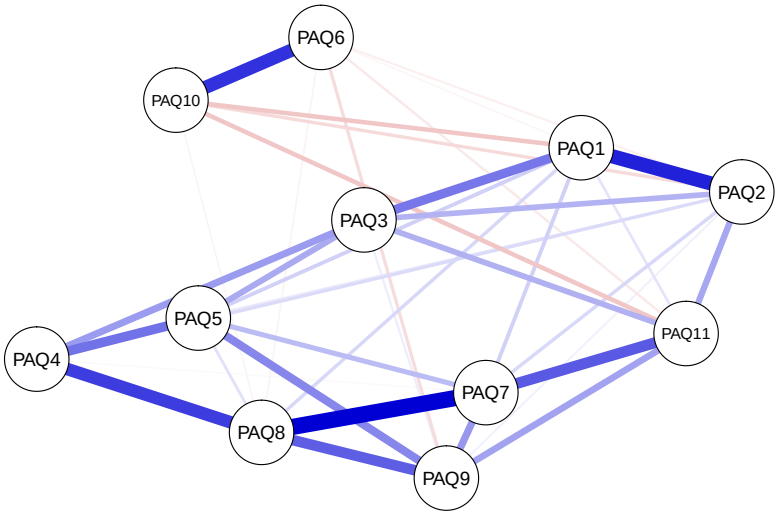

Figure 4.3: PAQ Network Graph

The network shows the strength of *relationships* between the variables. Some variables have quite strong connections (e.g. PAQ8 and PAQ7), whereas others have weak relationship (e.g. PAQ8 and PAQ10). Visual inspection of the network suggests that some variables may be related. However, visual inspection of the graphical display of complex relationships requires careful interpretation, especially if there are a large number of nodes in the network. Borsboom (Borsboom, 2016) recommends exploratory network analysists start by creating manageable networks because too many nodes can lead to a dense network which is difficult to make sense from.

A key concept in network analysis is the centrality of the symptom: if a behaviour or symptom (e.g. fatigue) has many and/or strong associations to other nodes, they are more central within the network than a less connected symptom.There are different centrality metrics, and *strength* most often used. Expected influence is related to strength but argued to sometimes be a better metric

From looking at strength, it appears PAQ7 and PAQ8 have the strongest influence on the nodes in the network. How might we interpret this? Network analysis is purely statistical, the measurement and inclusion of nodes depend on your theory / research question.

```
centralityPlot(Network, include = c("Strength"),
               orderBy = "Strength")
```

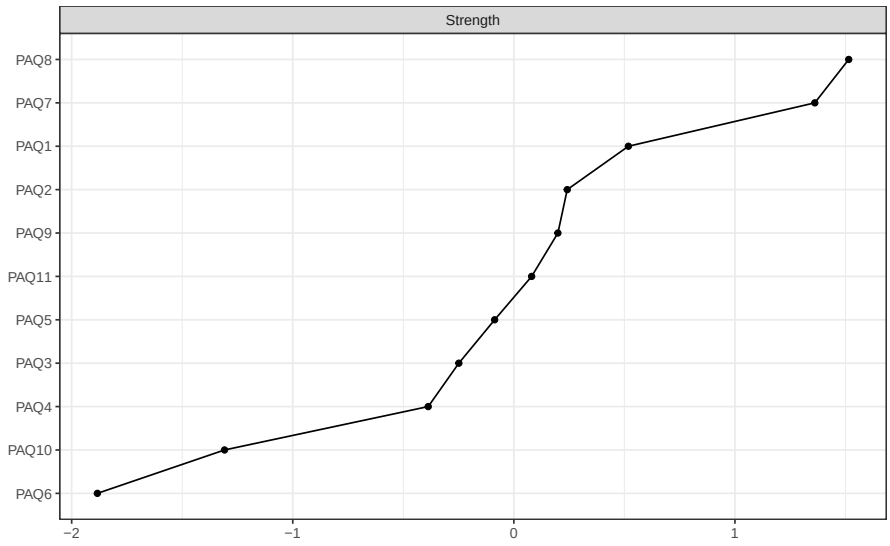

Figure 4.4: Centrality Plot

4.10 Assumptions

Network Analysis assumes normally disturbuted data which can be rare in psychology where Likert scales (such as that used to measure data here) are common.

"our data did not meet all assumptions of multivariate normality. This is not unusual in psychology, but as it is unclear at present how robust the employed methods are to such violations, results need to be interpreted with caution." (Aalbers, 2019)

4.11 Stability

When are estimating networks, we are estimating a structure. While we end up with a model, there is uncertainty about model parameters. Researchers in Amsterdam have developed a bootstrapping procedure so we can estimate stability with more confidence. For more information on this, please see a tutorial by Eiko Fried.

Appendix A

RStudio Cloud

Sometimes R might not work well on our own computers. However, there is an online version of R Studio (R Studio Cloud) which can be used in a pinch. Using R Studio Cloud is a little different to R Studio, so we have made a short guide to get you up and running.

A.1 Creating an Account

Head to RStudio Cloud and click "Sign Up" at the top of the page

Enter details you wish to sign in with and select "Sign up"

You'll receive an e-mail at the address you sign up with, make sure to click the link to activate your account fully.

A.2 Accessing RStudio Cloud

Head back to RStudio Cloud and select "log In", where we previously chose "Sign Up"

Once logged in, you'll be taken to the "Your Workspace" page, this is where all of your RStudio Cloud projects will be accessible from

Select "New Project" and then "New Project" again

You will see the message "Deploying Project" for a couple of minutes while it creates your Workspace

A.3 Getting Started with RStudio Cloud

Once loaded, you'll see a page that looks almost identical to the other screenshots in the learning material

A.3.1 Naming the Workspace

Let's give the project a better name!

Click on "Untitled Project" at the top of the page

This will allow you to rename to whatever you like, in this case we'll go for "Network Training"

Press the return key on your keyboard, or click on a different area on the page to complete the task

A.3.2 Uploading Files

Since this is on the web, files on your computer won't be immediately accessible to RStudio Cloud, you will need to upload them yourself

Click the "Upload" button on the Files tab

An "Upload Files" element will load up where you can click "Browse" and select the file(s) you wish to make available to RStudio Cloud

Once you have selected a file and chosen "OK", you'll be taken back to the main application and you will now see the file you uploaded

You can now interact with this file as described in the rest of the learning material!

Appendix B

References

Network Analysis Cookbook - Also covers R introduction

We are grateful to PsyTeachR from the University of Glasgow for allowing us to build upon their open source teaching materials.

Network Analysis Tutorial

Tasks on section 3 are based on that of Derek De Beurs who uses network analysis to understand suicide ideation. Check out his work, he explains things really well.

Useful website which lets you compare your own code to another