Laboratory #4.
Probability Values by Randomization.

In the sciences the classical approach to measurement variability in research results has been frequentist statistics. The frequentist approach addresses this uncertainty by asking: How often could a sample statistic, such as a mean, have been obtained by chance from a population consisting of many runs of the same study? To answer this question we compare the observed statistic with the distribution of results due only to chance. In the Fisher version of inferential statistics we make a judgement from a *p*-value computed from the distribution of outcomes under the null hypothesis, that results are due only to chance. In the Neyman-Pearson version of inferential statistics we make a formal decision against a pre-set criterion (*e.g.* 5%) using the *p*-value. The statistical distributions that are commonly used to calculate these *p*-values are the *F*, *t*, and χ^2 distributions.

We cannot always assume that one of these distributions will represent the distribution of our result under the null hypothesis. The assumptions for these distributions (errors are normal and homogeneous) may not hold for our data. Or we may have a statistic for which the statistical distribution is unknown.

In these cases we can always evaluate our statistic by constructing a frequency distribution of outcomes based on repeated sampling of outcomes when the null hypothesis has been made true by random sampling of the data. This approach to statistical inference is called a **randomization test**. It requires no assumptions about the deviations of the data from the model. It will work for any statistic you might devise, or any set of data you might encounter. Strictly speaking it applies only to the batch of data at hand. Inference beyond the batch depends on whether the batch is a representative sample from some larger population to which we wish to infer.

Table 4.1. Generic recipe for randomization test

1. Compute a statistic (observed outcome)
2. Make the null hypothesis true by randomizing the data
3. Re-compute the statistic to obtain an outcome when the null hypothesis is true
4. Repeat this many times
5. Construct a frequency distribution of outcomes when the null hypothesis is true
6. Compare the observed outcome to the distribution of outcomes, in order to calculate a probability value.

The goal of this laboratory is to demonstrate how to obtain the probability of a statistic via randomization. The examples will be simple: tests of whether two means differ to a statistically significant degree.

Once you have completed the lab and write-up, you should have

- an understanding of the logic of hypothesis testing, based on a randomization example
- a working knowledge of the mechanics of computing *p*-values by randomization

The data for his lab are on the course website: www.mun.ca/biology/schneider/b4605/Data/Labs

DaphniaAges.txt	Box 9.5 in Sokal and Rohlf 1995
LitterSize.txt	Box 13.12 in Sokal and Rohlf 2012

To begin, look again at Table 4.1 and then draw a flow diagram (boxes and arrows) showing how to do a randomization test.

Computing the *p*-value by randomization can be done in a spreadsheet. Because a spreadsheet by itself has limited statistical capacity, inference by randomization will be demonstrated using statistical packages, so that you become familiar in using these.

The randomization test will be demonstrated with data from Box 9.5 of Sokal and Rohlf (1995). The research question:

Does the average age at beginning of reproduction in one strain of *Daphnia longispina* differ from that of another strain?

To begin, -Download the *Daphnia* data file from the course website to your desktop.
-Double click on the file to open it and look at the data.
How many rows of data do you see? _____

Next we import the data into the statistical package.
This lab can be done in a spreadsheet (excel), in a code only package (R),
or in a package with spreadsheet and code (Minitab). Your choice for this lab is _____
With most statistical packages we can simply select the data by highlighting it, copy it, paste it into the spreadsheet in the package, and type in the name of each column: *Strain1 Strain2*.

Here is pseudocode (applies to any package) for defining data using copy and paste

Define Data from file

```
Open file that has data, usually on the desktop.
Use mouse to highlight the data
Copy the highlighted data
Paste onto the stat package spreadsheet interface or datafile
Name the columns
```

If you are using spreadsheet or a package do that now and skip the R-code.

```
In a spreadsheet, place names at top of a column
In the ASCII (text) file copy both columns.
Paste into spreadsheet.
This works well if the columns are separated by a tab, or a
single space, or a comma
```

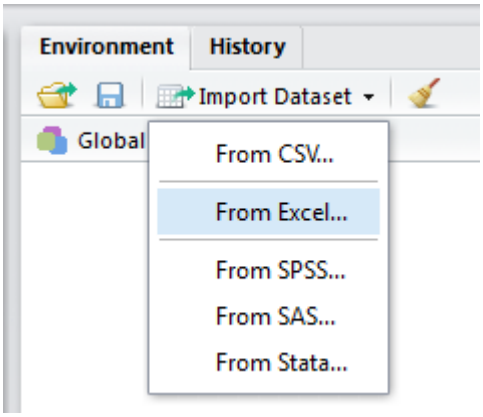
.

Copy and paste can be done in R.

R-code for columns separated by a tab. The space between " " is a tab.

```
# highlight data columns and copy to clipboard
# then in R issue commands
daphnia <- read.delim("clipboard", sep="  ",
header=FALSE)
names(daphnia) <-c("Strain1","Strain2")
```

Define Data from file



Until recently R-studio allowed Import Dataset for text files. R-studio now only allows import from CSV or proprietary software (Excel, SPSS, SAS, Stata).

In order to use Import Dataset in R-studio you will have to copy the data from the text file to a csv or excel file, then import it.

In R we can use code to browse for and import any data file, including text files.

```
Daphnia <-read.table(file.choose(),nrows=7)
names(Daphnia) <-c("Strain1","Strain2")
```

Define Data from file

This is a little risky, because of the problem of separators. But it usually works.
Names for each column of data (vector) do not appear in this text file, These are added in R.

Having defined the data, we calculate the difference in means between *Strain1* and *Strain2*

Here is pseudocode for calculating difference of means of two variables (columns)
Pseudocode applies to any package or spreadsheet.

Select a location to place *AvDiff*
Define the function *AvDiff = mean(Y1) – mean(Y2).*
Calculate *AvDiff = mean(Y1) – mean(Y2).*

Calculate statistic AvDiff

Now that you have looked at the pseudocode, use your statistical package to compute the difference in mean ages of the two strains of *Daphnia*..

	A	B
1	Strain I	Strain
9	Mean	Mean
10	7.514	7.557
11		
12	AvDiff	-0.043
13		
14	Grand Mean	
15	7.536	

Calculate statistic AvDiff in excel

A10 =AVERAGE(A2:A8)

A12 =A10-B10

A15 =MEAN(A2:B10)

To use code in Minitab, find out how to force the package to display the commands.

Here are the command lines, from Minitab

```
MTB > let k1 = mean(c1)
MTB > let k2 = mean(c2)
MTB > let k3 = k1 - k2
MTB > print k1 k2 k3
K1      7.51429
K2      7.55714
K3      -0.0428576
```

Calculate statistic AvDiff in Minitab

Here is the calculation in R.

```
k1 <- mean(Daphnia$Strain1)  #Filename$Variablename
k2 <- mean(Daphnia$Strain2)
k3 <- k1 - k2
k3
```

Calculate statistic AvDiff in R

Now, report the difference (which will be in units of days) to 3 significant figures; then re-write it in units of hours.

AvDiff = _____ days *AvDiff* = _____ hours

Laboratory #4. Randomization tests

Could this difference be merely a matter of chance? To find out we randomize the data, to compute the difference in time to reproduction due to chance variation.

Pseudocode for calculating random difference of means (applies to any package)

Stack variables *Y1* and *Y2* into variable *Y3*
Sample 7 values from variable *Y3* to random variable *RV1*
Sample 7 values from variable *Y3* to random variable *RV2*
Select a location to place *RanDiff*
Define the statistic *RanDiff = mean(RV1) – mean(RV2).*
Calculate *RanDiff = mean(RV1) – mean(RV2).*

Sample to RV1, RV2

Define RanDiff
Calculate RanDiff

Now that you have looked at the pseudocode, compute a random difference in mean age between the two strains of *Daphnia*.

Here is a typical code sequence in a menu-based package, Minitab

```
MTB > stack c1 c2 c3
MTB > sample 7 c3 c4;
SUBC> replace.
MTB > sample 7 c3 c5;
SUBC> replace.
MTB > let k1 = mean(c4)
MTB > let k2 = mean(c5)
MTB > let c8(1) = mean(c4) - mean(c5)
MTB > print k1 k2 C8
  K1     7.48571  <---your values may differ
   K2     7.5857     <---
   C8     -0.1       <---
MTB > stack c8 c6
```

Stack to single column
Sample to RV1, RV2

Calculate and Store RanDiff

Place RanDiff in a column (vector)

The random difference in time to reproduction, based on the randomization in Minitab, was 0.1 day, or about 2 hours earlier out of 7.5 days. This value is the difference in time to reproduction that can arise due to chance sampling of the data

Here is the same sequence in R-code. Run each step and display the result.

```
Y_all<-c(Daphnia$Strain1,Daphnia$Strain2)
```
Stack to single col

```
RV1<-sample(Y_all,size=7,replace=TRUE)
```
Sample to RV1

```
RV2<-sample( use copy paste from above to fill this in )
```
Sample to RV2

```
RanDiff <- mean(RV1) - mean(RV2)
```
Calculate RanDiff

```
RDvector <- Randiff
```
Store result

To obtain a random difference in means in R, assemble the code in the R-script window
To make sure it is correct and to undertand what you are doing, Run the code line by line.

```
Y_all<-c(Daphnia$Strain1,Daphnia$Strain2)
RV1<-sample(Y_all,size=7,replace=TRUE)
RV2<-sample(Y_all,size=7,replace=TRUE)       )
mean(RV1)
mean(RV2)
RanDif<-mean(RV1) - mean(RV2)
RanDif
RDvector<-(RanDif)
RDvector
```
Stack to single col
Sample to RV1, RV2

Calculate RanDiff

Store RanDiff in a column

Your random difference will differ from the one shown in the Minitab box because it is a new sample of the data.

To obtain a random difference in a spreadsheet we use a random number generator to assign a random rank to each value of Age, then use the random rank to choose values to place in the random vectors *RV1* and *RV2*

=RANK(F2,F$2:F$15,0)

	C	D	E	F	G	H	I
1	*Age*	Rank		Rand(0,1)	Rank	*RV1*	*RV2*
2	7.2	9		0.00	15,0)	6.7	7.4
3	7.1	13		0.56	7	7.5	7.3
4	9.1	1		0.39	11	7.6	7.5
5	7.2	9		0.00	14	7.2	9.1
6	7.3	8		0.48	10	7.7	7.2
7	7.2	9		0.99	1	7.2	7.1
8	7.5	5		0.81	4	7.2	8.8
9	8.8	2		0.01	12	Mean	Mean
10	7.5	5		0.75	5	7.300	7.771
11	7.7	3		0.50	9		
12	7.6	4		0.90	3	*RanDiff*	-0.471
13	7.4	7		0.70	6		
14	6.7	14		0.95	2	Grand Mean	
15	7.2	9		0.53	8	7.536	

D2 =RANK(C2,C$2:C$15,0)
Use cursor to copy downward from D2
F2 =RAND()
Use cursor to copy downward from F2
G2 =RANK(F2,F$2:F$15,0)
Use cursor to copy downward from G2
And now, a mysterious function to place a random value from *Age* (col 2) to *RV1*
H2 =INDEX($C$2:$C$15,G2)
Use cursor to copy downward from H2
I2 =INDEX($C$2:$C$15,G9)
Use cursor to copy downward from I2
Calculate means and *RanDiff*

In order to assign a probability to the observed difference *AvDiff* we need to accumulate many values of *RanDiff*. To accumulate chance differences, we define a sequence of commands that we can execute repeatedly. Once this sequence works, you can execute this batch again and again. Here is a shortened version of the sequence of commands to add a random difference to a column called RDvector.

```
RV1<-sample(Y_all,7,TRUE)
RV2<-sample(Y_all,7,TRUE)
RanDif<-mean(RV1) - mean(RV2)
RDvector<-c(RanDif,RDvector)  #stack values in RDvector
RDvector
```

Add RanDiff to column

Now, in your package, run your batch several times. As you execute this batch of commands repeatedly in your package, you should see the random differences accumulate in the appropriate column. In R, you can add a line of code to the batch file: `hist(RDvector,breaks=11)` to watch the histogram develop.

Running code step by step, then running a batch of files repeatedly gives us a 'hands on' sense of what a randomization test is doing. However, executing a batch repeatedly is impractical in most applications of randomization, which require thousands of runs. So the next step is to create a macro where we can specify the number of runs of the batch file.

Pseudocode for running a macro to accumulate random differences (applies to any package)

```
Define a sequence of commands
Place into a file (called a macro)
Execute the file a specified number of times.
```

Define macro

Execute macro

Macro routines are available in excel and in menu-based programs. In Minitab, it is easier to run your batch of code 100 times than to set up the Macro. In SAS a randomization Macro requires a page of code . See: *Behavior Research Methods, Instruments, & Computers* 25:406-409. Macro routines in R are not pretty.

```
# on Microsoft Windows (adjust the path to R.exe as
needed)
"C:\Program Files\R\R-2.13.1\bin\R.exe" CMD BATCH
   --vanilla --slave "c:\my projects\my_script.R"
```

In R we can combine commands to a single command and specify the number of runs

Here is the R batch file combined to a single command line, with 2 runs.

```
RDvector<-c(replicate(2,
 (mean(sample(Y_all,7,TRUE))
 -mean(sample(Y_all,7,TRUE)))))
```

Define and Execute macro

In today's lab we are going to restrain ourselves and accumulate only 100 runs, so that we can see what we are doing when we calculate a *p*-value. It only takes a minute to run your batch file 100 times. In R you can run the single code line shown above and specify 100 runs.

Whew! We're almost done. We have a column of random differences in the means of the age of first reproduction in *Strain1* compared to *Strain2* We made the null hypothesis true by random sampling and now we have the distribution of outcomes (differences of means) when the null hypothesis --no difference-- is true.

The next step is to display the distribution. All statistical packages have a readily available routine for displaying a histogram for a column of numbers (differences in means, in this case).

Pseudocode for creating and displaying a histogram

```
Name the variable
Apply histogram routine.
```

Display histogram

Here is Minitab code to generate the histogram (which will need labels)

```
MTB > name c6 'randiff'
MTB > histogram c6
```

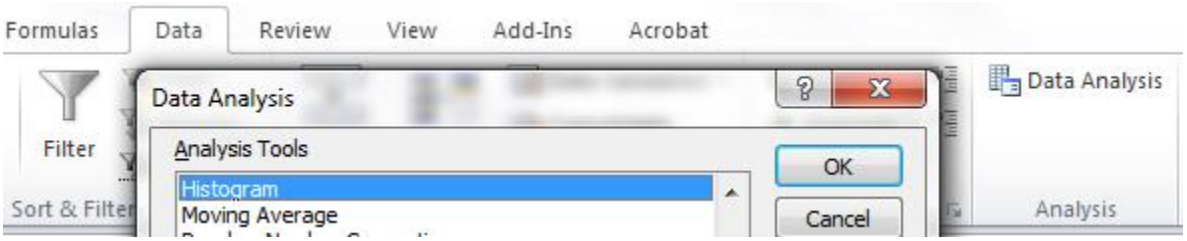
Display histogram

Here is the Rcode for a well-labelled histogram

```
hist(RDvector,breaks=15,
xlab="randomized difference",
main="Figure 1. Randomized difference in mean age of
first reproduction in two strains of Daphnia
longispina.")
```

Display histogram

The histogram routine in excel is found in the Data Analysis section of the Data tab. If the Data Analysis section does not show in the Data tab, you will need to load the Analysis Toolpak. Go to File tab, options, Add-ins to load the toolpak.



Once we have a distribution, we can use it to calculate the probability of *AvDiff*, our observed statistic (*St* = -0.04286). We will use a two-tailed test. That is, we will consider more extreme positive and more extreme negative values. The probability $Pr\{|X| \leq St\}$ is the proportion of the distribution that is equal to or more negative than the observed negative value $St = -0.04286$. The probability $Pr\{|X| \geq St\}$ is the proportion of the distribution that is equal to or more positive than the positive value $St = +0.04286$. We count the number of values in both tails, the left tail $N(X \leq St)$ and the right tail $N(X \geq St)$. We add the tails to obtain the proportion in both tails of the distribution.

Pseudocode for calculating a probability $Pr\{|X| < St\}$ from a histogram

Sort values	
Find closest value to St in left tail (small values)	***Sort to new column***
Count the number of values less than St	$N(X \leq St)$
Find closest value to St in right tail (large values)	$N(X \geq St)$
Count the number of values greater than St	$Pr\{\|X\| \geq St\}$
Add the counts, divide by the number of values in the histogram	

To calculate the probability find the sort command in your package, then use it to put the randomized values that you have accumulated into a new (sorted) column. With only 100 values you can count values from the top down to the observed difference $St = +0.04286$, then count from the bottom up to the observed difference $St = -0.04286$. In packages with a spreadsheet each row is numbered, so you can note the row number of the value nearest *St*, to obtain the count in both tails.

Use your package to compute the following:

N = Number of randomized differences = _____

n_{neg} = Number of randomized differences more negative than –0.043 = _____

n_{pos} = Number of randomized differences more positive than +0.043 = _____

For future reference here is R-code to sort, display the sorted values, and give you a count of the values less than the observed difference (in k3, from above) and greater than the observed difference.

```
sort(RDvector)
length(RDvector)
k3                                  #Avdiff
summary(RDvector<(-abs(k3)))        #lower tail
summary(RDvector>(abs(k3)))         #upper tail
```

Your Name _____

Now compute the p-value for the observed outcome, a difference of –0.043 hours.

% of the outcomes less than the observed outcome of –0.043: $\frac{n_{neg}}{N} =$ _____

% of the outcomes greater than + 0.043: $\frac{n_{pos}}{N} =$ _____

% of the outcomes that were either greater than +0.043

or less than –0.043 $\frac{n_{neg} + n_{pos}}{N} =$ _____

This is your estimate of the Type I error (*p*-value) under the null hypothesis that the true difference is zero (a two-tailed test). The reason for computing a two-tailed test is that at the outset we had no idea whether *Strain1* would reproduce earlier or later than *Strain2*. Hence the need to calculate the probability of ***either*** a negative ***or*** a positive difference in timing.

It turns out that assumption free p-values by randomization in this example are close to the value (p = 0.908) calculated from the F-distribution with 1 and 12 degrees of freedom ($F_{1,12}$ = 0.014).

```
MTB>cdf F 0.014, 1, 12
pf(0.014, 1,12,lower.tail=FALSE)
```

The calculation of this F-ratio ($F_{1,12}$ = 0.014) will be explained later in the course, as will the assumptions for using the cdf command.

Attach your frequency distribution of randomized differences for the *Daphnia* data to this page. Label both axes of the frequency distribution.

Name _____

Write-up for lab #4.

1. Complete the previous page and submit annotated pdf file to Brightspace.
   Note: Adobe Reader can be used to fill in the blanks on the lab handouts.
   Adobe Reader is free to download.

2. Carry out a randomization test for data in Box 13.12 of Sokal and Rohlf (1995), which shows mean litter size of two strains of Guinea pig, compared over 9 years.

   Assign a symbol to your statistic (difference of means) _____

   State the observed value of your statistic: _____ = _____

   Report your assumption-free p-value via randomization _____

   Make a frequency distribution of at least 200 randomized outcomes, be sure to label both axes of your graph. Submit your graph as a separate pdf file under Lab 4 in Brightspace.

This lab illustrates some of the trade-offs in the choice of a statistical package.
1. Spreadsheet. This lab can be executed in a spreadsheet. However, the computational formula is never visible in a spreadsheet, except by selecting a cell, clicking on the formula in the cell, and following the components of the formula to other locations in the spreadsheet. The computational sequence is difficult to follow when formulas are linked across cells. Because we can't see the linkages, spreadsheets do not lend themselves well to understanding model based statistics, and so will not be used in the remainder of this course.
2. Statistical package with spreadsheet and pull-down menu. Model-based statistics are readily grasped in any statistical package with a high quality pull-down menu, such as Minitab, SPSS, SPlus, JMP. MUN recently stopped supporting Minitab, so help with SPSS will be provided in 2018. Model based statistics are readily grasped by3rd and 4th year undergraduates using any statistical package with a GLM routine, regardless of whether they choose to use a pull-down menu or code.

You don't have to use code, to grasp and use model based stats!

3. Statistical packages with readily programmable code. Minitab code for all the labs is available on the course website. It is shown in the labs because it is simple and easy to grasp. SAS code is available for some of the labs but MUN long ago ceased supporting it because of its expense. R-code is shown in the labs because of its efficiency, versatility, no cost as freeware, and rapidly expanding use by professional statisticians. Documentation of R-code is often incomprehensibly technical. Help on the web is abundant and uneven in quality. R-code has a steeper learning curve than the code in Minitab and SAS. SPSS has easily used pull-down menus for GLM and a wide variety of error structures for Generalized linear Models. SPSS code is useful for storing and re-running an analysis set up with the menu. Simple calculations with SPSS code (k3 = mean(Strain1)-mean(Strain2)) are difficult.