Student Exercises | Intro to R

your name here

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
```

Exercise 1.1: Exploring R Studio

1. Take a few minutes to familiarize yourself with the R studio environment by locating the following features:

- The windows clockwise from top left are: the code editor, the workspace and history, the plots and files window, and the R console.
- In the plots and files window, click on the packages and help tabs to see what they offer.
- See what types of new files can be made in R studio by clicking the top left icon- open a new R script.

2. Now open the file called 'Exercises_for_R_Lectures.Rmd'. This file will serve as your digital notebook for parts of the workshop and contains the other exercises.

Exercise 1.2: Intro to `R Markdown` Files

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

1. When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded `R` code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

2. Each code chunk begins and ends in the same way- with a fence (three dashes). You can further specify what you want to show up in your final document using the `echo` and `eval` commands in the opening line. Insert a few code chunks below using the `insert` tab at the top of this window. Then, change the `echo` and `eval` arguments to `TRUE` or `FALSE` and see how different combinations of these arguments change the output when you knit. I have done the first one for you. Notice too that each `R` code chunk requires a unique title argument (here 'cars variant 1'), or the Rmd will not knit.

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
```

```
##  Max.   :25.0   Max.   :120.00
```

3. What do you think `echo` and `eval` do, based on your manipulations?

- In a new line press tab twice, add a `-` and a space, then type your answer (as I did here). This will indent your answer in RMarkdown (`knit` to visualize the difference). Use this format throughout the doc to format your answers.
- What are the defaults for `echo` and `eval`, based on your manipulations?

Excercise 1.3: RMarkdown advanced

Getting more familiar with RMarkdown

- If you want to beautify your output, it always starts here.
- There are many options, and a few are laid out below.
- The `knitr` package has lots of options explained
  - here and
  - here in detail.
- Part of configuring your script is loading the correct packages. Always load all packages together at the top in your `config` chunk. That way future users will know exactly what they need to install.

```
library(scales)
library(knitr)
opts_chunk$set(background = "gray80", tidy = FALSE, cache = FALSE,
    comment = "", dpi = 72, fig.path = "RMDfigs/", fig.width = 4,
    fig.height = 4)
```

2. Generate fake data

- The `x` value is just numbers 1-100 for an x axis value. This might be time or distance, etc.

- For the response variable, generate a random normal distribution with the `rnorm` function, and then add a trend with the `seq` function.
- Then we'll add some fake treatments with `letters`.

```
# setwd('~/Desktop')

x <- 1:100
y <- rnorm(100, sd=3) + seq(10.05, 20, 10/100)
z <- factor(rep(letters[1:5], each=20))
dat <- data.frame(x, y, z)
```

3. Tables in `knitr`

- This is an ugly way to preview data or display tables.

```
head(dat)
```

```
  x         y z
1 1  9.777707 a
2 2 13.822462 a
3 3  6.832800 a
4 4  7.771385 a
5 5  6.860758 a
6 6 13.033267 a
```

- The `knitr` package has a simple built-in function for dealing with tables. This works well in either html or pdf output.

```
kable(head(dat))
```

x	y	z
1	9.777707	a
2	13.822462	a
3	6.832800	a
4	7.771384	a
5	6.860758	a
6	13.033267	a

4. R commands embedded in prose

- One of the best features in `knitr` and RMarkdown generally, is the ability to embed real R commands in sentences, so that you can report actual values instead of constantly copying and pasting when results change a little bit.
- This table has 100 rows and 3 columns. The 'x' variable starts at 1 and ends at 100.
- -0.0061234

5. Formatting text in RMarkdown

- Create a formatted list with 2 levels and 2 sub levels; make one of the sub levels italic and the main levels bold
- Create a quote from one of your new class friends

Exercise 1.4: Basic Mathematics in `R`

Insert a code chunk below and complete the following tasks:

1. Add and subtract
2. Multiply and divide
3. Raise a number to a power using the ^ symbol
4. Create a more complex equation involving all of these operations to convince yourself that `R` follows the normal priority of mathematical evaluation

---

Exercise 1.5: Assigning Variables and Arithmetic Functions in `R`

Insert a code chunk below and complete the following tasks:

1. Assign three variables using basic mathematical operations
2. Take the log of your three variables
3. Use the print function to display your most complex variable
4. Use the `concatenate` function to print a sentence

Exercise 1.6: Vectors and Factors

Insert a code chunk below and complete the following tasks:

1. Create a numeric vector using the `c` function
2. Create a multi-level character factor using the `c` function

```
vec1<-c("I", "am", "great","at","R")
fac1<-as.factor(vec1)
print(fac1)
```

```
[1] I     am    great at    R
Levels: am at great I R
```

3. Use `str` and `class` to evaluate your variables

Exercise 1.7: Basic Statistics

Insert a code chunk below and complete the following tasks:

1. Create a vector and calculate the `mean`, `sd`, `sum`, `length`, and `var`
2. Use the `log` and `sqrt` functions on your vector
3. What happens when you try to apply these functions to a factor?
4. Type the first couple letters of a function within your R code chunk, then hit tab- what happens?

- What if you press tab with you cursor inside the function parentheses?

Exercise 1.8: Creating Larger Vectors and Random Sampling

Complete the following tasks in the codechunk below: - Note: If you ever want someone else to be able to perfectly reproduce your results, always set the random seed at the top. Any number will do. Note that it never hurts to set the seed, *but* robust results should always stand up to random number generators.

1. Create a vector with 100 elements using the `seq` function and calculate two basic statistics on your vector
2. Create a variable and `sample` it with equal probability

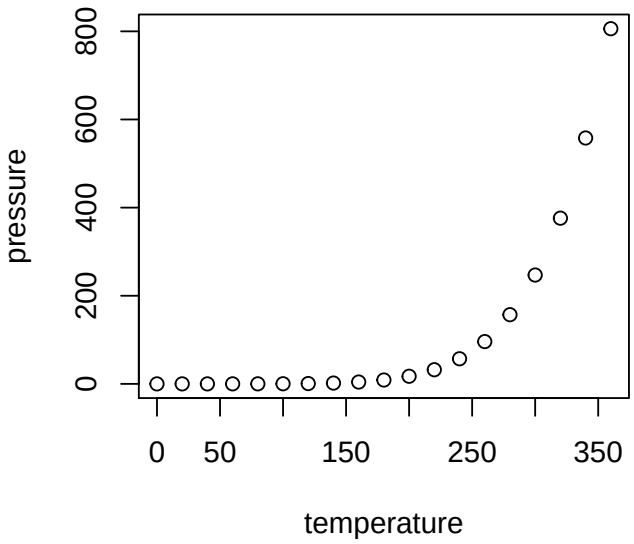
- Can you figure out what the arguments in the parentheses mean?
- Try varying the arguments to see what happens.

3. Create a normally distributed variable of 10000 elements using the `rnorm` function then `sample` that distribution with and without replacement
4. Use `hist` to plot your normally distributed variable

```
set.seed(1415)
```

Including Plots

You can also embed plots in your pdf document (`knit` to view), for example:



- Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
- Note that you can also alter the size of the plot in the chunk header (`{}`) section.

Exercise 1.9: Basic Visualization

Insert a code chunk below and complete the following tasks, make sure to label all plot axes and have fun with colors!

1. Create a variable using `seq` and make two different plots by changing the `type` argument
2. Create a normally distributed variable using `rnorm` and make two different plots using `hist` by varying the `breaks` argument (what does `breaks` appear to do?)
3. Modify your `par()` arguments to create a composite figure of the above graphs.

Exercise 1.10: Creating a Data Frame and Evaluating Class

Insert a code chunk below and complete the following tasks:

1. Recreate the dataframe from the slides by creating each vector then using `data.frame`
2. Assign rownames to your dataframe using `rownames` and `c`
3. Get class assignments for your whole dataframe using `str`
4. Calculate the `mean` of each numeric variable
5. Make a descriptive plot of your choosing
6. What happens when you use the functions `head` and `tail` on your dataframe?

Exercise 1.11: Datasets and Indexing

By opening this .Rmd file, you have automatically set your working directory to the folder containing it. Now, you can access data from this directory or a sub-directory in this folder. You can do this by including that part of the path in the `read.csv` function. Insert a code chunk below and complete the following tasks:

1. Save the file we created together in a sub-directory of your current working directory
2. Use `read.csv` to read your file in

3. Use `str` and `head` to view your data structure
4. Use the `$` and `[ ]` operators to select out different parts of the dataframe.
5. Plot temperature over elevation using `$`.
6. Use the `tapply` function to calculate the `mean` and `var` of temp by habitat type and temp by elevation.
7. Export your data frame with a different file name
