**Learning Journal weeks 8-13**

**Ellen Kirkpatrick**

**Week 8**

R Studio introduction

Copied and pasted the project management and exporting frames episodes from week 7 journal. All other episodes attempted for the first time in this journal.

Project Management of rstudio

1. Create new project. Select file > new project > new directory > (empty project did not appear) > selected R project instead. Named file r-geospatial. Clicked create project.
2. Entered getwd() into r console. Responded with the user directory of the file.
3. In lower lefthand panel, created 3 new folders: data, doc and results.
4. Saved 3 datasets from online: nordic-data.csv, nordic-data-2.csv, gapminder\_data.csv into new data directory. Found this folder using the response from R studio, went to documents, found the geospatial directory and then the new data directory within that. 3 csv files are now there.
5. Downloaded the zip folder of data from data carpentry. Move downloaded zip file to data directory. Unzip all files.
6. Files all appear in data directory.

Data structures

1. Download and read file: nordic-data.csv. Save as an object named nordic. Inputted nordic < - read.csv(”data/noridc-data.csv”). Ran in console. Recieved error message.
2. Opened new file nordic.data.csv. New tab appears in top left corner, but don’t know where to input the message.
3. Tried to move on, but I can’t type in the console, the mouse won’t click.
4. Made console full page. Then it worked. Typed in nordic <- read.csv(”data/nordic-data.csv”) . Successful. Output appeared in top right corner.
5. Entered nordic$country. Returned with output of the countries.
6. Entered nordic$lifeExp. Output appeared with 3 numbers.
7. Entered nordic$lifeExp + 2. Output added 2 to each of the numbers
8. Entered nordic$lifeExp + nordic$country. Received error message and NA NA NA. Command not meaningful in R. Not meaningul as combines life expectancy and country type.
9. Entered class(nordic$lifeExp). Response was [1] “numeric”.
10. Entered class(3.14) . Response numeric.
11. Entered class(1L). Response integer
12. Entered class(1+1i). Response complex
13. Entered class(TRUE). Response logical
14. Entered class (’banana’). Response character.
15. Entered class(factor(’banana’)). Response factor.
16. How to understand this: R interprets a specific data class.
17. Load file: open file > selected nordic-data-2.csv.
18. Entered: nordic\_2 <- read.csv("data/nordic-data-2.csv") and class(nordic\_2$lifeExp). Response was factor.
19. Entered nordic\_2$lifeExp + 2. Received NA NA NA and error message.
20. Entered class(nordic). Responded with data frame.
21. Start vectors and type coercion.
22. Entered my\_vector <- vector(length = 3), my\_vector . Received FALSE FALSE FALSE.
23. Entered another\_vector <- vector(mode = 'character', length = 3) and another\_vector. Received [1] "" "" "".
24. Entered str(another\_vector). Received chr [1:3] "" "" "".
25. Entered str(nordic$lifeExp). Received num [1:3] 77.2 80 79.
26. Discussion 1 - Why is R so opinionated about what we put in our columns of data? R is trying to keep the data clean and consistent so that each column has one type.
27. Entered combine\_vector <- c(2, 6, 3), combine\_vector. Received [1] 2 6 3.
28. Entered quiz\_vector <- c(2, 6, '3'). Did not receive direct output response but a new loutput appeared in the top right corner under values.
29. Entered coercion\_vector <- c('a', TRUE), coercion\_vector. Received [1] "a" "TRUE".
30. Entered another\_coercion\_vector <- c(0, TRUE), another\_coercion\_vector. Received [1] 0 1.
31. Entered character\_vector\_example <- c('0', '2', '4'), character\_vector\_example. Received [1] "0" "2" "4".
32. Entered character\_coerced\_to\_numeric <- as.numeric(character\_vector\_example)
33. character\_coerced\_to\_numeric. Received [1] 0 2 4.
34. Entered numeric\_coerced\_to\_logical <- as.logical(character\_coerced\_to\_numeric)
35. numeric\_coerced\_to\_logical. Received FALSE TRUE TRUE.
36. Challenge 1 - why is lifeExp different between nordic and nordic 2.
37. Entered str(nordic\_2$lifeExp). Received Factor w/ 3 levels "77.2","79.0 or 83",..: 1 3 2. Data is stored as factors.
38. Ran str(nordic$lifeExp). Received num [1:3] 77.2 80 79. Stored as numbers,
39. Using combine function. Entered ab\_vector <- c('a', 'b'), ab\_vector. Received [1] "a" "b"
40. Entered combine\_example <- c(ab\_vector, 'DC'), combine\_example. Received [1] "a" "b" "DC".
41. Make series of numbers. Entered my\_series <- 1:10. Output appeared in top right panel. Not in console.
42. **Error:** Realised read the instructions wrong. Re-entered my\_series <- 1:10, my\_series. Received output in console [1] 1 2 3 4 5 6 7 8 9 10. Successful.
43. Entered seq(10). Same output received.
44. Entered seq(1,10, by = 0.1). Received 6 rows of output with multiple numbers in each row.
45. Ask questions about vectors. Entered sequence\_example <- seq(10), head(sequence\_example,n = 2). Received [1] 1 2.
46. Entered tail(sequence\_example, n = 4). Received [1] 7 8 9 10.
47. Entered length(sequence\_example). Received [1] 10.
48. Entered class(sequence\_example). Received [1] "integer".
49. Give names to elements in vector. Ran my\_example <- 5:8, names(my\_example) <- c("a", "b", "c", "d"), my\_example. Received a b c d 5 6 7 8
50. Challenge 2 - make vector with numbers 1-26. Multiply vector by 2 and give resulting names A to Z. Used solution on data carpentry to work through it. Would not have got there otherwise.
51. Entered x <- 1:26, x <- x \* 2, names(x) <- LETTERS. No response in consolse but output in top right corner. New row of data.
52. Factors.
53. Entered str(nordic$lifeExp). Received num [1:3] 77.2 80 79.
54. Entered str(nordic$year). Received int [1:3] 2002 2002 2002.
55. Entered str(nordic$country). Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2.
56. Make vector labelling nordic countries for all in study. Entered nordic\_countries <- ('Norway', 'Finland', 'Denmark', 'Iceland', 'Sweden'), nordic\_countries
57. Received [1] "Norway" "Finland" "Denmark" "Iceland" "Sweden" .
58. Entered str(nordic\_countries). Received chr [1:5] "Norway" "Finland" "Denmark" "Iceland" "Sweden".
59. Turn vector into factor. Entered categories <- factor(nordic\_countries), class(categories). Received [1] "factor".
60. Entered str(categories). Recieved Factor w/ 5 levels "Denmark","Finland",..: 4 2 1 3 5.
61. Entered class(nordic\_countries). Received [1] "character".
62. Entered class(categories). Received [1] "factor".
63. Challenge - why are these numbers used to represent these countries? Alphabetical order.
64. Challenge - is there a factor in nordic data frame? Did not understand the questino or what to do. Used solution to copy over the responses and see how they got there. Entered nordic <- read.csv(file = "data/nordic-data.csv", stringsAsFactors = FALSE), str(nordic$country). Received chr [1:3] "Denmark" "Sweden" "Norway".
65. Used colclasses solution. Entered nordic <- read.csv(file="data/nordic-data.csv", colClasses=c(NA, NA, "character")), str(nordic$country). Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2.
66. Know where baseline levels are. Entered mydata <- c("case", "control", "control", "case"), factor\_ordering\_example <- factor(mydata, levels = c("control", "case")), str(factor\_ordering\_example). Received Factor w/ 2 levels "control","case": 2 1 1 2.
67. Lists.
68. Entered list\_example <- list(1, "a", TRUE, c(2, 6, 7)) list\_example. Received successful output, the data all listed by numbers between brackets.
69. Entered another\_list <- list(title = "Numbers", numbers = 1:10, data = TRUE ), another\_list. Received right output again.
70. Entered str(nordic). Received 'data.frame': 3 obs. of 3 variables: $ country: Factor w/ 3 levels "Denmark","Norway",..: 1 3 2 $ year : int 2002 2002 2002 $ lifeExp: num 77.2 80 79.
71. Entered str(another\_list). Received List of 3 $ title : chr "Numbers" $ numbers: int [1:10] 1 2 3 4 5 6 7 8 9 10 $ data : logi TRUE.
72. Entered nordic$country. Received [1] Denmark Sweden Norway Levels: Denmark Norway Sweden
73. Entered nordic[, 1]. Received [1] Denmark Sweden Norway Levels: Denmark Norway Sweden.
74. Entered class(nordic[, 1]). Received [1] "factor".
75. Entered str(nordic[, 1]) Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2
76. Entered nordic[1, ]. Recieved country year lifeExp 1 Denmark 2002 77.2
77. Entered class(nordic[1, ]). Received [1] "data.frame"
78. Entered str(nordic[1, ]). Received 'data.frame': 1 obs. of 3 variables: $ country: Factor w/ 3 levels "Denmark","Norway",..: 1 $ year : int 2002 $ lifeExp: num 77.2

Reflection on R studio

I feel I am just copying the commands in from the data carpentry lessons, although getting the correct responses, I have no real understanding of what I am doing or why.

I am unsure at this stage how it can help me in research or what I can do with it.

Exploring data frames - data carpentry episode

1. Imported gapminder dataset from bottom left panel. R studio had to update, new console popped up and it took time to load and get through. Was installing packages.
2. Entered str(gapminder). received error message
3. entered class (gapminder$year). received error message
4. Re-visited data carpentry page. Copied and pasted the command from miscallaneous tips about getting the data.
5. re-entered str(gapminder). Output appears. Successful.
6. Entered class(gapminder$year). Response integer.
7. Entered class(gapminder$country). Response factor.
8. Entered str(gapminder$country). Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
9. Entered length(gapminder). Response 6
10. Entered class(gapminder). Response data frame.
11. Get number of rows. Entered nrow(gapminder). Got 1704.
12. Number of columns. Entered ncol(gapminder). Got 6.
13. Rows and columns together . Entered dim(gapminder). Response 1704 6.
14. Titles of columns. Entered colnames(gapminder). Response [1] "country" "year" "pop" "continent" "lifeExp" "gdpPercap".
15. Entered head(gapminder). Recieved list of Afghanistan in different years with multiple columns.
16. Challenge 1 - check the last lines of data. Enter tail(gapminder), tail(gapminder, n = 15). Received list of Zambia and Zimbabwe, the final 15 lines of data.
17. Entered gapminder[sample(nrow(gapminder), 5), ]. Abritrary lines of data.
18. Adding columns and rows in data frames.
19. Create new column for whether life expectancy is below world average. Entered below\_average <- gapminder$lifeExp < 70.5 head(gapminder). Successful.
20. Add this as a column. Entered cbind(gapminder, below\_average). New column appears in response.
21. Want first 6 lines of output. Entered head(cbind(gapminder, below\_average)). First 6 lines of afghanistan data appears.
22. Tried to add vector of below average. Entered below\_average <- c(TRUE, TRUE, TRUE, TRUE, TRUE) head(cbind(gapminder, below\_average)). Received error message.
23. Factors.
24. Entered levels(gapminder$continent). Response [1] "Africa" "Americas" "Asia" "Europe" "Oceania" .
25. Entered levels(gapminder$continent) <- c(levels(gapminder$continent), "Nordic")
26. gapminder\_norway <- rbind(gapminder, list('Norway', 2016, 5000000, 'Nordic', 80.3,49400.0, FALSE)), tail(gapminder\_norway). Received output. successful.
27. Change factor into a character vector. Enter str(gapminder) and gapminder$continent <- as.character(gapminder$continent), str(gapminder). Successful.
28. Appending to a data frame.
29. Glue two data frames together. Entered gapminder <- rbind(gapminder, gapminder), tail(gapminder, n=3). Successful.
30. Remove rownames. Entered rownames(gapminder) <- NULL, head(gapminder). Successful.
31. Challenge 3 - create new data frame within R. Copied commands from the challenge, successful, new panel appeared in top right.

Subsetting data

1. Accessing elements using their indices.
2. Entered x[1]. Got A, 2. Different to a, 5.4
3. Entered x[4]. Got D, 8. Different to d, 4.8 Think responses still okay, command is getting the proper answer from the data.
4. Get multiple elements at once. Entered x[c(1, 3)].
5. Slices of vector. Entered x[1:4]. Getting correct format of output.
6. Entered 1:4. Response [1] 1 2 3 4
7. Entered c(1, 2, 3, 4). Same response.
8. Ask for same elementm, multiple times. Entered x[c(1, 1, 3)].
9. Entered x[0]. Response named numeric(0).
10. Skipping and removing elements
11. Entered x[-2]. All elements except one specified.
12. Skip multiple elements. Entered x[c(-1, -5)].
13. Challenge 1 - used solution to work backwards. Did not understand by myself.
14. Subsetting by name
15. extract elements by name. Entered x <- c(a = 5.4, b = 6.2, c = 7.1, d = 4.8, e = 7.5), x[c("a", "c")]. Received a c, 5.4 7.1 . Successful.
16. Subsetting through other logical operations. Entered x[c(FALSE, FALSE, TRUE, FALSE, TRUE)]. Received c e , 7.1 7.5.
17. Entered x[names(x) == "a"]. Received a , 5.4 .
18. Challenge 2 - worked through solutions as could not do by myself.
19. Data frames.
20. Entered head(gapminder[3]). Got data f rame in output.
21. Entered head(gapminder[["lifeExp"]]). Got [1] 28.801 30.332 31.997 34.020 36.088 38.438.
22. Extract columns by name. Entered head(gapminder$year). Got [1] 1952 1957 1962 1967 1972 1977.
23. Select specific rows/columns. Entered gapminder[1:3, ]. Specific rows and columns received.
24. Entered gapminder[3, ]. Received country year pop continent lifeExp gdpPercap, 3 Afghanistan 1962 10267083 Asia 31.997 853.1007.
25. Challenge - gapminder[1:20] returns error because undefined columns selected. gapminder[1:20, ] subsets data to first 20 rows.
26. Create new data frame called gapminder\_small. Entered gapminder\_small <- gapminder[c(1:9, 19:23),]. New data frame appears in top right panel.

Data frame manipulation with dplyr

1. Entered mean(gapminder[gapminder$continent == "Africa", "gdpPercap"]). Received [1] 2193.755.
2. Entered mean(gapminder[gapminder$continent == "Americas", "gdpPercap"]). Received [1] 7136.11.
3. Entered mean(gapminder[gapminder$continent == "Asia", "gdpPercap"]). Received [1] 7902.15.
4. DPLYR package.
5. Install package. Entered install.packages('dplyr'). Pop up screens. Successful.
6. Load package. library("dplyr"). Successful.
7. Keep variables you select. year\_country\_gdp <- select(gapminder, year, country, gdpPercap).
8. Repeat with year\_country\_gdp <- gapminder %>% select(year,country,gdpPercap).
9. Filters. copied and pasted long command. filtered only with european countries.
10. Challenge - worked backwards from solution to produce data frame for african values of lifeexp, country and year. Used year\_country\_lifeExp\_Africa <- gapminder %>%, filter(continent=="Africa") %>%, select(year,country,lifeExp). New frames.
11. Enter str(gapminder).
12. Entered gapminder %>% group\_by(continent) %>% str(). Successful.
13. Using summarise. Copy and pasted long command. data frame split into multiple pieces.
14. Challenge - worked backwards from solutions. copied and pasted long commands to see effect. noticed min and max to get shortest and highest life expectancies.
15. Function. Copied and pasted command. Groups multiple variables.
16. count function. copied and pasted command. successful.
17. calculate minimum, maximum, mean and se of each continent life expectancy. Copied and pasted command. Successful.
18. Mutate. Copied and pasted command. created new variables prior.

Introduction to visualisation

1. Use ggplot function. Entered library("ggplot2"), ggplot(data = gapminder, aes(x = lifeExp)) + geom\_histogram(). Received **error**: Error in ggplot(data = gapminder, aes(x = lifeExp)) : could not find function "ggplot"
2. Repeated with other commands. kept receiving error message. ggplot not found.
3. **Problem:** Had copied and pasted the commands...not sure why error being received. Moved onto next episode for now. Can’t work out why after repeatedly trying different commands. None recognised.
4. **Solution** - Googled ggplot2 problems. Realised I had to install the package first before working with it. Got the code from this website: https://www.dummies.com/programming/r/how-to-install-and-load-ggplot2-in-r/
5. Entered install.packages("ggplot2"). Successful.
6. Re-entered library("ggplot2"), ggplot(data = gapminder, aes(x = lifeExp)) + , geom\_histogram()
7. Successful!! Plot appears in bottom left panel.
8. Entered ggplot(data = gapminder, aes(x = lifeExp)). Not enough to draw the plot.
9. Entered ggplot(data = gapminder, aes(x = lifeExp)) + , geom\_histogram(). Received `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`. and plot.
10. Challenge - modify command so plot shows distributino of gdp per capita, rather than life expectancy. Entered ggplot(data = gapminder, aes(x = gdpPercap)) + , geom\_histogram(). Successful. Plot created for gdp per capita.
11. Histogram. Look at data from most recent year and only from Americas. Entered gapminder\_small <- filter(gapminder, year == 2007, continent == "Americas") and ggplot(data = gapminder\_small, aes(x = country, y = gdpPercap)) + geom\_col(). Successful. Plot appears.
12. Use coord flip function to swap axes. Entered ggplot(data = gapminder\_small, aes(x = country, y = gdpPercap)) + , geom\_col() +, coord\_flip(). Countries swapped to other axis. Successful.
13. Challenge - create a new bar plot showing the gdp per capita of all countries in the americas for 1952-2007, colour coded by year. Used solution and copied over commands to watch changes. Colours were changed.

Writing data

1. Saving plots.
2. Copied and pasted command from episode. Then entered dev.off(). Received RStudioGD, 2.
3. Challenge - copied and pasted commands from challenge solution and turned off the pdf device after.
4. Writing data.
5. Copied and pasted command from exercise. Error- received message: Error in file(file, ifelse(append, "a", "w")) : cannot open the connectionIn addition: Warning message:In file(file, ifelse(append, "a", "w")) : cannot open file 'cleaned-data/gapminder-aus.csv': No such file or directory
6. Re-attempted but received same message. Don’t understand the error message.

Data carpentry - R

Looking at the link provided on week 8 homework, realised I did the wrong data carpentry lessons. Re-started the epsiodes

1. Open r studio. File > Create new project > new directory > new project.
2. Create file for scripts. File > new file > r script. Click save icon. Saved “script.R”
3. Entered dir.create("data"), dir.create("data\_output"), dir.create("fig\_output") . Created new directories in bottom right panel.
4. Download dataset. Copied and pasted: download.file("https://ndownloader.figshare.com/files/11492171" "data/SAFI\_clean.csv", mode = "wb"). Successful. Dataset downloaded.
5. Exercise - install tidyverse package. Clicked on packages tab in bottom right panel, inputted tidyverse to textbox. Multiple minutes to install and a lot of output in the bottom console. Successful

Introduction to R

1. Tested math. typed 3 + 5 and 12 / 7. Output displayed correct numbers.
2. Typed area\_hectares < - 1.0, (area\_hectares < - 1.0). Output 1.
3. Typed area\_hectares. Output 1.
4. Artithmetic. Convert to acres. Enter 2.47 \* area\_hectares. Output 2.47
5. Change object’s value, enter area\_hectares <- 2.5, 2.47 \* area\_hectares. Output 6.175
6. exercise - what is current content of the object. No change, area still 6.175. Not re-run the line.
7. Exercise - create length and width variables and assign values. Entered length <- 2.5, width <- 3.2, area <- length \* width area. Output 8. successful.
8. Functions. Entered b <- sqrt(a). Error. Object not found.
9. Function with multiple arguments. Output 3.
10. Entered args(round). Successful. function (x, digits = 0) NULL
11. Different number of digits. Enter round(3.14159, digits = 2). Output 3.14.
12. Switch the order of arguments. Enter round(digits = 2, x = 3.14159). Output 3.14.
13. Exercise - type ?round and look at output. ceiling, floor, trunc, round, signif.
14. Vectors and data types. Ran hh\_members <- c(3, 7, 10, 6)
15. hh\_members. Output successful.
16. Vector containing characters. Enter respondent\_wall\_type <- c("muddaub", "burntbricks", "sunbricks") respondent\_wall\_type. Output with characters. Successful.
17. Inspect content of vector. Enter length(hh\_members). Output 4.
18. Enter length(respondent\_wall\_type\_. Output 3.
19. Class function. Enter class(hh\_members). Output numeric. Successful.
20. Enter class(respondent\_wall\_type). Output character. Successful.
21. Overview of object structure and elements. Enter str(hh\_members). Output successful.
22. Enter str(respondent\_wall\_type). Output successful.
23. Add other elements to vector. Use C function. Copied and pasted command. Successful.
24. Exercise - what happens when you mix types in a single vector. R converts them to be the same type.
25. Copied and pasted multi line command from exercise. R finds common denominator to convert content.
26. How many values in combined logical are true - 1.
27. Extract values from a vector through indices/square brackets. Copied and pasted command. Output successful.
28. Entered: respondent\_wall\_type[c(3, 2)]. Output successful.
29. Repeat incides to create object with more elements. Copied and pasted long command. Successful.
30. Conditional subsetting. Copied and pasted long command. Successful.
31. Wanted to select only particular variables. Entered hh\_members > 5. successful.
32. Entered hh\_members[hh\_members > 5]. Successful.
33. Combine multiple tests. Entered hh\_members[hh\_members < 3 | hh\_members > 5]. Successful.
34. Entered hh\_members[hh\_members >= 7 & hh\_members == 3].
35. Search for strings in vector. Copied and pasted long command. successful.
36. Entered possessions %in% c("car", "bicycle", "motorcycle", "truck", "boat"). Successful.
37. Entered possessions[possessions %in% c("car", "bicycle", "motorcycle", "truck", "boat")]. Successful.
38. Missing data. Entered rooms <- c(2, 1, 1, NA, 4), mean(rooms). Output NA. Successful.
39. Entered max(rooms). Output NA. successful.
40. Entered mean(rooms, na.rm = TRUE). Ouytput 2. successful.
41. Entered max(rooms, na.rm = TRUE). Output 4. successful.
42. For data missing values. Entered rooms[!is.na(rooms)]. Successful.
43. Entered na.omit(rooms). Successful.
44. Entered rooms[complete.cases(rooms)]. Successful.
45. Exercise - create a new vector with the NAs removed. Looked at solution to work backwards. Used median function. Copied and pasted solution to check. Successful.

Starting with data

1. Load data into R memory using read csv function. Copied and pasted multi line command. A lot of output, took some time to load but successul.
2. Check data has been loaded. Entered interviews. Successful. data appears in output.
3. Data frames. Entered class(interviews). successful.
4. Indexing and subsetting data frames. Entered interviews[1, 1]. Successful.
5. entered interviews[1, 6]. Successful.
6. Entered interviews[[1]]. Multiple numbers appear in output. Successful.’
7. Entered interviews[1]. Successful.
8. Entered interviews[1:3, 7]. Successful.
9. Entered interviews[3, ]. Successful.
10. Entered head\_interviews <- interviews[1:6, ]. Change in top right panel.
11. Entered interviews[, -1]. successful.
12. Entered interviews[-c(7:131), ] . Successful.
13. Subset data frames by calling column names. Copied and pasted multi line command. Successful, column names appeared in output.
14. Exercise - create data frame, use tail and nrow functions. create new data from from last row. combine functions to reproduce behaviour. Could not work out solution by myself, looked at the provided solution to understand how they produced it. Transferred solution into r.
15. Factors. entered respondent\_floor\_type <- factor(c("earth", "cement", "cement", "earth")). Changes in top right panel.
16. Entered levels(respondent\_floor\_type). successful.
17. Entered nlevels(respondent\_floor\_type). successful.
18. reorder levels in vector. Entered respondent\_floor\_type # current order. Successful.
19. Entered respondent\_floor\_type <- factor(respondent\_floor\_type, levels = c("earth", "cement")), respondent\_floor\_type. successful.
20. Converting factors. Entered as.character(respondent\_floor\_type). Successful.
21. Entered year\_fct <- factor(c(1990, 1983, 1977, 1998, 1990)),as.numeric(year\_fct). Not correct.
22. Entered as.numeric(as.character(year\_fct)) . Years appear. successful.
23. Recommended way. Entered as.numeric(levels(year\_fct))[year\_fct]. successful.
24. Renaming factors. Copied directly the long command from episode. Output appears. successful.
25. Create bar plot. entered plot(memb\_assoc). Plot appears in bottom left panel. Successful.
26. Copied and pasted multi-line command. Successful.
27. Create bar plot for different data. Plot replaced with new one in bottom left corner.
28. exercise - rename levels of factors and create bar plot. Looked at solution to see how they did it. Transferred to R. Successful.
29. Formatting dates. Entered str(interviews).
30. Use lubridate package. Load package. Entered library(lubridate). **Error:** received message saying date was masked from package. not sure why. It says masked from package:base. Maybe not available on what I am using but did successfully install tidyverse.

Introducing dplyr and tidyr

1. Load tidyverse.
2. Use dplyr functions.
3. Select colums and filter rows. Enter select(interviews, village, no\_membrs, years\_liv). Successful.
4. Select rows on specific criteria. Entered filter(interviews, village == "God"). successful.
5. Pipes. Entered interviews2 <- filter(interviews, village == "God"), interviews\_god <- select(interviews2, no\_membrs, years\_liv). Changes in top right panel.
6. Nest functions inside one another. Entered interviews\_god <- select(filter(interviews, village == "God"), no\_membrs, years\_liv).
7. Entered interviews %>%, filter(village == "God") %>%, select(no\_membrs, years\_liv). Successful.
8. Create new object with smaller data, assign a new name. Entered interviews\_god <- interviews %>%, filter(village == "God") %>%, select(no\_membrs, years\_liv), interviews\_go. Successful.
9. exercise - using pipes, subset interview data. Looked at solution to work out how they did it. Transferred into R to check. Successful.
10. Mutate. Enter interviews %>%, mutate(people\_per\_room = no\_membrs / rooms). Successful.
11. Insert filter into chain. Entered interviews %>%, filter(!is.na(memb\_assoc)) %>%, mutate(people\_per\_room = no\_membrs / rooms). Successful.
12. exercise - create new data frame that meets specific criteria. Entered interviews\_total\_meals <- interviews %>%, mutate(total\_meals = no\_membrs \* no\_meals) %>%, filter(total\_meals > 20) %>%, select(village, total\_meals).
13. Split apply combine data analysis and summarise function.
14. Entered interviews %>%, group\_by(village) %>%, summarize(mean\_no\_membrs = mean(no\_membrs)). Successful.
15. Group by columns. Entered interviews %>%, group\_by(village, memb\_assoc) %>%, summarize(mean\_no\_membrs = mean(no\_membrs)) .
16. rearrange results to inspect values. Entered interviews %>%, filter(!is.na(memb\_assoc)) %>%, group\_by(village, memb\_assoc) %>%, summarize(mean\_no\_membrs = mean(no\_membrs), min\_membrs = min(no\_membrs)) %>%, arrange(min\_membrs). Successful.
17. Sort in descending order. Copied and pasted long command. successful.
18. Counting. Entered interviews %>%, count(village). Successful.
19. Entered interviews %>%, count(village, sort = TRUE).
20. Exercise - how many households have an average 2 meals/day, 3 meals/day. used solution to see how the command worked, transferred to R.
21. Spreading and gathering. Copied and pasted multi line commands. Successful.
22. Applying spread to clean data. Copied and pasted multi line command. Successful.
23. Entered interviews\_items\_owned <- interviews %>%.
24. Entered separate\_rows(items\_owned, sep=";") %>%
25. Entered mutate(items\_owned\_logical = TRUE) %>% spread(key = items\_owned, value = items\_owned\_logical, fill = FALSE)
26. make a table showing number of respondents in each village. copied and pasted multi-line command. Successful.
27. Calculate average number of borrowed items. Copied and pasted multi line command. successful.
28. Exercise - create new data frame that has a column for each month. records true and false. did not know how to do this on my own, used solution. Transferred to R, successful.
29. exercise - how many months were respondents without food. Used solution and transferred. Successful.
30. exporting data. Copied and pasted very long command. Changes in top right panel. Successful.
31. Save data frame to data output directory. Entered write\_csv(interviews\_plotting, path = "data\_output/interviews\_plotting.csv"). manually checked data output directory, file appears. Successful.

Backups

Googled and installed Duplicati.

Not sure how to use it, or what I can link to it. Watching episodes of tutorials on youtube here: https://www.youtube.com/watch?v=sJUydn2feIo&list=PL0gtWo1zHMjypJQAP1bcxRkr-lls7uztm&index=4

Configured new back up. Named duplicati backup\_foar705. encryptions settings left on default. passphrase: foar705 (weak password but for purposes of this backup - okay).

All uni work stored on my computer in the directory “Uni” will be backed up on duplicati every 3 days at 9pm. Next back up will be 5 October. Backups won’t occur on weekends.

username for backup: ellenkirkpatrick password:duplicatibackup123

**Proof of concept project**

Started again on github project for PoC and working through user stories.

Closed user story/issue: identifying commonalities. Process identified in previous journal. In short, download sources directly to Zotero and a test folder, upload sources to Voyant tools. Voyant server dropped due to unreliability and unresponsiveness.

Issue closed and moved to tested and complete in project.

Extracting metadata

Moved second issue/user story: extracting metadata and source directly from Voyant to in progress.

Opened issue. Submitted comment about how this user story is no longer required. Due to changes in the first user story, identifying commonalities, the source is saved already in Zotero. There is no need to re-extract the source and metadata from Voyant.

Issue closed. Created a new column in project “not required/removed”. Moved closed issue to that column.

Saving source/metadata in same spot

Moved issue/user story: saving metadata and source in the same place to in progress column. Commented on issue the steps that are required to do this. This user story will now take place second, due to changes with other user stories. It is now much more straightforward as it does not need to extract from Voyant. When extracting the source from the database, it comes with all information attached. Zotero is successful at storing these together, clicking on the source in the library demonstrates it can store both together.

Annotations

Annotations user story is dependent on the source being successfully saved to Zotero. It is also dependent on being able to identify common themes through Voyant otherwise there will be insufficient information for annotations.

Opened issue. Followed acceptance criteria to test whether Zotero can add annotations. Successful. Notes can be added to each source, and they can be edited and accessed whenever by clicking on the source.

Issue closed and moved to tested and complete.

Grouping

Final user story of proof of concept project. Completion of grouping means that sources are succesfully stored together, and annotated and grouped depending on their relevance. This means further down the research process they can be accessed at any time and content not lost. It also makes the final reporting stages much easier as the metadata is stored in Zotero and can be transferred over to publishing programs.

The grouping user story is dependent on the annotations user story. Without annotations, there is insufficient information to decide on how the sources should be grouped.

Opened issue. Followed acceptance criteria to add tags to sources. Can be done in Zotero. Select source and in right hand panel click tags. Input relevant tag. Multiple tags can be added at once. Repeat steps to add tags to other sources. If typing similar, an existing tag will pop up. Labelling sources with the same tag will link them together.

Search for tags in internal Zotero searchbar. Sources will appear by their tag names. Zotero successful at grouping sources together.

Closed issue. Moved to tested and complete.

Project - general

All user stories/issues now closed. All in tested and complete, except for extracting source/metadata from Voyant which is no longer required. It is in its own column.

Not sure where to go from here, have attempted to automate the process through Voyant shell but was unsuccessful. Can begin writing an entire workflow process on how to complete these steps in order to store data more efficiently. Otherwise unsure where to go from now.