R Tutorials: R Coding Etiquette

This tutorial was taken from [This book](http://adv-r.had.co.nz/Introduction.html), which I will go through next

# Script Setup Syntax

## Using Comments is Key way to Start Script

Use # to comment

Use —- to add section to an outline (need at least 4 dashes) (need to do — at the end of the top comment) Doing this will allow you to collapse the section To view outlien use the button on the top right corner Edit/folding/collapse all to collapse all sections expand all to display all code

## Script Structure

Usually are the following sections – each one is outlined with a Title here

###Intro What the script does, author names and contact details ### Libraries Install packages and library all packages beign used It is good to also comment what each is used for

### Functions

Define the functions you are using   
A function will be saved in the global environment and can be used on any script open   
If you save workspace image it will open all same things in global environment but if not it will not. They reccoment not saving workspace image becuase it will make things run more slowly

### Set Working directory

This allows you to keep everything in the same folder

home directory and then set the path using forward slashes / ex: # Set the working directory on Mac/Linux —- setwd(“~/Work/coding\_club/CC-etiquette-master”)

### Importing datasets

# Script Analysis Syntax

Keep analysis in order so it is always consistant The below is an example of ordering

### Section 1: Formatting data

Preeform data transformations Comment if something needs a specific package

use tolower() to make names of all variables lower case

Check data type using str() fix data types needing to be fixed

parse\_number - saves numbers  
 levels() shows the options of var

### Calc summary statistics —-

In piping can do multiple commands  
 group by groups by a variable   
 summarise creates columns

### Vusualize with ggplot2

If put ggplot in () it will visualize- if no outer parenthesis it will only create object  
   
 (barplot <- ggplot(LPI\_biome\_summ, aes(biome, color = biome, y = populations)) + geom\_bar(stat = "identity") +   
 theme.LPI() + # Use of personal theme function  
 ylab("Number of populations") +  
 xlab("Biome") +  
 theme(legend.position = "none")) # Removal of legend for simplicity

#### Outputs

.csv files are easiest to work with

.pdf are the best way to save graphs

.png are easier to insert into docs

good to save images into image subdirectory (must create fulter manually 1st)

# Naming Variables

Not good to give generic names - especially if clean your environment regularly Short but specific

objects and variables should be lowercase

Variable names = Nouns Function names = verbs

Use underscore to seperate words in scripts Use . (dot) to seperate words in functions

Do not mix capital and lowercase words – keep to underscores and dots

# Spacing

Need spaces around all operators Put space after comma No space before left parenthesis No spaces between code in parentheses (example)

More than 1 space is good if it helps allignment

### Commenting

Use two spaces then a # then a space then your comment

### Curly Braces

Always indent the code in a curly bracket closing curly bracket ALWAYS on its own line

Short statements ok on 1 line

## Line spacing

Official rules are 80 characters per line Tools/Global Options/Code/Display/Show Margin/80 characters

when using pipes, each pipe is the end of a line

for ggplot 2, keep the + at the end of each line

### Indents

when having long fuction, indent the lines of code that are not the 1st line

# Tidying Old Code

R can help using R/Reformat Code will add in spaces but will also add new line after each comma Also Make sure to backup scripts

can also use   
 # Reformat your old code to add in spaces and limit line length  
install.packages("formatR")  
library("formatR")  
  
# Set working directory to wherever your messy script is  
tidy\_source("messy\_script\_2017-02-25.R", file = "tidy\_script\_2017-02-25.R", width.cutoff = 100)

Width cutoff is cool and will cutoff the width of any line

Use find and replace within selection to change cariables in a code to nicer values can also change names of variables or how they are written with this code

names(dataframe) <- gsub(".", "\_", names(dataframe), fixed = TRUE)  
# This code takes all of the variable names in the imaginary dataset `dataframe` and replaces `.` with `\_`  
# Depending on the naming style you are using, you might want to go the other way around and use `.` in all variable names  
  
names(dataframe) <- tolower(names(dataframe))  
# This code makes all of the variable names in the imaginary dataset lowercase  
  
colnames(dataframe)[colnames(dataframe) == 'Old\_Complicated\_Name'] <- 'new.simple.name'  
# Renaming an individual column in the imaginary dataset

# RStudio Addins

There are addins to Rstudio to help with formatting   
 his includes littleboxes whish makes boxes around the intro section