Advanced Psychological Research Methods (PSY4034)

Christopher J. Wilson

2020-09-10

Contents

Chapter 1

Welcome to the module

Note: You should be able to see a video immediately below this text. To watch the video, you will need to be logged into Teesside University's Blackboard site and be a part of this course. If you are on the course:

- Press *Ctrl* + *T* to open a new tab (keeping this one open) and go to https://eat.tees.ac.uk.
- Login to Blackboard. Keep the blackboard tab open and switch back to this page.
- Refresh this page (hit F5 on your keyboard).
- You should now be able to play all of the video content.

1.1 Module Overview

This Level 7 module for first year Doctorate in Clinical Psychology trainees aims to enable you to:

- Refresh and extend your knowledge, skills and critical understanding of advanced research methods using both qualitative and quantitative approaches;
- Creatively apply the principles of quantitative and qualitative research methods to clinical psychology research and practice;
- Refresh and extend your skills in project design, management, analysis and presentation.

The module is also designed to explicitly prepare you for the two Doctorate level research modules which occur in Years Two and Three of the programme, ensuring that you have the requisite knowledge and skills to successfully engage with those modules.

The key foci for this module include:

- critical review of established literature
- project design
- project management
- data analysis
- dissemination of research findings

The module is taught using a variety of techniques to best enhance your knowledge and understanding of the application of research theory and methods in the context of clinical psychology. These include lectures, seminars, guided statistical analysis and tutorials with the latter being used to provide individual guidance and formative feedback. The module has its own site on the University's Virtual Learning Environment http://eat.tees.ac.uk - known as Blackboard), with resources and literature designed to support learning.

1.2 Module timetable and delivery for 2020/2021

The timetable for this module appears on the Clinical Psychology Programme Site/Timetables. The majority of the sessions will take place on Monday mornings. Please note that the timetable should be checked on a regular basis.

Due to COVID19 pandemic, the sessions for this module will be delivered online in 2020/21

1.2.1 Learning and Teaching Strategies

The module is taught using a variety of techniques to best enhance your knowledge and understanding of the application of research theory and methods in the context of clinical psychology. This includes activities designed to encourage independent learning, a key skill for successful performance in research modules in Years Two and Three of the programme. Evidence of independent learning is expected in the assignments for this module. Specific links are made with research informed activity in practice.

You will be provided with two papers for critical review at the start of the module and asked to decide which one you will use for your summative critical review assignment; one of these papers is from a quantitative research tradition and the other is from a qualitative research tradition.

All presentations (with added annotations) are available, along with additional support materials, via an e-learning@tees on the VLE. E-learning is enabled through group activities on the VLE or Microsoft Teams where discussion and problem solving is undertaken in relation to tasks set during teaching sessions. The discussion boards or Microsoft Teams site will be used to ask and answer questions that arise from the taught material and also your independent work.

1.2.2 How will the online sessions work?

The video below explains how online sessions will work:

1.3 Assessment

1.3.1 Formative assessment

Formative feedback is provided throughout the module through practical exercises and in seminars on trainee presentations.

By the end of year one, trainees are expected to have identified a thesis topic and have a completed research proposal. As such, there are a number of formative milestones across the year that will be monitored by the module team. Please see Appendix 1 of this guide for details.

The required format for thesis research proposals can be found in Appendix 2 of the DClinPsy Programme Research Handbook.

The formal formative assessment is of a presentation of the thesis research proposal to be presented during the research panels, which take place on 20, 21 and 22 July 2020. The presentation will be 20 minutes' long and will outline the thesis project that the trainee will develop in Years 2 and 3. There will also be 10 minutes allotted for questions from the panel which will have two academic members and one clinical member. This formative assessment is intended as a starting point for the Year 2 and 3 research methods modules. The timing is important as it should enable trainees to start the process of ethics approval for their dissertation. earlier. The trainees will hand in a printed copy of their slides with explanatory notes and references.

Formative Assessment Criteria

The following criteria will be used to assess the assignment:

- Effective justification for the study.
- Clearly defined research question.
- Comprehensive and critical review of the literature (within time constraints).
- Realistic research design.
- Effective consideration of ethical issues.
- Clear plan for writing up and dissemination.
- Fulfilment of professional research ethics requirements.
- Adherence to the relevant guidance for presentation as advised by the Module Tutor.

1.3.2 Summative Assessments

Assessment consists of an ICA and an ECA, each worth 50% of the overall module mark. The deadlines for these assessments can be found in the assessments section on Blackboard or in the programme assessment timetable.

ICA (50%) - A critical review of a published primary research paper (choice to be made by a trainee from three papers with different methodologies provided by the tutor). One of these papers is from a quantitative research approach, one from a qualitative research approach and the third will use mixed methods (2,000 words). *Learning outcomes: (KU 1-4, CIS 1-3, KTS 1-3)*

ICA Assessment Criteria (Critical Appraisal of Published Primary Research Paper)

The following criteria will be used to assess the assignment:

- Demonstrate a critical understanding of the role of the reviewed paper for clinical psychologists in service delivery and/or practice.
- Demonstrate a critical and comprehensive understanding of the relevant methodological issues.
- Systematically and critically evaluate stages of the research process.
- Demonstrate a comprehensive and critical understanding of the ethical issues involved in the research.
- Reach effectively argued conclusions.
- Demonstrate independent learning ability through reflection on the critical review process.
- Adhere to the American Psychological Association (APA) guidelines for presentation and referencing.

ECA (50%) - A research project proposal which both addresses limitations identified in the ICA critical review (1) and develops the research further with the use of an alternative methodology (2,000 words).
Learning outcomes: (KU 1-4, CIS 1-3, PPS 1-2, KTS 1-5)

ECA Assessment Criteria (Research Project Proposal)

The following criteria will be used to assess the assignment:

- Identify a project that demonstrates a detailed and critical understanding of the research evidence reviewed and wider methodological issues, including the role of the project for informing service delivery and/or practice.
- Provide detailed and appropriately justified solutions to the design of the research project.
- Consider both methodological and ethical issues in the design of the research project.
- Demonstrate a detailed and critical understanding of the data analysis required for the proposed study.
- Adhere to the American Psychological Association (APA) guidelines for presentation and referencing.

1.3.3 Word limits for assessment

Word limits are as stated above (note that there is no allowance of +10%). The word count refers to the assignment itself and does not include the reference list or tables/graphs. The references cited in the main body of text are included in the word count.

1.3.4 Submitting work

All work should be submitted electronically, using the apporpriate links on the module Blackboard site. A printed hard copy of the assignment is **not** required and should not be submitted. **The university's policy is that all assessments must be submitted by 4 p.m. on the day of the deadline.**

1.3.5 A note about referencing

There is an expectation that all academic assignments conform to current American Psychological Association referencing and citation conventions. Poor referencing will be taken into consideration when marking. It is recommended that you use a digital reference management system (e.g., Refworks, Mendley), which are freely available (and will save you time). The following online resources are also useful:

http://reciteworks.com/ - good for checking fine details (e.g., missing references)

http://www.apastyle.org - detailed guidance for APA style

https://owl.english.purdue.edu/owl/resource/560/01/- additional advice for APA style

1.4 Statistical analysis software

You may be familiar with SPSS from your undergraduate statistics teaching. Please note that we do not use SPSS for teaching and instead use R Statistics. The reason for this is that R is a free statistical package, meaning that it can be accessed in NHS settings that do not have funding for SPSS. This will enable you to run statistical analyses whilst on placement where required, and also enables you to conduct statistical analyses as a qualified Psychologist without incurring any software costs. R is also more flexible than SPSS and has greater functionality. During the teaching you will be shown how to set up and install R, and how to run statistical analyses in this software.

1.4.1 Downloading R and R Studio

You can obtain R and R Studio from the following links:

https://cran.r-project.org/

https://rstudio.com/

1.5 Academic Support and Guidance

Please contact the module team if you have any questions, concerns or any other areas you wish to discuss.

1.5.1 Module Team Contact Details

Module Leader: Dr Christopher Wilson: christopher.wilson@tees.ac.uk

Module Team: Dr Alan Bowman: A.Bowman@tees.ac.uk

Guest lecturers from Schools within the University and Local NHS clinicians also contribute to some teaching

1.6 Considering your thesis

Your doctoral thesis is one of the largest pieces of work you will undertake during your training and it is important to start thinking about it early on.

You are advised to read around the area of your thesis topic on a continuing basis, and make use of tutorials with your supervision team when needed.

It is also advised that you consider research governance and ethics as you develop your project. Please speak to your academic supervisor about this as they will be able to advise or direct you to someone with appropriate expertise to address queries.

As specified in the Research Handbook, **please note that a revised version of your research proposal forms one of your year two summative assignments**. The deadline for this assignment is **early on in the start of second year** and it is therefore advised that you work on your revised proposal as soon as you receive feedback from the panels and have discussed this with your academic supervisor.

Chapter 2

Research Methods Concepts Revision

2.1 Basic concepts that you should already know

2.2 Probability and hypothesis testing

One of the most important things to remember about hypothesis testing in statistics is why we use the approaches we do. That is, we need statistical approaches to test hypotheses because we can only collect data from samples of the population but our research questions and hypotheses apply to whole populations. For that reason, we need a way to estimate how well the *sample* reflects the *population.*

It is common for us to want to know what the mean (average) response of the population is on certain measures. For example, we might ask the question "what is the average score on this measure of happiness?". In reality we can only measure a subset (sample) of the population, so we test as many people as we can. Below is a sample of 20 participants:

```
## Warning: package 'kableExtra' was built under R version 4.0.2

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Table 2.1: Some sample data for happiness score

participant	intervention	happiness
1	1	11.126738
2	2	11.455096
3	1	9.448889
4	1	6.567100
5	2	11.080759
6	1	9.866935
7	1	11.066079
8	2	7.115263
9	1	8.478822
10	2	9.126518
11	1	11.077905
12	2	11.238149
13	2	6.796180
14	2	7.745382
15	1	9.542996
16	1	7.913203
17	1	9.601031
18	1	10.571534
19	2	9.679823
20	2	12.890654

2.3 Variance in sample data influences our confidence in population estimates

We can see from the table above that the mean of the sample is 9.6194528. However, this is not to say that the population mean is 9.6194528. For one thing, we can see that the range of scores in the sample is between 6.5671002 and 12.890654. The standard deviation of the sample is 2.

The fact that there is so much variance from person to person within our sample indicates that we are likely to be incorrect if we assume that the sample mean is the same as the population mean. The more variance there is within the sample data, the less confident we can be that the sample mean is an accurate representation of the population mean.

Another thing that affects our ability to generalise from sample to population is that the sample size is only 20. Larger samples are less influenced by individual outliers, so the larger the sample size is, the more confident we can be that the sample mean is representative of the population mean (provided that the participant sample is representative of the population and recruited in a way to minimise bias).

> The **standard error** of the mean can be calculated to estimate how far the mean of the sample data is likely to be from the true population mean. It uses the concepts of variance and sample size to make this estimate. Standard error is calculated by dividing standard deviation by the square root of the sample size ($SE = \frac{SD}{\sqrt{[n]}}$)

In R, we can calculate the standard error of the happiness data like so:

```
 standardError <- sd(happiness)/sqrt(length(happiness)) ## Calculate standard error

standardError #Display the standard error
```

```
## [1] 0.3901769
```

The standard error of our sample mean is 0.39. This suggests that using the sample mean is likely to be 0.39 away from the population mean.

2.4 We can use confidence intervals to make educated guesses about the population mean

Using the standard error, we can also create **Confidence intervals**, which are a range of values, within which the population mean is likely to fall. For example, we know from normal distribution that 95% of the population lies between +/- 1.96 standard deviations of the mean. If we use our sample mean ($\bar{x}$) in place of the population mean and include the standard error to account for errors in our estimate, we come up with the following formula for 95% confidence intervals of the mean:

Lower confidence interval $= \bar{x} - 1.96 * SE$

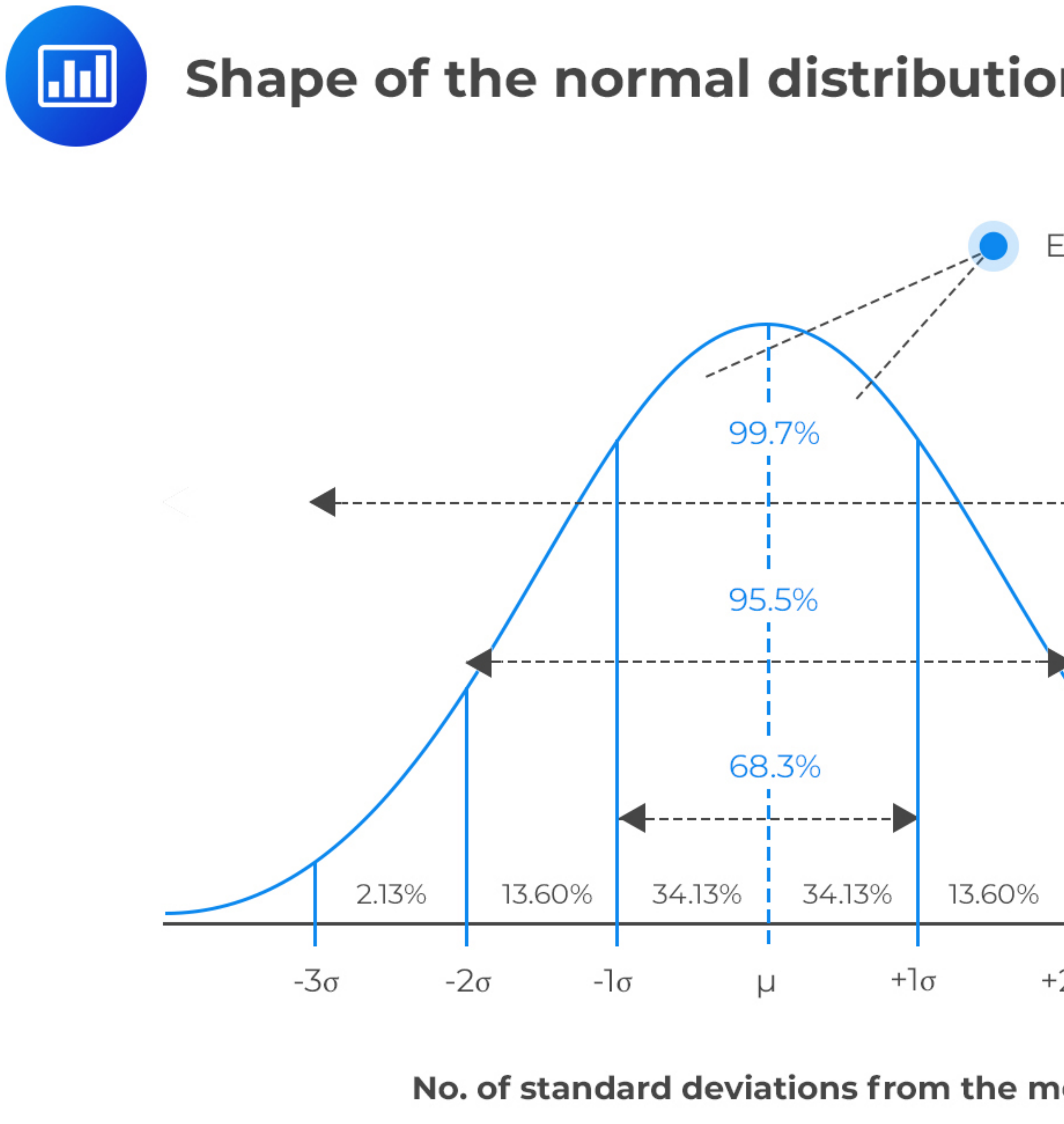
Upper confidence interval $= \bar{x} + 1.96 * SE$

```r
mean(happiness) - 1.96 * standardError # Lower confidence interval
```

```
## [1] 8.854706
```

```r
mean(happiness) + 1.96 * standardError # Upper confidence interval
```

```
## [1] 10.3842
```


Shape of the normal distributio
E
99.7%
95.5%
68.3%
2.13%
13.60%
34.13%
34.13%
13.60%
-3σ
-2σ
-1σ
μ
+1σ
+2
No. of standard deviations from the m

The value of 1.96 come from the normal distribution, where 95% of the population lies between +/- 1.96 standard deviations of the mean. If we did not already know this, we could use the *qnorm()* function in R to calculate the value:

```
# Calculate the number of standard deviations that contains 0% to 97.5% of the data (1
# We can then say that 95% of the data lies between + or - the answer:

qnorm(0.975)
```

```
## [1] 1.959964
```

However, with smaller samples, since we are less confident about generalising to the population, we use the t-distribution to calculate that value. The shape of a t-distribution changes based on the sample size, so the smaller the sample size is, the wider the range that 95% of values lie between. We can calculate the 95% value for a particular sample size in R using the *qt()* function:

```
# The qt function relates to t-distribution

qt(0.975,df=20-1)
```

```
## [1] 2.093024
```

```
# for qt, we need to specify the degress of freedom, which is sample size minus 1
```

We can see that when we have a sample size of 20, 95% of values in our predicted population distribution will lie between + and - 2.0930241 standard deviations. Therefore, we can calculate more accurate confidence intervals using this value:

```
mean(happiness) - qt(0.975,df=20-1) * standardError # Lower confidence interval
```

```
## [1] 8.802803
```

```
mean(happiness) + qt(0.975,df=20-1) * standardError # Upper confidence interval
```

```
## [1] 10.4361
```

This tells us: if we were to take infinite number of similar samples, about 95% of their confidence intervals would contain the population mean. Therefore, we think it is reasonable to estimate that the population mean is somewhere in this range.

> Often people say that a 95% confidence interval means that there is a 95% chance that the population mean is between the lower and upper confidence interval. This is **not** an accurate statement, but it is often used as a shorthand to help people conceptualise what confidence intervals are.

Table 2.2: Some sample data for happiness score with participants divided into 2 groups

participant	intervention	happiness
1	1	11.126738
2	2	11.455096
3	1	9.448889
4	1	6.567100
5	2	11.080759
6	1	9.866935
7	1	11.066079
8	2	7.115263
9	1	8.478822
10	2	9.126518
11	1	11.077905
12	2	11.238149
13	2	6.796180
14	2	7.745382
15	1	9.542996
16	1	7.913203
17	1	9.601031
18	1	10.571534
19	2	9.679823
20	2	12.890654

2.5 We can also make confidence intervals of differences between means

Often when we test hypotheses, we are testing the difference between two samples. For example, we might have 2 groups who have undergone different psychological interventions and want to know whether the difference we see our participant samples is likely to generalise to the population.

Using the same approach as in the previous section, we can estimate a confidence interval based on the difference in means and the sample size:

```
# Calculate the number of standard deviations for 95% of the data
qt(0.975,df=20-2) # since there are 2 intervention groups, degrees of freedom is now 20-2
```

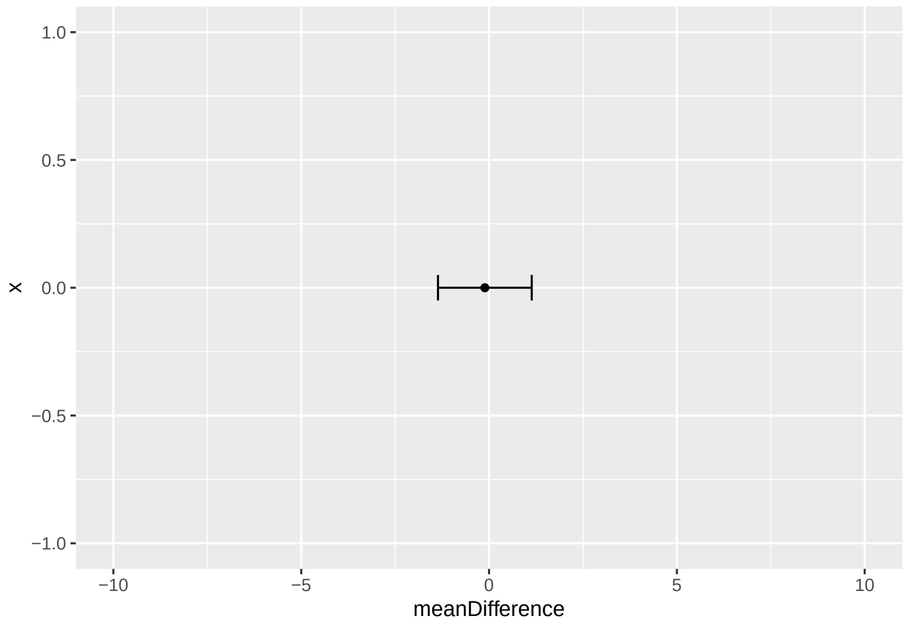
```
## [1] 2.100922
```

```
 group1 <- happinessSample %>% filter(intervention ==1) %>% summarise(mean = mean(happiness), sd=

  group2 <- happinessSample %>% filter(intervention ==2) %>% summarise(mean = mean(happiness), sd
# calculate mean of difference
meanDifference <- group1$mean - group2$mean
seDifference <- sqrt(((group1$sd^2)/19) + ((group2$sd^2)/19))


# calculate 95% CI of this
meanDifference - seDifference * qt(c(0.975), 20-2) # lower CI
```

```
## [1] -1.35913
```

```
meanDifference + seDifference * qt(c(0.975), 20-2) # upper CI
```

```
## [1] 1.135797
```



This tells use that the 95% confidence interval of the difference is between -1.3591301 and 1.1357974. An important part of interpreting this, is to notice whether any point between these values is equal to zero. If the confidence interval of a difference contains a zero value, this means that in future research, with similar samples, it would be possible to see zero difference between the groups. If, on the other hand, the confidence interval does not cross zero, then it is likely that in future research, with similar samples, we would see some difference between the means.

> The fact of whether confidence intervals cross zero (or not) is linked directly to the idea of hypothesis testing and statistical significance.

2.6 The null hypothesis and statistical significance

Using the same study from the previous example: we know that the null hypothesis can be phrased as "in the population, there is no difference between groups". We then see how the confidence interval of a difference can help us test the null hypothesis: if the null hypothesis were not true, then it is unlikely that the confidence interval of the difference would contain zero.

Therefore, if confidence intervals overlap, then there is a possibility of no difference existing between the populations. As such, we are unable to reject the null hypothesis.

Chapter 3

Introduction to R and R Studio

3.1 By the end of this section, you should be able to:

- Download R and R studio
- Identify the R script, R console, Data environment and file browser in R studio
- Write and run R code from a script
- Install and load R packages

3.2 Why learn / use R?

3.2.1 Some information about R

- R is developed and used by scientists and researchers around the world
- Open source = no cost
- Constant development
- Connects to other data science/research tools
- Worldwide community: training widely available
- Encourages transparency and reproducibility
- Publication-ready outputs

3.2.2 Moving from other software to R

- Workflow is different
  - Organise files and data differently
  - Workspace can contain data and outputs

  - Can manage multiple datasets within a workspace
- Learning curve can be steep initially
  - e.g. Variables and coding, scripts
- Need to know what you want
  - e.g. building your regression model / ANOVA error terms

3.3 R has many advantages

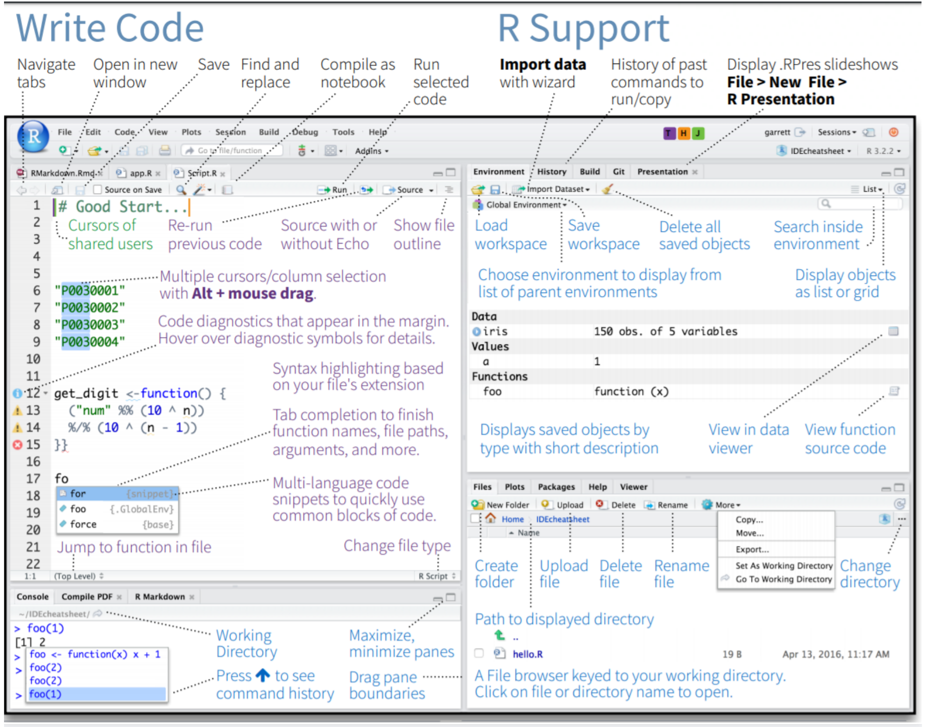
- Using scripts means analysis is easy to follow and reproduce
- R scripts are small, online collaboration, no SPSS "older version" problems
- Data can be organised and reorganised however you need it (tidyr)
- Packages are available for "cutting edge" analysis: e.g. Big Data & Machine Learning
- A robust language for precise plots and graphics (ggplot)
- R analysis code can be embdeded into documents and presentations (R Markdown)

3.4 Download R and R Studio

Click on these links to download:

- R project
- RStudio

3.5 The R Studio environment

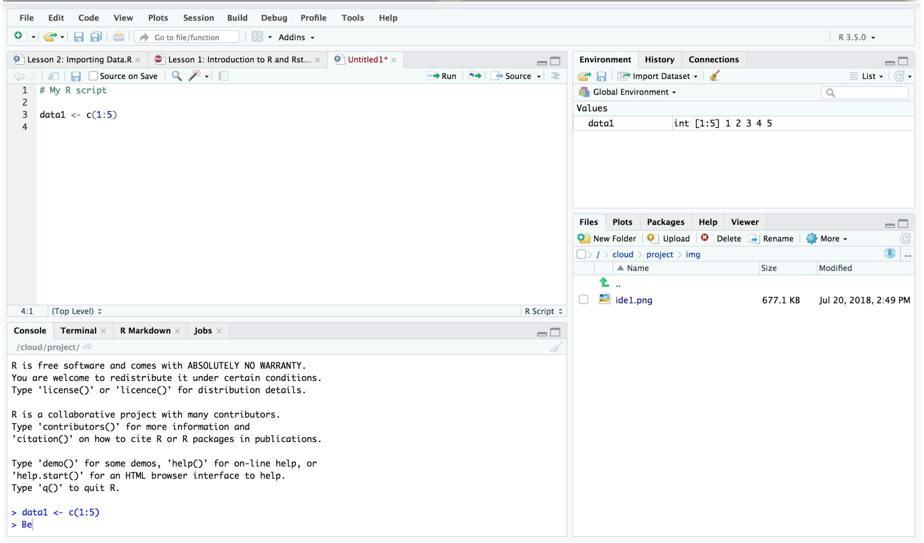

Write Code
Navigate tabs
Open in new window
Save
Find and replace
Compile as notebook
Run selected code
R Support
Import data with wizard
History of past commands to run/copy
Display .RPres slideshows File > New File > R Presentation
Cursors of shared users
Re-run previous code
Source with or without Echo
Show file outline
Multiple cursors/column selection with Alt + mouse drag.
Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
Syntax highlighting based on your file's extension
Tab completion to finish function names, file paths, arguments, and more.
Multi-language code snippets to quickly use common blocks of code.
Jump to function in file
Change file type
Working Directory
Maximize, minimize panes
Press ↑ to see command history
Drag pane boundaries
Load workspace
Save workspace
Delete all saved objects
Search inside environment
Choose environment to display from list of parent environments
Display objects as list or grid
Displays saved objects by type with short description
View in data viewer
View function source code
Create folder
Upload file
Delete file
Rename file
Change directory
Path to displayed directory
A File browser keyed to your working directory. Click on file or directory name to open.

```
# Good Start...

"P0030001"
"P0030002"
"P0030003"
"P0030004"

get_digit <-function() {
  ("num" %% (10 ^ n))
  %/% (10 ^ (n - 1))
}}

fo
```

```
> foo(1)
[1] 2
> foo <- function(x) x + 1
> foo(2)
  foo(2)
> foo(1)
```

```
Data
iris        150 obs. of 5 variables
Values
a           1
Functions
foo         function (x)
```

```
hello.R        19 B    Apr 13, 2016, 11:17 AM
```



```
# My R script

data1 <- c(1:5)
```

```
Values
data1     int [1:5] 1 2 3 4 5
```

```
Name       Size      Modified
ide1.png   677.1 KB  Jul 20, 2018, 2:49 PM
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> data1 <- c(1:5)
> Be
```

The interface for R Studio looks daunting at first. However, there are 4 main sections, 2 on the left and 2 on the right.

- MAIN TOP: R Script files or R Document Files

  - Where we usually type our code as a script before we run it. Script files are usually saved so we can work on them and rerun the code again later (.R files).
- MAIN BOTTOM: Console
  - Shows the output of our R code. We can type R code directly into the console and the answer will ouput immediately. However, it is more convenient to use script files.
- RIGHT TOP: Environment
  - Contains all of the objects (e.g. data, analysis, equations, plots) that are currently stored in memory. We can save all of this to a file and load it later (.RData files).
- RIGHT BOTTOM: File Browser
  - The folder that R is working from is called 'the working directory' and it will automatically look for files there if we try to import something (e.g. a data file). Using the more button on the file browser allows you to set your desired working directory.

3.6 Working with a script

Scripts can be opened from the **File** menu.

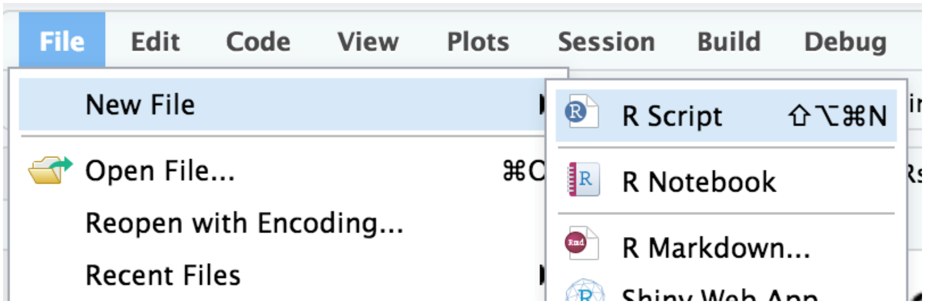


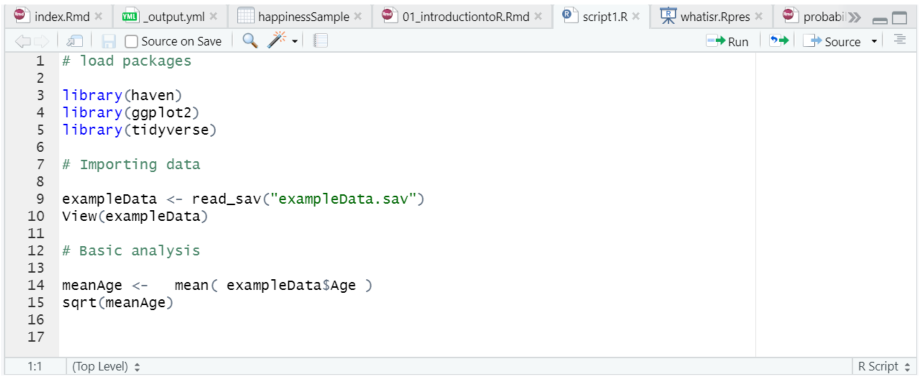
Figure 3.1: Creating a new script

The purpose of scripts is to allow you to type your analysis code and save it for use later. Scripts include, for example:

- Code for importing data into R
- Your analysis code (e.g. t-test or descriptive statistics)
- Code for graphs and tables
- Comments and notes (preceded by the '#' symbol)

To run a script, you click the **Run** button. You can choose to:

- Run the whole script
- Run the selected line of code



```
# load packages

library(haven)
library(ggplot2)
library(tidyverse)

# Importing data

exampleData <- read_sav("exampleData.sav")
View(exampleData)

# Basic analysis

meanAge <-   mean( exampleData$Age )
sqrt(meanAge)
```

Figure 3.2: Example of an R script

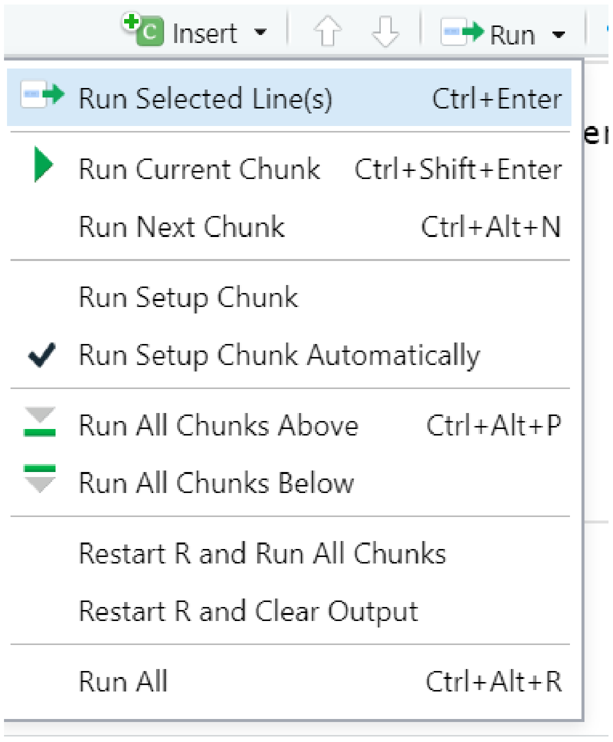
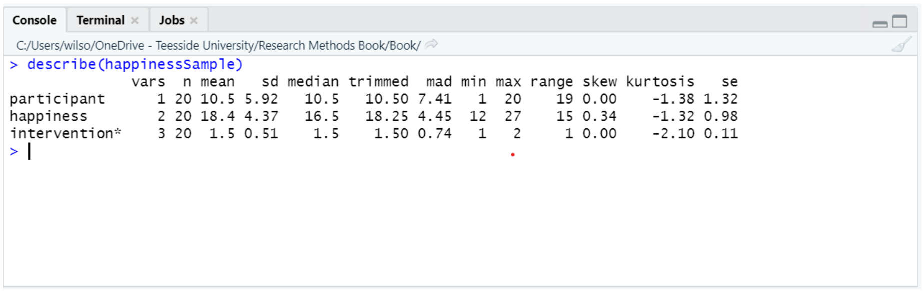

Insert
Run
Run Selected Line(s) Ctrl+Enter
Run Current Chunk Ctrl+Shift+Enter
Run Next Chunk Ctrl+Alt+N
Run Setup Chunk
Run Setup Chunk Automatically
Run All Chunks Above Ctrl+Alt+P
Run All Chunks Below
Restart R and Run All Chunks
Restart R and Clear Output
Run All Ctrl+Alt+R

Figure 3.3: The run button

When you run the script, you will normally see output in the **console.**


Console
Terminal
Jobs
C:/Users/wilso/OneDrive - Teesside University/Research Methods Book/Book/

```
> describe(happinessSample)
              vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
participant      1 20 10.5 5.92   10.5   10.50 7.41   1  20    19 0.00    -1.38 1.32
happiness        2 20 18.4 4.37   16.5   18.25 4.45  12  27    15 0.34    -1.32 0.98
intervention*    3 20  1.5 0.51    1.5    1.50 0.74   1   2     1 0.00    -2.10 0.11
> 
```

Figure 3.4: Output appears in the console

If your script contains code for a plot (graph), it will appear in the **Plots** window in the bottom right.

3.7 Installing and loading packages

install Packages from RStudio, Inc. on Vimeo.

Packages add functionality to R and allow us to do new types of analysis.

- They can be installed via the menu (Tools -> Install Packages)
- The can also be installed using code:

  `install.packages()`

For example, TidyR is a package that contains functions for sorting and organising data. To install the package:

or use the code:

`install.packages("tidyr")`

Once a package is has been installed, you need tp load it using the *library()* command. For example:

`library("tidyr")`

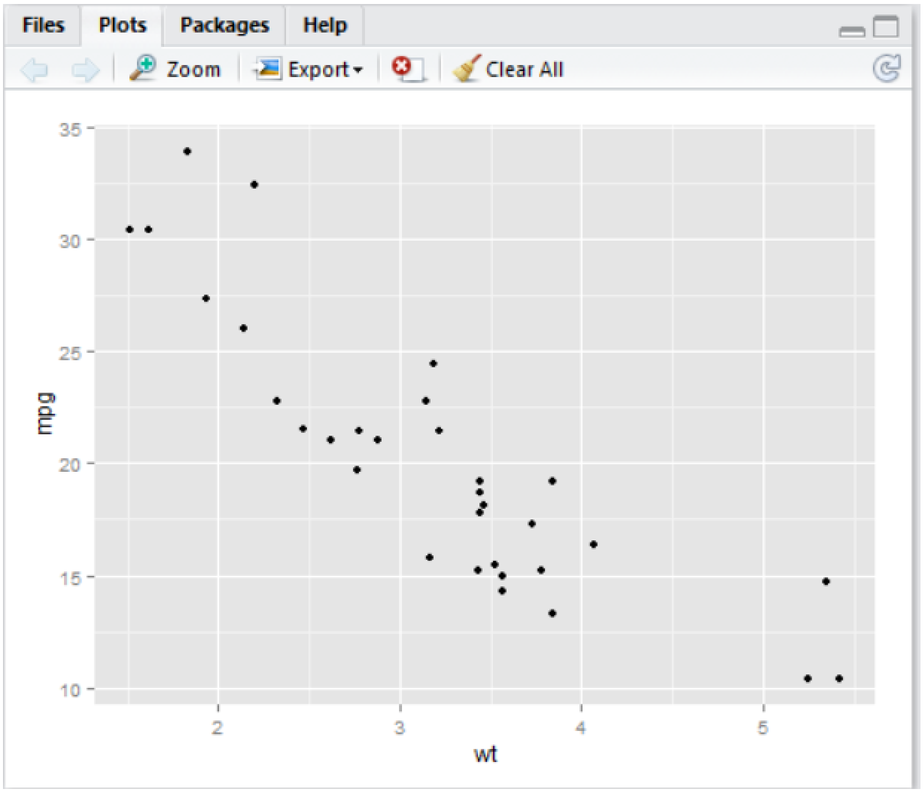


Figure 3.5: Plots appear in the plot window

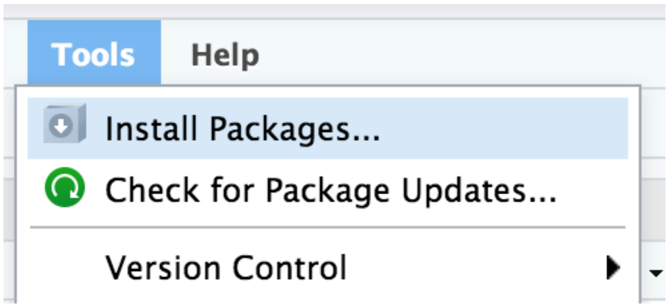


Figure 3.6: Installing a package in RStudio

Chapter 4

Working with data in R

4.1 By the end of this section, you will be able to:

- Import data into R from excel, SPSS and csv files
- Save data to objects
- Identify different data structures and variable types
- Convert variables from one type to another
- Order, filter and group data
- Summarise data
- Create new variables from data

4.2 In this section, we will use the Tidyverse set of packages

- A 'toolkit' of packages that are very useful for organsing and manipulating data
- We will use the haven package to import SPSS files
- We will use the dplyr to organise data
- Also includes the ggplot2 and tidyR packages which we will use later

To install:

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

(See the previous section on installing packages)

4.3 Import data into R from excel, SPSS and csv files

We can import data from a range of sources using the **Import Dataset** button in the **Environment** tab:

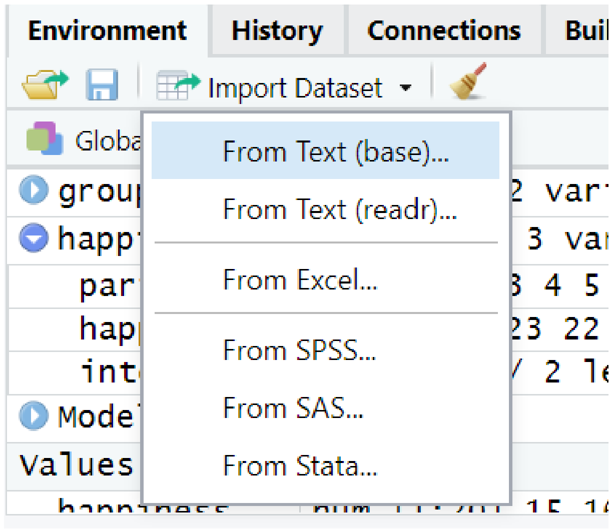


Figure 4.1: Importing data

It is also possible to import data using code, for example:

```
# importing a .csv file
library(readr)
studentData <- read_csv("Datasets/studentData.csv")


#importing an SPSS file
library(haven)
mySPSSData <- read_sav("Datasets/salesData.sav")
```

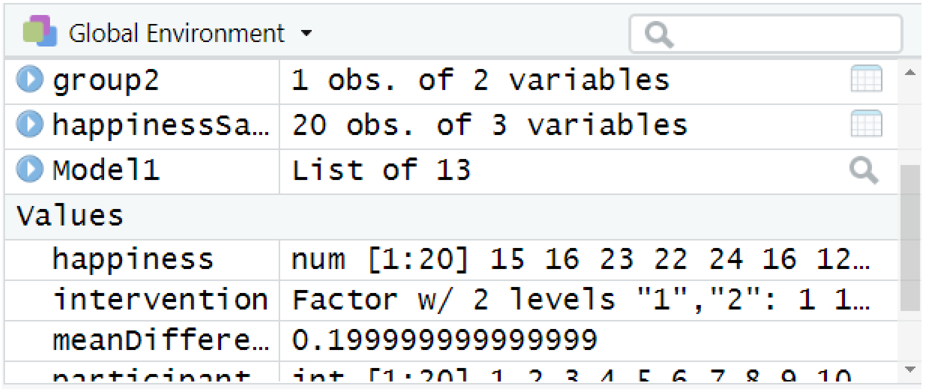
Once the data are imported, it will be visible in the environment:

4.4 Understanding objects in R

In R, an **object** is anything that is saved to memory. For example, we might do some analysis:

```
mean(happiness)
```

However, in the example above, the result would appear in the console but not be saved anywhere. To store the result for reuse later, we save it to an object:



```
group2              1 obs. of 2 variables
happinessSa…        20 obs. of 3 variables
Model1              List of 13
Values
  happiness         num [1:20] 15 16 23 22 24 16 12…
  intervention      Factor w/ 2 levels "1","2": 1 1…
  meanDiffere…      0.199999999999999
```

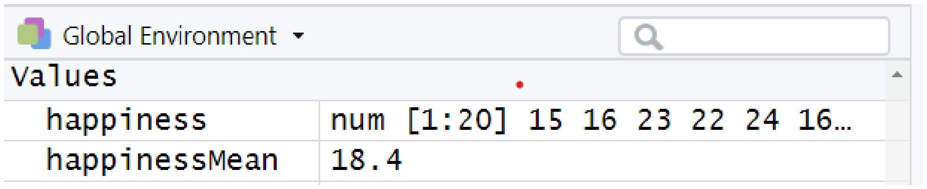
Figure 4.2: Imported data in the environment

```
happinessMean <- mean(happiness)
```

In the above code (reading left to right):

- We name the object "happinessMean". This name can be anything we want.
- The arrow means that the result of the code on the right will be saved to the object on the left.
- The code on the right of the arrow calculates the mean of *happiness* data

When this code is run, *happinessMean* will be stored in the environment window:



```
Values
  happiness         num [1:20] 15 16 23 22 24 16…
  happinessMean     18.4
```

Figure 4.3: Result of a calculation in the environment

To recall an object from the environment, we can simply type its name. For example:

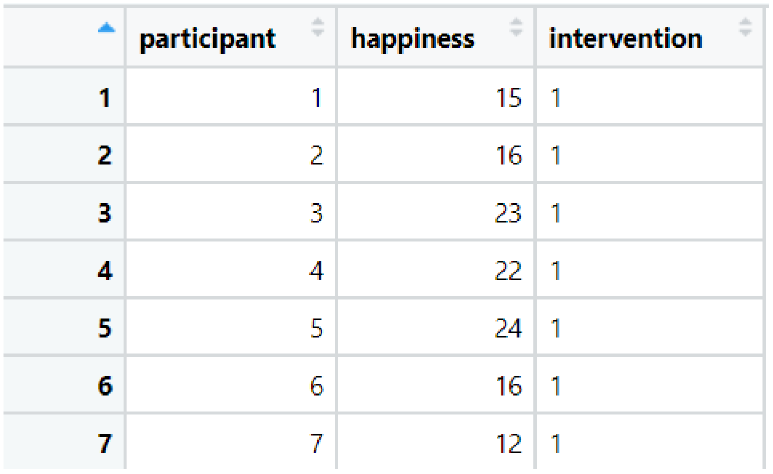
```
happinessMean
```

> Its important to note that anything can be stored as an object in R and recalled later. This includes, dataframes, the results of statistical calculations, plots etc.

4.5 Identify different data structures and variable types

4.5.1 Data structures

There are many different types of data that R can work with. The most common type of data for most people tends to be a **dataframe**. A **dataframe** is what you might consider a "normal" 2-dimensional dataset, with rows of data and columns of variables:



	participant	happiness	intervention
1	1	15	1
2	2	16	1
3	3	23	1
4	4	22	1
5	5	24	1
6	6	16	1
7	7	12	1

Figure 4.4: A dataframe example

R can also use other data types.

A vector is a one-dimensional set of values:

```r
# a vector example

scores <- c(1,4,6,8,3,4,6,7)
```

A matrix is a multi-dimensional set of values. The below example is a 3-dimensional matrix, there are 2 groups of 2 rows and 3 columns:

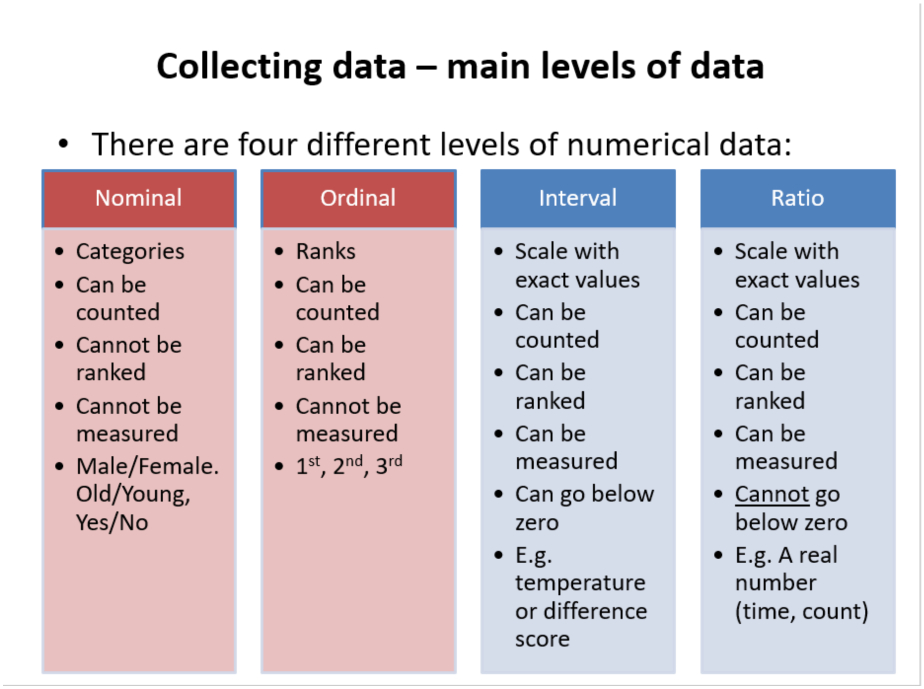
```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    7    9   11
## [2,]    8   10   12
```

We will primarily work with dataframes (and sometimes vectors), as this is how the data in psychology research is usually structured.

4.5.2 Variable types

With numerical data, there are 4 key data types:

- Nominal (a category, group or factor)
- Ordinal (a ranking)
- Interval (scale data that can include negative values)
- Ratio (scale data that cannot include negative values)



Collecting data – main levels of data

- There are four different levels of numerical data:

Nominal	Ordinal	Interval	Ratio
• Categories • Can be counted • Cannot be ranked • Cannot be measured • Male/Female. Old/Young, Yes/No	• Ranks • Can be counted • Can be ranked • Cannot be measured • 1st, 2nd, 3rd	• Scale with exact values • Can be counted • Can be ranked • Can be measured • Can go below zero • E.g. temperature or difference score	• Scale with exact values • Can be counted • Can be ranked • Can be measured • Cannot go below zero • E.g. A real number (time, count)

R can use all of these variable types:

- **Nominal** variables are called **factors**
- **Ordinal** variables are called **ordered factors**
- **Interval and ratio** variables are called **numeric** data and can sometimes be called integers (if they are only whole numbers) or doubles (if they all have decimal points)

R can also use other data types such as text (**character**) data.

4.5.3 Convert variables from one type to another

When we first import data into R, it might not recognise the data types correctly. For example, in the below data, we can see the **intervention** variable :

```
##    participant intervention happiness
## 1            4            1  6.567100
## 2           13            2  6.796180
## 3            8            2  7.115263
## 4           14            2  7.745382
## 5           16            1  7.913203
## 6            9            1  8.478822
## 7           10            2  9.126518
## 8            3            1  9.448889
## 9           15            1  9.542996
## 10          17            1  9.601031
```

In the **intervention** variable, the numbers 1 and 2 refer to different intervention groups. Therefore, the variable is a **factor** variable. To ensure that R understands this, we can resave the intervention variable as a factor using the *as.factor()* function:

```
happinessSample$intervention <- as.factor(happinessSample$intervention)
```

4.6 Working with dataframes

Dataframes are the more standard data format that were are used to (think of how a dataset looks in SPSS or Excel).

In a dataframe, variables are columns and each row usually reperesents one measurement or one participant.

4.6.1 View dataframe

To view a dataframe, we can click on it in the environment window and it will display:

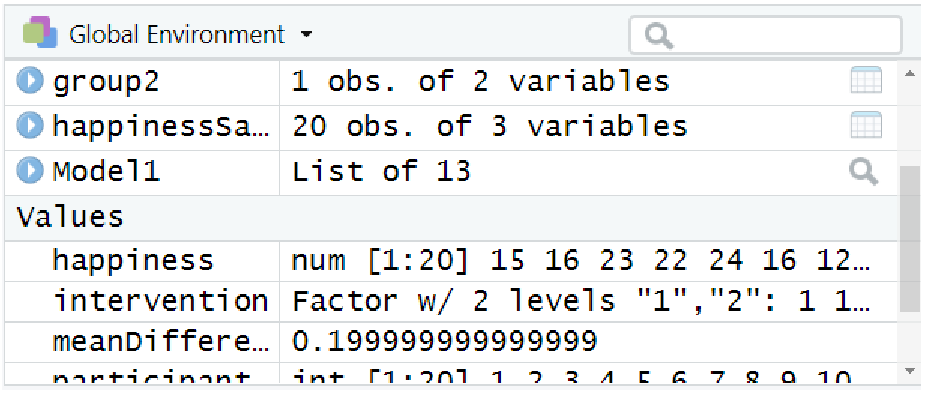
4.6.2 Refer to variables (columns) in a dataframe

Columns in a dataframe are accessed using the "$" sign. For example, to access the *happiness* column in the *happinessSample* dataframe, we would type:

```
happinessSample$happiness
```

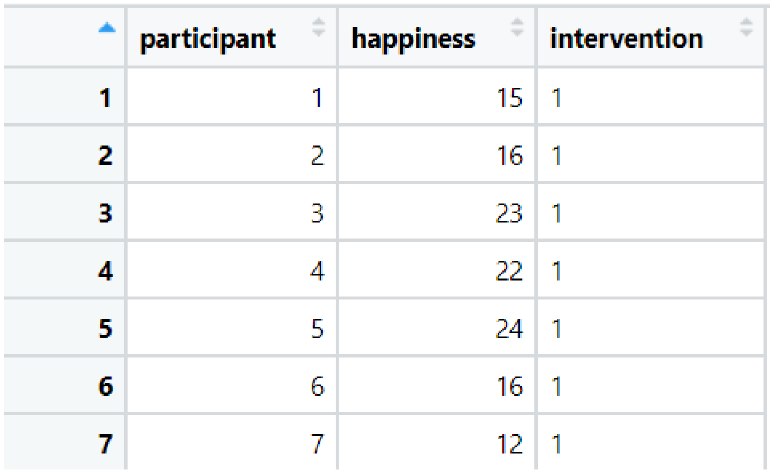
```
##  [1] 11.126738 11.455096  9.448889  6.567100 11.080759  9.866935 11.066079
##  [8]  7.115263  8.478822  9.126518 11.077905 11.238149  6.796180  7.745382
## [15]  9.542996  7.913203  9.601031 10.571534  9.679823 12.890654
```

As we can see above, the result is then displayed.



```
group2          1 obs. of 2 variables
happinessSa…    20 obs. of 3 variables
Model1          List of 13
Values
  happiness     num [1:20] 15 16 23 22 24 16 12…
  intervention  Factor w/ 2 levels "1","2": 1 1…
  meanDiffere…  0.199999999999999
  participant   int [1:20] 1 2 3 4 5 6 7 8 9 10
```

Figure 4.5: Clicking on datasets in hte environment will open them up for viewing



	participant	happiness	intervention
1	1	15	1
2	2	16	1
3	3	23	1
4	4	22	1
5	5	24	1
6	6	16	1
7	7	12	1

Figure 4.6: Viewing a dataframe

4.7 Order, filter and group data

If you have the **tidyverse** package loaded, it is easy to organise and filter data.

```
arrange(happinessSample, happiness)
```

```
##    participant intervention happiness
## 1            4            1  6.567100
## 2           13            2  6.796180
## 3            8            2  7.115263
## 4           14            2  7.745382
## 5           16            1  7.913203
## 6            9            1  8.478822
## 7           10            2  9.126518
## 8            3            1  9.448889
## 9           15            1  9.542996
## 10          17            1  9.601031
## 11          19            2  9.679823
## 12           6            1  9.866935
## 13          18            1 10.571534
## 14           7            1 11.066079
## 15          11            1 11.077905
## 16           5            2 11.080759
## 17           1            1 11.126738
## 18          12            2 11.238149
## 19           2            2 11.455096
## 20          20            2 12.890654
```

```
arrange(happinessSample, desc(happiness)) # Arrange in descending order
```

```
##    participant intervention happiness
## 1           20            2 12.890654
## 2            2            2 11.455096
## 3           12            2 11.238149
## 4            1            1 11.126738
## 5            5            2 11.080759
## 6           11            1 11.077905
## 7            7            1 11.066079
## 8           18            1 10.571534
## 9            6            1  9.866935
## 10          19            2  9.679823
## 11          17            1  9.601031
## 12          15            1  9.542996
## 13           3            1  9.448889
## 14          10            2  9.126518
## 15           9            1  8.478822
## 16          16            1  7.913203
## 17          14            2  7.745382
## 18           8            2  7.115263
## 19          13            2  6.796180
## 20           4            1  6.567100
```

- Show clients with a happiness score of less than 4

```
filter(happinessSample, happiness < 4)
```

```
## [1] participant  intervention happiness
## <0 rows> (or 0-length row.names)
```

- Show Intervention group 2 with happiness scores above 7

```
filter(happinessSample, happiness > 7 & intervention == 2)
```

```
##   participant intervention happiness
## 1           2            2 11.455096
## 2           5            2 11.080759
## 3           8            2  7.115263
## 4          10            2  9.126518
## 5          12            2 11.238149
## 6          14            2  7.745382
## 7          19            2  9.679823
## 8          20            2 12.890654
```

- Group by intervention and show the mean happiness score

```
happinessSample %>% group_by(intervention) %>% summarise(mean = mean(happiness))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   intervention  mean
##   <fct>        <dbl>
## 1 1             9.57
## 2 2             9.68
```

4.8 Create new variables from data

To create new variables from data, we can use the **mutate()** function.

For example, let's say we wanted to calculate the difference between each person's happiness score and the mean happiness score.

We could do the following:

```
happinessSample %>% mutate(difference = happiness - mean(happiness))
```

```
##    participant intervention happiness  difference
## 1            1            1 11.126738  1.50728529
## 2            2            2 11.455096  1.83564318
## 3            3            1  9.448889 -0.17056384
## 4            4            1  6.567100 -3.05235257
## 5            5            2 11.080759  1.46130629
## 6            6            1  9.866935  0.24748237
## 7            7            1 11.066079  1.44662657
## 8            8            2  7.115263 -2.50418981
## 9            9            1  8.478822 -1.14063086
## 10          10            2  9.126518 -0.49293472
## 11          11            1 11.077905  1.45845198
## 12          12            2 11.238149  1.61869598
## 13          13            2  6.796180 -2.82327322
## 14          14            2  7.745382 -1.87407077
## 15          15            1  9.542996 -0.07645695
## 16          16            1  7.913203 -1.70624991
## 17          17            1  9.601031 -0.01842209
## 18          18            1 10.571534  0.95208145
## 19          19            2  9.679823  0.06037042
## 20          20            2 12.890654  3.27120122
```

Chapter 5

Exploratory and descriptive analysis with R

5.1 Working example - record sales data

Let's import the data {.smaller}

```
Album_Sales <- read_csv("Datasets/album_sales.csv")
```

```
## Parsed with column specification:
## cols(
##   Adverts = col_double(),
##   Sales = col_double(),
##   Airplay = col_double(),
##   Attract = col_double(),
##   Genre = col_character()
## )
```

Let's look at the data {.smaller}

```
head(Album_Sales)
```

```
## # A tibble: 6 x 5
##   Adverts Sales Airplay Attract Genre
##     <dbl> <dbl>   <dbl>   <dbl> <chr>
## 1    10.3   330      43      10 Country
## 2   986.    120      28       7 Pop
## 3  1446.    360      35       7 HipHop
## 4  1188.    270      33       7 HipHop
## 5   575.    220      44       5 Metal
## 6   569.    170      19       5 Country
```

5.2 Let's make sure our data types are correct #1

- This variable is currently stored as charcters, not as a factor / category variable

```
str(Album_Sales$Genre)
```

```
##  chr [1:200] "Country" "Pop" "HipHop" "HipHop" "Metal" "Country" "Pop" ...
```

- We can save it as a factor

```
Album_Sales$Genre <- as.factor(Album_Sales$Genre)

str(Album_Sales$Genre)
```

```
##  Factor w/ 4 levels "Country","HipHop",..: 1 4 2 2 3 1 4 4 3 2 ...
```

5.3 Measures of central tendency

The main measures of central tendency are: - Mean - Median - Mode

5.3.1 Mean

"What is the mean of album sales?"

```
mean(Album_Sales$Sales)
```

```
## [1] 193.2
```

5.3.2 Trimmed mean

- The trimmed mean is used to reduce the influence of outliers on the summary

```
mean(Album_Sales$Sales, trim = 0.05)
```

```
## [1] 192.6667
```

5.3.3 Median

"What is the median amount of Airplay?"

```
median(Album_Sales$Airplay)
```

```
## [1] 28
```

5.3.4 Mode

"What is the most common attractiveness rating of bands?"

- The easiest way to get the mode in R is to generate a frequency table

```
table(Album_Sales$Attract)
```

```
##
##  1  2  3  4  5  6  7  8  9 10
##  3  1  1  4 17 44 73 44 12  1
```

- We can then look for the most frequently occuring response

5.4 Measures of dispresion or variance

5.4.1 Range

The range is the difference between the lowest and highest values

- You can calculate it using these values

```
max(Album_Sales$Airplay) - min(Album_Sales$Airplay)
```

```
## [1] 63
```

- Or you can use the range command to get the min and max values in one go

```
range(Album_Sales$Airplay)
```

```
## [1]  0 63
```

5.4.2 Interquartile range

- We know that the median is the "middle" of the data = 50th percentile
- The interquatile range is the difference between the values at the 25th and 75th percentiles

```
quantile( x = Album_Sales$Airplay, probs = c(.25,.75) )
```

```
##   25%   75%
## 19.75 36.00
```

- Interquartile range = 36 - 19.75 = 16.25

Sum of squares {.smaller}

- The difference between each value and the mean value, squared, and then summed together

```
sum( (Album_Sales$Adverts - mean(Album_Sales$Adverts))^2 )
```

```
## [1] 46936335
```

5.4.3 Variance

- Variance: Sum of sqaures divided by n-1

```
# variance calculation
varianceAdverts <- sum( (Album_Sales$Adverts - mean(Album_Sales$Adverts))^2 ) / 199
```

5.4.4 Standard deviation

- Standard deviation is square root of the variance

```
# sd calculation


sqrt(varianceAdverts)
```

```
## [1] 485.6552
```

- Can be calculated using the sd() command

```
sd(Album_Sales$Adverts)
```

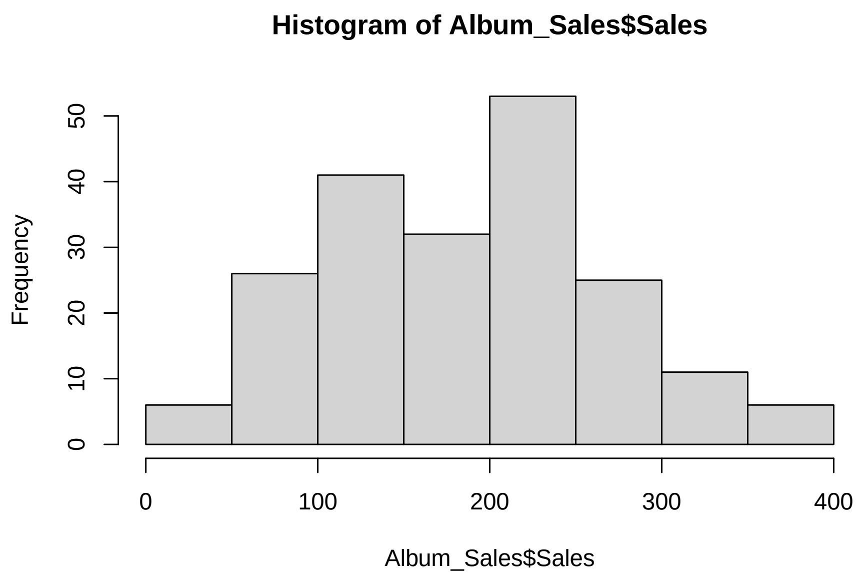
```
## [1] 485.6552
```

5.5 Skewness and Kurtosis

5.5.1 Assessing skewness of distribution #1

- It is possible to use graphs to view the distribution
- We will focus on graphic presentation of data next week

```
hist(Album_Sales$Sales)
```



5.5.2 Assessing skewness of distribution #2

- We can check raw skewness value using the *skew()* command in the **psych** package

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:car':
##
##     logit

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
skew(Album_Sales$Sales)
```

```
## [1] 0.0432729
```

5.5.3 Kurtosis

informal term	technical name	kurtosis value
"too flat"	platykurtic	negative

informal term	technical name	kurtosis value
"just pointy enough"	mesokurtic	zero
"too pointy"	leptokurtic	positive

```
kurtosi(Album_Sales$Sales)
```

```
## [1] -0.7157339
```

5.5.4 Assessing normality of distribution

- We can use the shapiro-wilk test of normality
- This is part of "base" r (no package needed)

```
shapiro.test(Album_Sales$Sales)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Album_Sales$Sales
## W = 0.98479, p-value = 0.02965
```

5.6 Getting and overall summary

5.6.1 summary() - in "base R"

```
summary(Album_Sales)
```

```
##     Adverts             Sales          Airplay         Attract
##  Min.   :   9.104   Min.   : 10.0   Min.   : 0.00   Min.   : 1.00
##  1st Qu.: 215.918   1st Qu.:137.5   1st Qu.:19.75   1st Qu.: 6.00
##  Median : 531.916   Median :200.0   Median :28.00   Median : 7.00
##  Mean   : 614.412   Mean   :193.2   Mean   :27.50   Mean   : 6.77
##  3rd Qu.: 911.226   3rd Qu.:250.0   3rd Qu.:36.00   3rd Qu.: 8.00
##  Max.   :2271.860   Max.   :360.0   Max.   :63.00   Max.   :10.00
##      Genre
##  Country:46
##  HipHop :53
##  Metal  :48
##  Pop    :53
##
##
```

5.6.2 describe() - in the "psych" package #1

```
describe(Album_Sales)
```

```
##         vars   n   mean     sd median trimmed    mad  min     max   range  skew
## Adverts    1 200 614.41 485.66 531.92  560.81 489.09  9.1 2271.86 2262.76  0.84
## Sales      2 200 193.20  80.70 200.00  192.69  88.96 10.0  360.00  350.00  0.04
## Airplay    3 200  27.50  12.27  28.00   27.46  11.86  0.0   63.00   63.00  0.06
## Attract    4 200   6.77   1.40   7.00    6.88   1.48  1.0   10.00    9.00 -1.27
## Genre*     5 200   2.54   1.12   3.00    2.55   1.48  1.0    4.00    3.00 -0.02
##         kurtosis    se
## Adverts     0.17 34.34
## Sales      -0.72  5.71
## Airplay    -0.09  0.87
## Attract     3.56  0.10
## Genre*     -1.37  0.08
```

5.6.3 describe() - in the "psych" package #2

- We can describe by factor variables

```
describeBy(Album_Sales, group = Album_Sales$Genre)
```

```
##
##  Descriptive statistics by group
## group: Country
##         vars  n   mean     sd median trimmed    mad  min     max   range  skew
## Adverts    1 46 656.22 507.96 574.14  620.40 581.96  9.1 1985.12 1976.01  0.51
## Sales      2 46 201.74  73.64 210.00  200.79  66.72 60.0  360.00  300.00  0.03
## Airplay    3 46  29.07  10.53  28.00   28.50  11.12  9.0   54.00   45.00  0.44
## Attract    4 46   6.52   1.63   7.00    6.71   1.48  1.0   10.00    9.00 -1.49
## Genre*     5 46   1.00   0.00   1.00    1.00   0.00  1.0    1.00    0.00   NaN
##         kurtosis    se
## Adverts    -0.65 74.89
## Sales      -0.52 10.86
## Airplay    -0.10  1.55
## Attract     3.54  0.24
## Genre*       NaN  0.00
## ------------------------------------------------------------
## group: HipHop
##         vars  n   mean     sd median trimmed    mad   min  max   range  skew
## Adverts    1 53 606.32 452.84 601.43  568.33 501.36 10.65 2000 1989.35  0.70
## Sales      2 53 199.62  92.71 200.00  200.70 103.78 10.00  360  350.00 -0.10
## Airplay    3 53  28.09  13.86  30.00   28.33  14.83  0.00   55   55.00 -0.14
## Attract    4 53   6.96   1.13   7.00    7.00   1.48  3.00    9    6.00 -0.80
## Genre*     5 53   2.00   0.00   2.00    2.00   0.00  2.00    2    0.00   NaN
##         kurtosis    se
## Adverts     0.05 62.20
## Sales      -0.91 12.74
## Airplay    -0.83  1.90
## Attract     2.03  0.15
## Genre*       NaN  0.00
## ------------------------------------------------------------
## group: Metal
##          vars  n   mean     sd median trimmed    mad  min     max   range  skew
## Adverts     1 48 693.45 534.06  593.0  640.19 521.34 45.3 2271.86 2226.56  0.92
## Sales       2 48 197.71  75.18  200.0  198.25  88.96 40.0  340.00  300.00 -0.07
## Airplay     3 48  27.96  11.37   27.5   28.00  11.12  2.0   57.00   55.00  0.02
## Attract     4 48   6.85   1.34    7.0    6.90   1.48  2.0    9.00    7.00 -0.84
## Genre*      5 48   3.00   0.00    3.0    3.00   0.00  3.0    3.00    0.00   NaN
##          kurtosis    se
## Adverts      0.21 77.08
## Sales       -0.94 10.85
## Airplay     -0.26  1.64
## Attract      1.74  0.19
## Genre*        NaN  0.00
## ------------------------------------------------------------
## group: Pop
##          vars  n   mean     sd median trimmed    mad   min     max   range  skew
## Adverts     1 53 514.63 446.04  429.5  453.85 438.01 15.31 1789.66 1774.35  1.01
## Sales       2 53 175.28  77.92  160.0  171.86  88.96 40.00  360.00  320.00  0.34
## Airplay     3 53  25.13  12.75   26.0   25.02  11.86  1.00   63.00   62.00  0.25
## Attract     4 53   6.72   1.47    7.0    6.81   1.48  1.00    9.00    8.00 -1.11
## Genre*      5 53   4.00   0.00    4.0    4.00   0.00  4.00    4.00    0.00   NaN
##          kurtosis    se
## Adverts      0.27 61.27
## Sales       -0.67 10.70
## Airplay      0.46  1.75
## Attract      2.51  0.20
## Genre*        NaN  0.00
```

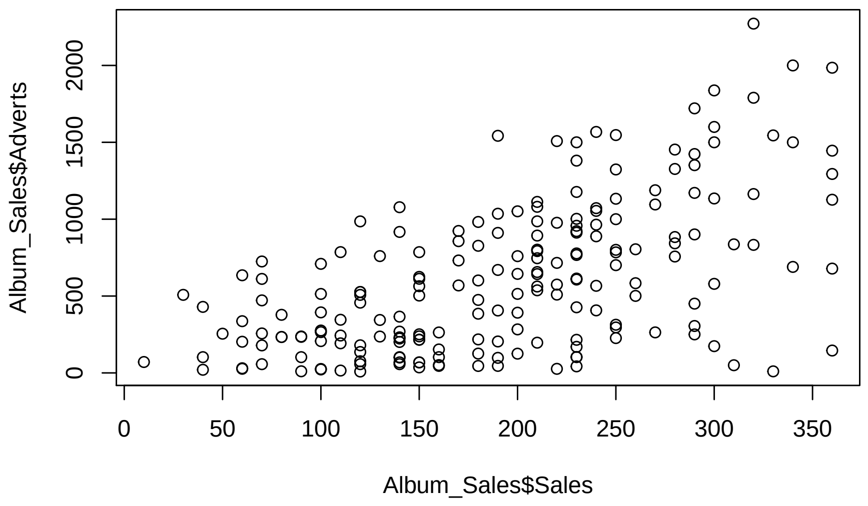
5.7 Basic statistical tests (more detail in later sections)

5.7.1 Corrleation

"Is there a relationship between advert spend and sales?"

- We would use an correlational analysis to answer this question

```
plot(Album_Sales$Sales,Album_Sales$Adverts)
```



"Is there a relationship between advert spend and sales?"

- We would use an correlational analysis to answer this question

```
cor.test(Album_Sales$Sales,Album_Sales$Adverts)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Album_Sales$Sales and Album_Sales$Adverts
## t = 9.9793, df = 198, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4781207 0.6639409
## sample estimates:
##       cor
## 0.5784877
```

5.7.2 Tests of difference - t-test

"Is there a significant difference in sales between the Country and Hip-hop musical genres?"

- We would use a t-test to answer this question

```
myTtestData <- Album_Sales %>% filter(Genre == c("Country", "HipHop"))
```

```
t.test(myTtestData$Sales ~ myTtestData$Genre)
```

```
##
##  Welch Two Sample t-test
##
## data:  myTtestData$Sales by myTtestData$Genre
## t = 0.80489, df = 40.62, p-value = 0.4256
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -27.80146  64.62904
## sample estimates:
## mean in group Country  mean in group HipHop
##              216.0000              197.5862
```

5.7.3 Tests of difference - ANOVA

"Is there a significant difference in sales between all musical genres?"

- We would use an ANOVA to answer this question

```
myAnova <- aov(Album_Sales$Sales ~ Album_Sales$Genre)
summary(myAnova)
```

```
##                    Df  Sum Sq Mean Sq F value Pr(>F)
## Album_Sales$Genre   3   23530    7843   1.208  0.308
## Residuals         196 1272422    6492
```

Chapter 6

Graphing and data visualisation with R

6.1 Presenting data visually

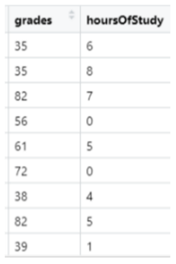
6.2 By the end of this section, you will be able to:

- Describe the ggplot "grammar of visualisation": coordinates and geoms
- Write a graph function to display multiple variables on a plot
- Amend the titles and legends of a plot
- Save plots in PDF or image formats

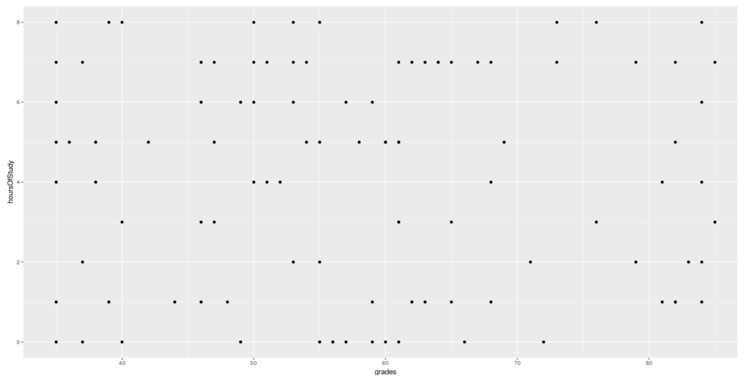
6.3 The "grammar of visualisation"

- Graphs are made up of 3 components:
  - A dataset
  - A coordinate system
  - Visual marks to represent data **(geoms)**

The "grammar of visualisation" #2



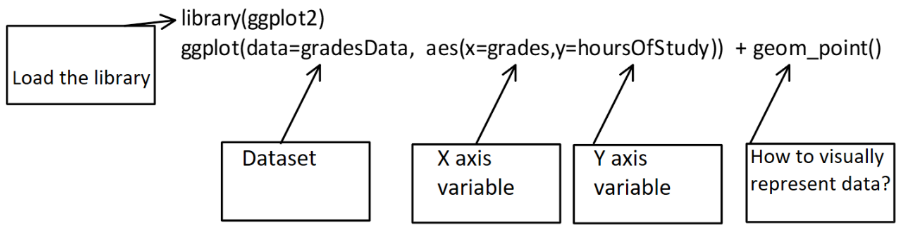
grades	hoursOfStudy
35	6
35	8
82	7
56	0
61	5
72	0
38	4
82	5
39	1



- In the above example, the dataset is the *studentData* that we used previously.
- The *grades* variable is mapped to the X axis
- The *hoursOfStudy* variable is mapped to the Y axis

6.4 How to code a graph

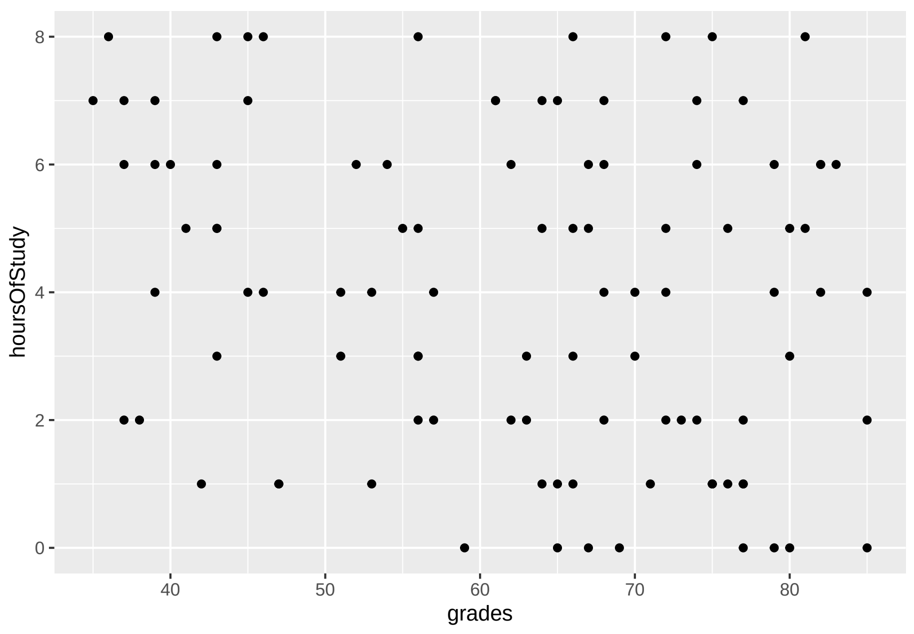
- The graph is created using the following code:


library(ggplot2)
ggplot(data=gradesData, aes(x=grades,y=hoursOfStudy)) + geom_point()
Load the library
Dataset
X axis variable
Y axis variable
How to visually represent data?

- In this code, we specify the dataset, the variables for the X and Y axes and the **geom** that will represent the data points visually (in this case, each datum is a point)

6.5 The graph output

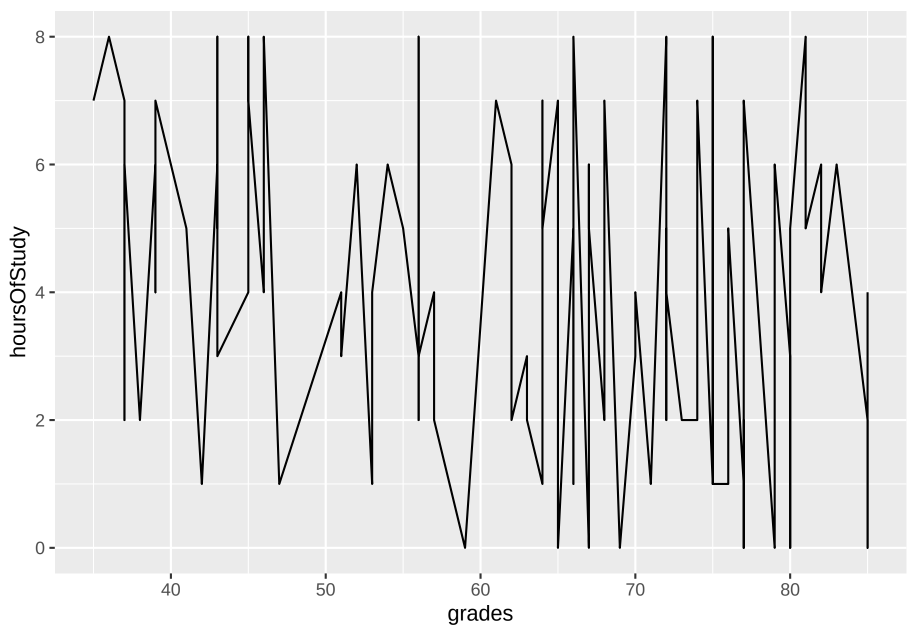
```
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_point()
```



6.6 Changing the geoms leads to different visualisations

- If we change from points to lines, for example we get a different plot:

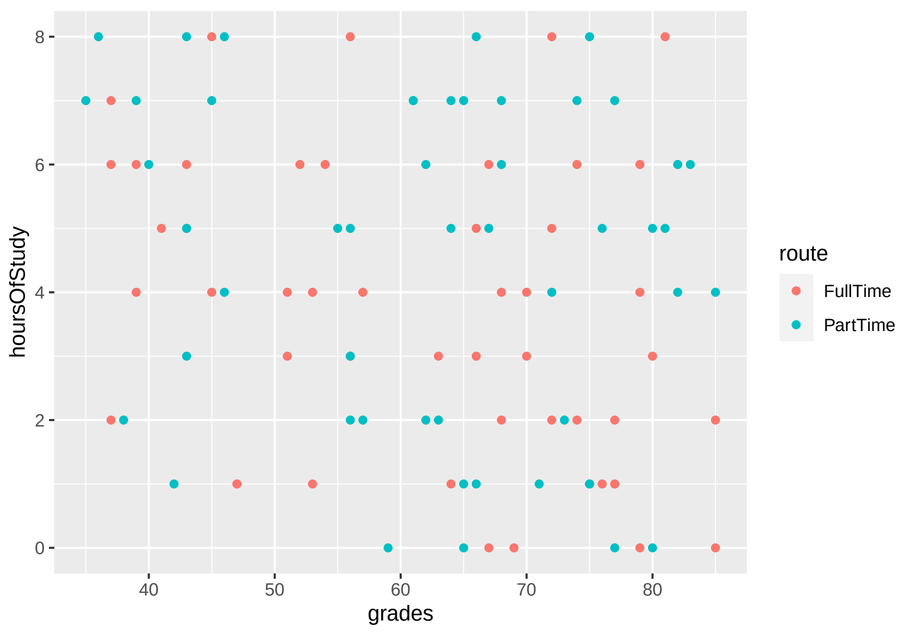
```
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_line()
```



6.7 It is possible to represent more variables on the plot

- By specifying that colours of our points should be attached to the **route** variable, the data is now colour-coded

```
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_point(aes(color = route)
```

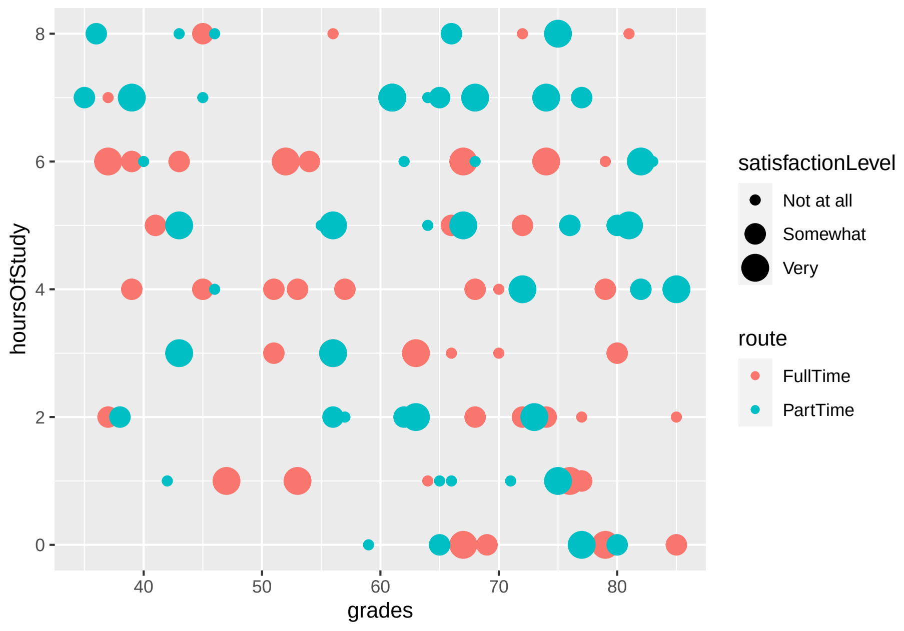


6.8 It is possible to represent more variables on the plot #2

- By specifying that size of our points should be attached to the **satisfactionLevel** variable, the size of the points adjusts

```
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_point(aes(color = route, size=satis
```

```
## Warning: Using size for a discrete variable is not advised.
```

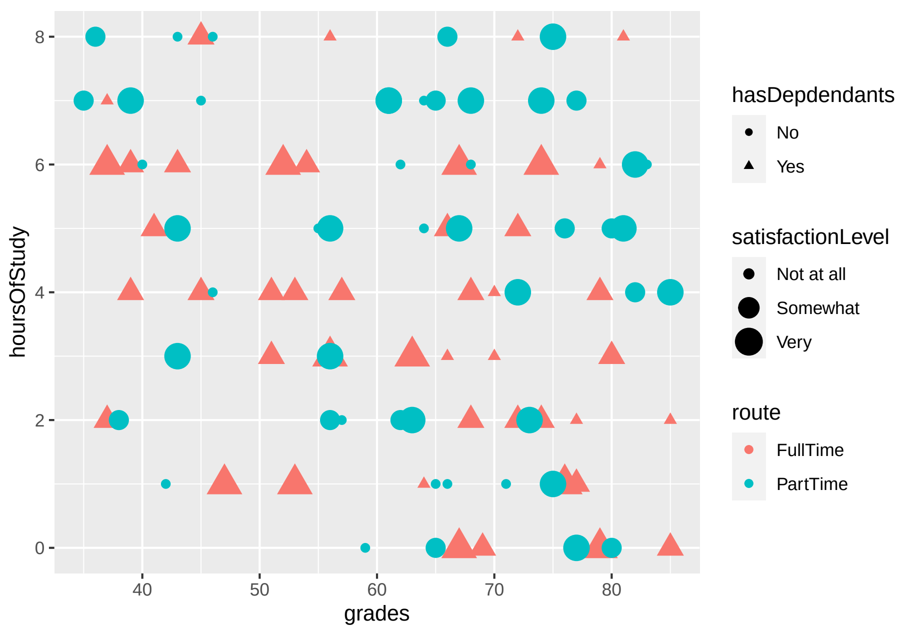


6.9 It is possible to represent more variables on the plot #3

- By specifying that shape of our points should be attached to the **hasDependents** variable, the shape of the points changes accordingly

```
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_point(aes(color = route,
```

```
## Warning: Using size for a discrete variable is not advised.
```

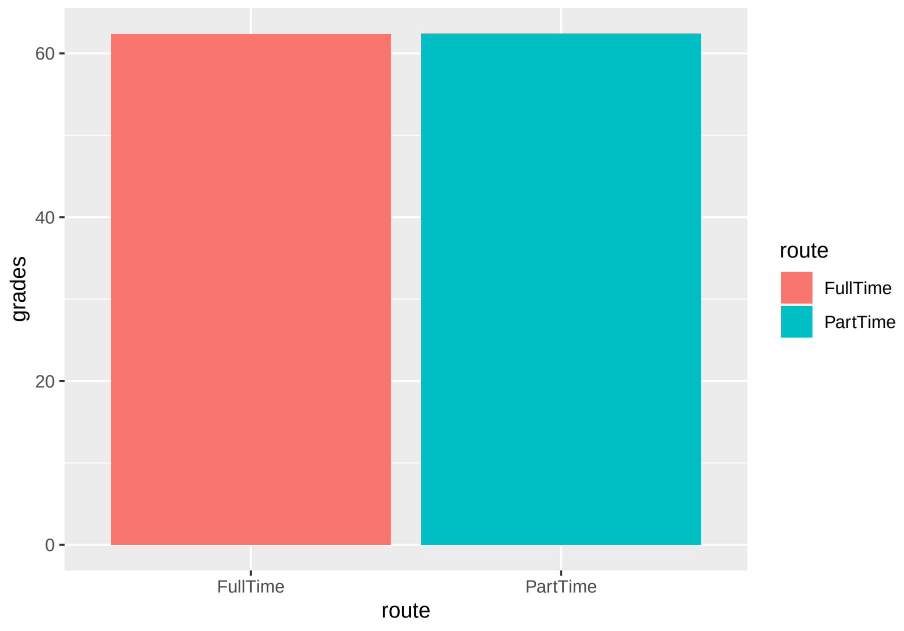


6.10 Plotting summaries of data

- We can summarise the data (e.g. get the mean or sd) using the *stat_summary()* function
- Below we are making a bar chart with the mean grade for each route

```
ggplot(data=studentData, aes(x=route, y= grades, fill=route)) + stat_summary(fun.y = "mean", geom
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

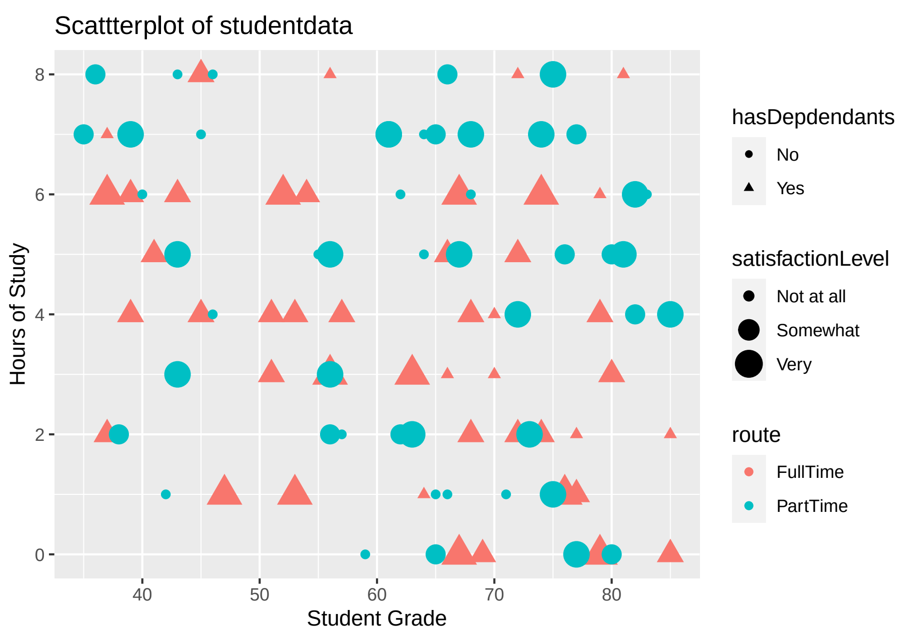


6.11 Changing the axis labels and title on a plot

We can change the axis labels and title using the **labs()** command:

```
labs(x="Student Grade", y="Hours of Study", title = "Scattterplot of student data")
library(ggplot2)

ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) + geom_point(aes(color = route,
```

```
## Warning: Using size for a discrete variable is not advised.
```



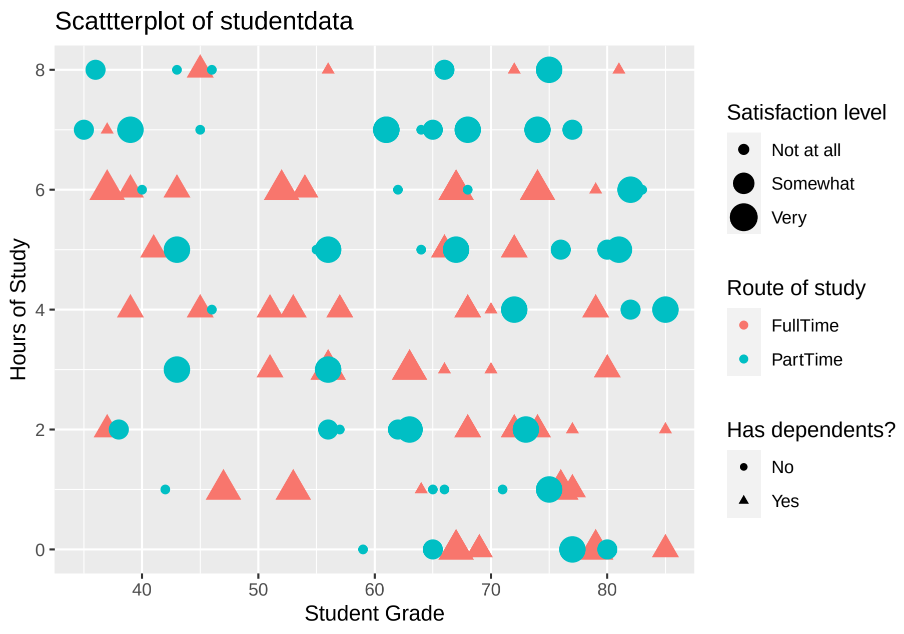
6.12 Changing the legend on a plot

To change the legend, we use the **labs()** command too, and reference the relevant property (e.g. size, shape, colour)

```
labs(x="Student Grade", y="Hours of Study", title = "Scattterplot of student data", color="Route
```

```r
library(ggplot2)

  ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) +
    geom_point(aes(color = route, size=satisfactionLevel, shape=hasDepdendants)) +
  labs(x="Student Grade", y="Hours of Study", title = "Scattterplot of studentdata", color="Route
```

```
## Warning: Using size for a discrete variable is not advised.
```



6.13 Storing plots to be recalled later

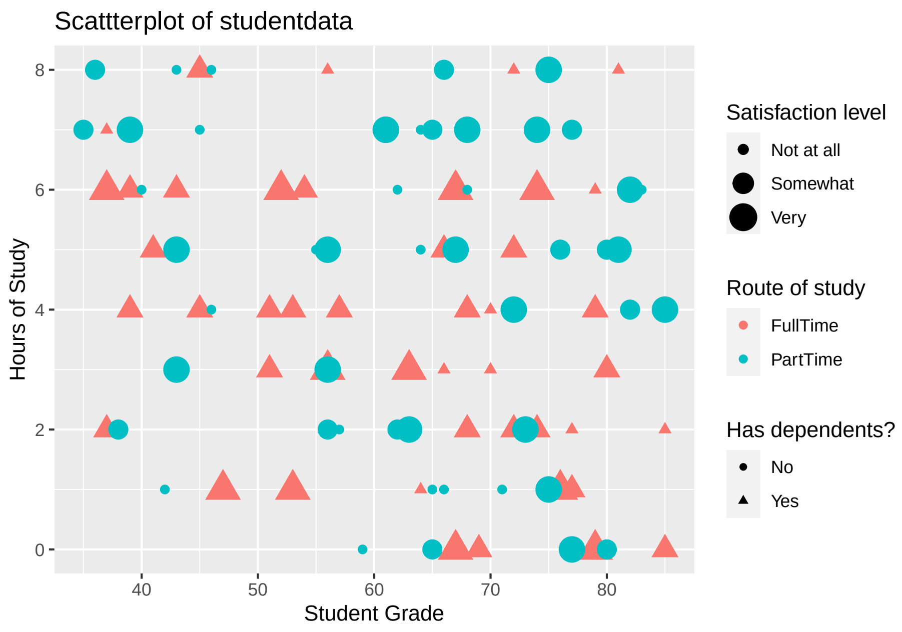
- Plots can be assigned to objects in R and recalled later, just like any other piece of data

```
library(ggplot2)

## Create plot and store it as "myPlot" object

myPlot <- ggplot(data=studentData, aes(x=grades,y=hoursOfStudy)) +
  geom_point(aes(color = route, size=satisfactionLevel, shape=hasDepdendants)) +
  labs(x="Student Grade", y="Hours of Study", title = "Scattterplot of studentdata", c
```

6.14 Recalling a stored plot

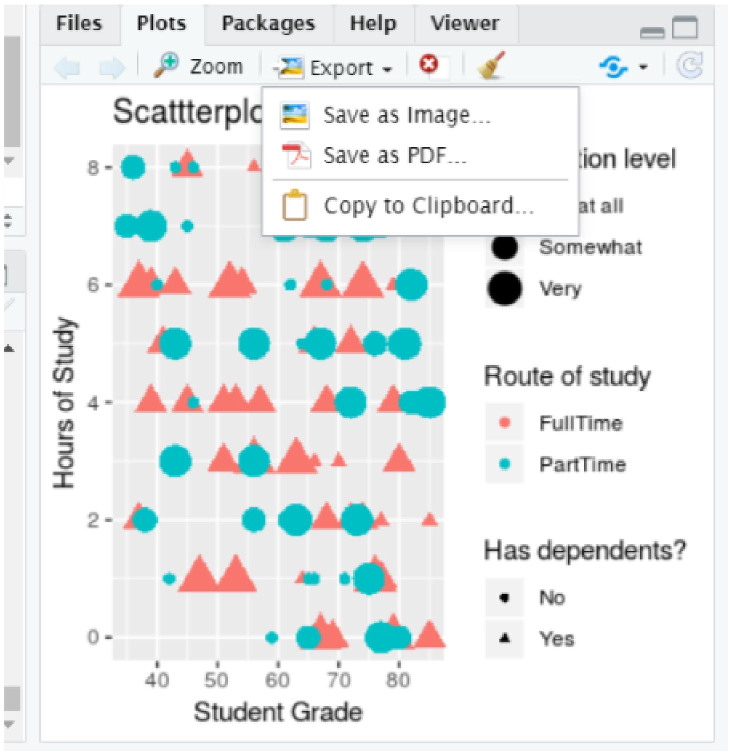
```
 #Recall myPlot
myPlot
```

```
## Warning: Using size for a discrete variable is not advised.
```



6.15 Saving plots # 1

- Plots can be save using the **export** button in the plots tab



6.16 Plots can also be saved using code

- You might want to include code to save your plot in a script, for example
- This can allow greater control over the output file and plot dimensions:

```r
ggsave(plot= myPlot, file="myPlot.pdf", width = 4, height = 4)
```

```
## Warning: Using size for a discrete variable is not advised.
```

```r
ggsave(plot= myPlot, file="myPlot.png", width = 4, height = 4, units="cm", dpi=320)
```

```
## Warning: Using size for a discrete variable is not advised.
```