# R and RMarkdown Notes

Lisa Luff

6/14/2020

# R Instructions (Using RStudio)

### Setting up a new R project

1. Click on project on the top right hand corner
2. Choose new project
3. Choose existing directory
4. Use the browse button
5. Select the directory you want to use
6. Select create project

### Setting up a new R script

1. Click on the empty page with a green plus just under file
2. Select R Script
3. Set up the top of the document with:

- Name of the script
- Your name and the date
- Put in edit dates as you go
- The purpose of the script

4. Then you can write the script

### Useful packages to have installed

- To install packages use install.packages(packagename)
- Need to call on packages to use them, do this with library(packagename)
- rtools (Basic tools for using R)
- ggplot2 (Get good graphs)
- DiagrammeR (Get fancy graphs)
- rjson (Use data from JSON files)
- RMarkdown (Create documents in RStudio)

### Importing data into an R project

- If you have the data in a file/document on your computer, you can look in the directory and click on the appropriate data file
  - Or you can type in load("DocumentName")
    - This can be used for .RData documents
- If you have text, excel, SPSS, etc documents, you can import them by clicking the import dataset button at the top of the environment window
  - Or you can use read.table() to pull in the information and assign it to a variable
- To use JSON files
  - You will need the package rjson and to call on it
  - Use fromJSON(file= "filname" OR "URL")

**R basics**

- Standard symbols:
  - + plus
  - − minus
  - ∗ multiply
  - / divide
  - ˆ to the power of
  - () brackets
- Standard functions:
  - sqrt(x) Square root of x
  - exp(x) e to the power of x
  - log(x) natural log
  - log10(x) log with base 10
  - sin(x), cos(x), tan(x), etc for trig functions
  - pi to represent the number pi
  - choose(n, k)
  - factorial(x)
- Assigning variables <-, which can be done with Alt -
- If writing two or more commands on the same line use ;
- If you want to reuse the previous command use the up arrow
- To view data use print()
- Use the little broom at the top of the environment window to erase the global environment

**Creating vectors**

- Use c(x, y, z)
- If you assign a standard function to a variable, it will create a vector based on the specifications you enter in the function
- To create a vector of numbers within a range by assigning a variable as x:y:z, where x is the start number, y is how much to go up by with each step and z is the end of the range
- To view or create a subset of data:
  - To view, just use the following without assigning
  - To create you can assign
    - x [x:y:z]
    - x [c(x, y, z)]
- To use a non-standard function to create a vector:
  1. Set up the empty vector:
  - Variable = as.vector(NULL)
  2. Use a for loop to tell it how many elements to run the upcoming function through (the length you want the function to be will be x)
  - for (i in 1:x)
  3. Now use the the function to create the value for each element i
  - Variable[i] = function()
    - You can use [i] in the function if you want the elements to use the value for i in the function
    - Or you can use a preset vector in the function or the same length as the new vector you want to create, and it will use the value in the matching element
    - Doesn't have to be a pre-made function, you can put something like 4 + 2*[i]/PreSetVariable

- Generate a set of random variables:
  - Standard random variables
    - sample(x = a, size = b)
    - a is the maximum value you want the variable to possibly take
    - b is the number of variables you want to generate
  - Random variables of various distributions
    - runif(n) for n variables from a uniform distribution
    - rnorm(n) for n variables from a normal distribution
    - rexp(n) for n variables from an exponential distribution

**Analysing vector data**

- head(x) first few elements
- tail(x) last few elements
- length(x) how many elements are in a vector
- min(x) smallest value
- max(x) largest value
- median(x) median
- mean(x) mean
- weighted.mean(datavector, probabilityvector) discrete E[X]

- sd(x) standard deviation
- var(x) variance
- fivenum(x) 5 number summary( min, Q1, median, Q3, max)
- quantile(vector, nthp)

**Tables and data frames**

- A data frame allows a mix of numerical and non-numerical data, and factors in a single matrix
  - You can call on a specific variable in a dataframe:
    - DataFrame$Category

  - To create a dataframe use data.frame()
    1. Create vectors - The vector names will be will be the names of the columns, and what you call on
    2. Use data.frame(Vector1, Vector2, Vector 3) to create the dataframe
    3. Use stringsAsFactors = TRUE/FALSE depending on if you want your non-numerical factors to be considered as factors or not (probably not)
- To set up a table, you can use as.table()
  - You can put imported data into a table by calling on the variable holding the data
  - Or you can manually create a table yourself by combining vectors with rbind()(To combine as rows) and cbind()(To combine as columns)
    - SampleTable <- as.table(rbind(c(a, b, c), c(d, e, f), c(g, h, i)))
  - You can label your dimensions using dimnames()
    - You can set the names as string using list()
    - dimnames(SampleTable) <- list(RowName = c(“A”, “B”, “C”), ColumnName = c(“D”, “E”, “F”))

**Graphs**

- plot(x = a, y = b, ...)
  - Create a standard graph
  - a is the vector for the x axis
  - b is the vector for the y axis
    - These don't have to be given by separate vectors, but can be
  - main = "x" is the title for the graph
  - xlab = "x", ylab = "y" is the labels for the specified axis
  - type = "z" is the type of marking the plot will use
    - p for points
    - l (lower case L) is line
    - b for both
    - h for vertical lines
    - s for steps
  - col = "colourname" to decide the colour of the markings
- barplot(x, y, ...)
  - Create a graph that shows the number of values that are equal to the same value
- hist(x, y, ...)
  - Similar to a barplot, but shows how many are within a range of values
- boxplot(x)
  - Gives a boxplot with the values of the fivenum() function
- stem(x, scale = x, width = y)
  - Create a stem and leaf plot (the first value of a decimal or large number on the left, and all the following values to the right)
  - scale = x tells it how many variables you want on the left of the line
  - width = how many to show to the right
- If you want to overlay information from multiple graphs
  - Use the function lines(a, b, col = "c") to overlay a line over the graph specified directly above it

**Analysis using distributions**

- Approximations:
  - Normal distribution
    - Probability
      - pnorm(Z)
      - OR pnorm(x, mean = a, sd = b)
    - Z score
      - qnorm(p)
  - T distribution
    - Probability
      - pt(x, df)
    - T score
      - qt(p, df)
  - Chi-squared
    - Probability ($\alpha$)
      - pchisq($\chi^2_{df}$, df)
    - $\chi^2_{df}$
      - qchisq($\alpha$, df)

- Discrete distributions:
  - Binomial
    - Probability
      - pbinom(x, n, p(S))
    - Random variable X
      - qbinom(p, n, p(S))
  - Negative Binomial
    - Probability
      - pnbinom(x, r, p(S))
    - Random variable X
      - qnbinom(p, r, p(S))
  - Geometric
    - Probability
      - pgeom(x, p(S))
    - Random variable X
      - qgeom(p, p(S))
  - Hypergeometric
    - Probability
      - phyper(x, m, n, k)
    - Random variable X
      - qhyper(p, m, n, k)
  - Poisson
    - Probability
      - ppois(x, lambda)
    - Random Variable X
      - qpois(p, lambda)
- Continuous distributions
  - Uniform
    - Probability
      - punif(x, min = a, max = b)
    - Random Variable X
      - qunif(p, min = a, max = b)
  - Log normal
    - Probability
      - plnorm(x, meanlog = a, sdlog = b)
    - Random Variable X
      - qlnorm(p, meanlog = a, sdlog = b)
  - Gamma
    - Probability
      - pgamma(x, alpha, rate = beta, scale = 1/beta)
    - Random variable X
      - qgamma(p, alpha, rate = beta, scale = 1/beta)
  - Exponential
    - Probability
      - pexp(x, lambda)
    - Random variable X
      - qexp(p, lambda)

**Comparing Data**

- T-test
  - Hypothesis testing
    - t.test(Vector, $\mu_0$, alternative = "less""greater""two.sided", conf.level = X.XX)
  - Comparing population means
    - t.test(SampleVector, ComparisonVector, mu = 0, alternative = "less""greater""two.sided", conf.level = X.XX)
- Covariance
  - Joint probability distribution
    - cov(table or dataframe for XY)
  - Matched pair design
    - Pearson: cov(table or dataframe for XY, paired = TRUE, method = "Pearson")
    - Spearman: cov(table or dataframe for XY, paired = TRUE, method = "Spearman")
- Correlation
  - corr(table or dataframe of XY)
- Chi-squared test
  - Goodness-of-fit testing
    - chisq.test(Vector)
  - Independence of two-way contingency tables
    - chisq.test(table or dataframe of XY)

# RMarkdown Instructions

### Setting up a new RMarkdown

1. Click on the empty page with a green plus just under file
2. Select RMarkdown...
3. Enter the name and choose to output as html, you will still be able to output as cdf or word by clicking the arrow to the right of the knit button at the top of the RMarkdown window
4. You can erase everything up to the r setup box
5. Now you can write the document

### RMarkdown basics

- You can use # to set headings. The more #'s the smaller the heading
  - Headings are used by RMarkdown and pdf's as a way to jump straight to that sectoin of the document, so handy to use them a fair bit even if they're getting really small
- Using a * at the start of a line create a bullet point, and if you use tab it will indent and use a different type of bullet point
  - If you don't use it for every line in a paragraph it messes up though
- You can create a numbered list just using 1. and so on, however you cannot put bullet points inside numbers, if you want to indent you have to use a), i), etc.
- If you want to use any mathemtical symbols you can use LaTeX to write them by putting a dollar sign either side
  - See LaTex commands for more details
- If you aren't using lists or some kind, or you are using LaTeX insert, you need to put a double space to indicate when you want it to be a new line

- You can access a number of commands using backslash
  - Eg. newline or newpage

### Entering code into a document

- You can write code directly into the document by clicking on the green square with a C and a plus, you can choose from several languages
- When coding with R, you can enter a number of commands in the header of the insert, the main ones I use:
  - echo = TRUE/FALSE depends if you want to print the script in the document, if you put FALSE, but want to display the output, just use the print() function
  - comment = NA will hide the line numbering in the document

### Displaying graphs side by side

- To display graphs side by side you need to size them down to an appropriate percentage of the width, depending on how many graphs you want side by side
  - This is done by putting the following in the header of your inserted R code
    - fig.hold='hold', out.width="XX%"
  - If you want side by side, you will need to make sure you don't display your code (echo = FALSE), or it will separate them, so if you want to show code, put the graphs in a separate chunk of code

### Inserting images

- If you want to include images in your document just put:
  - ![ ] (DocumentName)
  - You need to have it saved in your current working file

**Creating columns in RMarkdown**

- If you want columns to appear in html and pdf you will need to enter this at the top of your file in the YAML header under output:
  output:
  pdf_document:
  includes:
  in_header: preamble.tex
  html_document:
  css: preamble.css
  - And be able to access the text documents it references
  - You will also need to put the following in your r setup insert:
    knitr::opts_chunk$set(echo = TRUE, cache = TRUE)
  - To knit to html -
    * The text in the document should say:
      /* See: https://bookdown.org/yihui/rmarkdown-cookbook/multi-column-layout.html */
      .columns {display: flex; }
  - To knit to pdf -
    * The text in the document should say (need to replace work backslash with actual backslash - no spaces):
      %% See: https://bookdown.org/yihui/rmarkdown-cookbook/multi-column-layout.html
      backslash newenvironment{columns}[1][]{}{}
      %%
      backslash newenvironment{column}[1]{\begin{minipage}[t]{#1}}{%
      backslash end{minipage}
      backslash ifhmode backslash unskip backslash fi
      backslash aftergroup backslash useignorespacesandallpars}
      %%
      backslash def backslash useignorespacesandallpars#1 backslash ignorespaces backslash fi{%
      #1 backslash fi backslash ignorespacesandallpars}
      %%
      backslash makeatletter
      backslash def backslash ignorespacesandallpars{%
      backslash @ifnextchar backslash par
      { backslash expandafterbackslash ignorespacesandallpars backslash @gobble}%
      {}%
      }
      backslash makeatother
- Then use the following to create the columns (Without the bullet points and with an actual backslash where I've written backslash), obviously you can change the percentages to add more columns, and the middle one is just there to leave some space between the two columns:
  - :::::: {.columns}
  - ::: {.column width="48%" data-latex="{0.48 backslash textwidth}"}
  - text
  - :::
  - ::: {.column width="4%" data-latex="{0.04 backslash textwidth}"}
  - backslash
  - :::
  - ::: {.column width="48%" data-latex="{0.48 backslash textwidth}"}
  - text
  - :::
  - ::::::
  - backslash newline

**LaTeX Commands**

- Formatting
  - small - makes the symbols approximately the same size as the text
  - to - an arrow pointing right
  - text{} - will display any text normal-ish and will allow for spaces which otherwise don't show
  - being{aligned} end{aligned} - will align multiple lines with each other where an & represents the point you want to be centred in the line and to show a break between the end of one line and the beginning of the next
  - { - will show a curly bracket at the front (works the same for other bracket types), and you can use this with Bigg in front this to create a curly bracket two lines tall, if you have both in front of aligned text, you can create a bracket spanning two lines of information
- Mathematical symbols
  - infty - $\infty$
  - pm - $\pm$
  - ne - $\neq$
  - le - $\leq$
  - ge - $\geq$
  - sim - $\sim$
  - a^{b} - $a^b$
  - a_{b} - $a_b$
  - bar{a} - $\bar{a}$
  - hat{a} - $\hat{a}$
  - sqrt{a} - $\sqrt{a}$
  - frac{a}{b} - $\frac{a}{b}$
  - sum^a_b - $\sum_b^a$
  - int^a_b - $\int_b^a$
  - {a choose b} - $\binom{a}{b}$ OR binom{a}{b} - $\binom{a}{b}$
  - cap - $\cap$
  - cup - $\cup$
  - subset - $\subset$ and subseteq - $\subseteq$
- Greek symbols
  - alpha - $\alpha$
  - beta - $\beta$
  - mu - $\mu$
  - sigma - $\sigma$
  - eta - $\eta$
  - phi - $\phi$
  - lambda - $\lambda$
  - Gamma - $\Gamma$
  - chi - $\chi$
  - nu - $\nu$