Network Analysis in R

2019-10-20

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.**

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.

1.1.5 Where to Get Help

Chapter 2

Working With Data in R

2.1 Overview

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.

2.1.4 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.

2.1.5 Keeping Environment Clean

As you work, you will notice your environment will 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

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.

3.3 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) can be defined as a graph

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`.

Chapter 4

Practical

4.1 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.

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

4.2 Visualising Data

To do - write a bit about plotting options in R.

4.3 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.

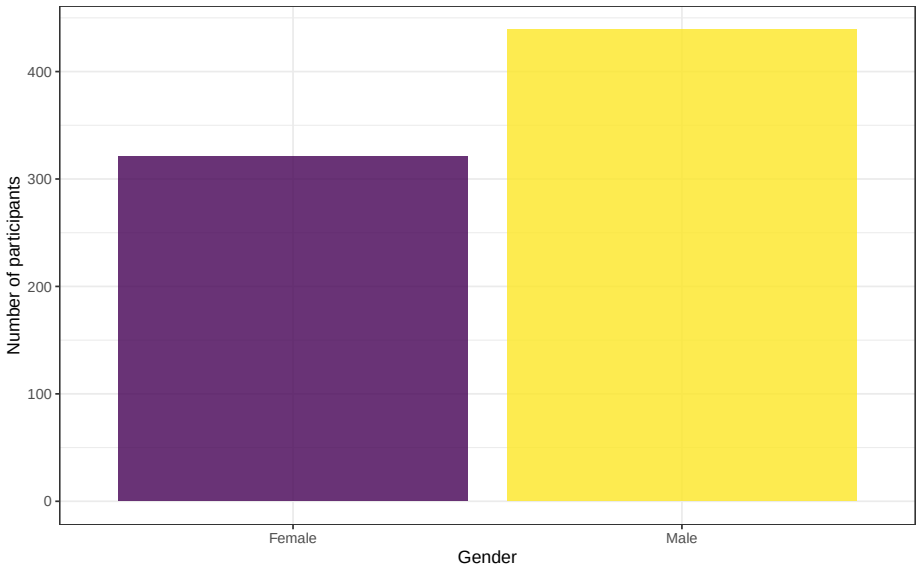

Figure 4.1: Participant Gender

```
# 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, aes(x = as.factor(Gender), fill = as.factor(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")
```

4.4 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.

The pipe operator %>% is something you will see a lot and (in terms of code) means “and then”

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

4.5 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 = as.factor(Place), fill = as.factor(Place), y = tot_TEQ)) + geom_violin(
  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

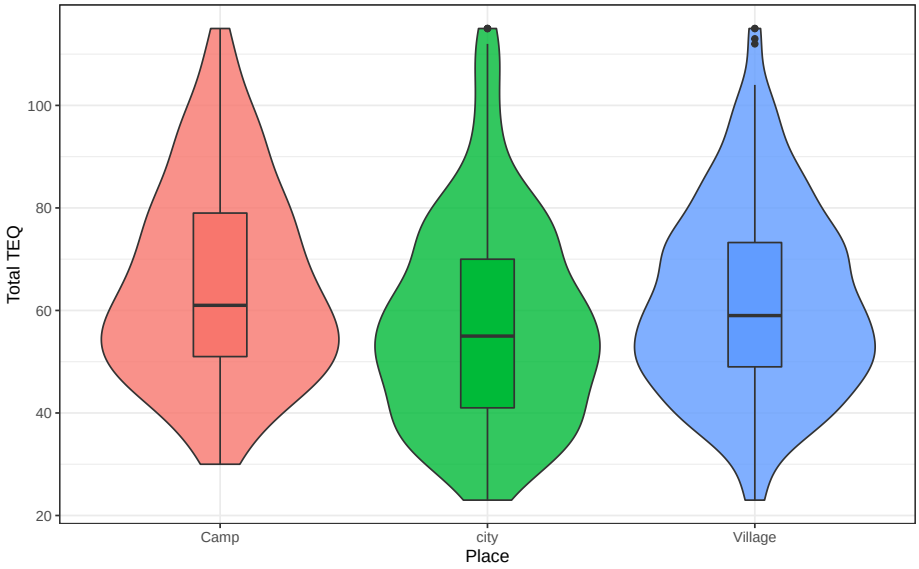

Figure 4.2: TEQ Scores

Appendix A

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.