The National Animal Nutrition Program (NANP)
Modeling Committee

A National Research Support Project (NRSP-9)
Supported by the Experiment Station Committee on Organization and Policy,
The State Agricultural Experiment Stations, and Hatch Funds provided by the
National Institute of Food and Agriculture, U.S. Department of Agriculture

Installing and Using R

National Animal Nutrition Program
Leveraging Resources, Linking Researchers

2017

Table of Contents

A short introduction to R

R is a free software environment for statistical computing and graphics. R provides a new implementation of the S language, which was developed at Bell Laboratories

1. Installing R

The following instructions were written for the install of version 3.4.1 for Windows. The choices presented and links to click may be slightly different for other operating systems, and may change over time.

- If you do not already have R on your computer, the first step is to navigate to the CRAN (Comprehensive R Network Archive) website (https://cran.r-project.org/);
- Click the "Download R for *XXXXX*" link where *XXXXX* indicated the operating system running on your computer, e.g. Windows;
- On the next page click "Install R for the first time" which is listed after the base directory link;
- Select "Download R *N.N.N* for *XXXXX*" and "Run" where *N.N.N* should be the latest version available and *XXXXX* is again your operating system;
- Next, select the language to use during the installation, click Next, and Next again, and select a destination location, normally one would accept the default, e.g. "C:\Program Files\R\R-3.4.1".
- Click Next, and Next again, and for Startup options select "No (accept defaults)"
- Click Next, and Next. You should see a message indicating it is Installing, Completing the R for Windows 3.4.1 Setup Wizard. When completed click Finish.

1.1 Help and Resources

R has downloadable manuals (https://cran.r-project.org/manuals.html) devoted to the language, import/export procedures, and use of R statistical software. The Comprehensive R Archive Network (https://cran.r-project.org/) is a network of web servers around the world that store identical, up-to-date versions of code and documentation for R.

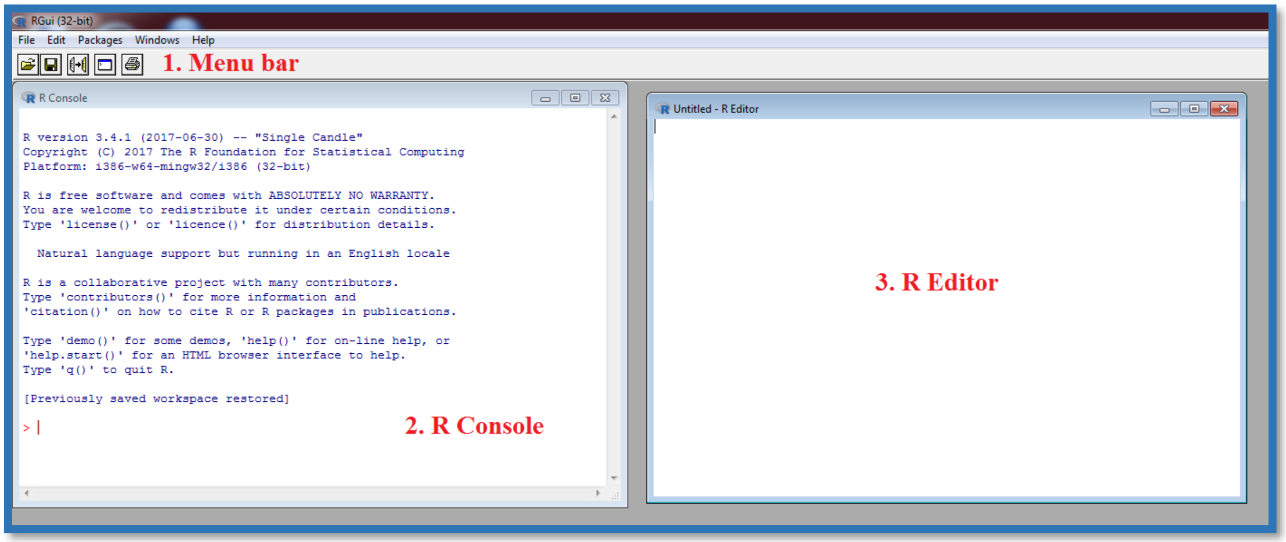
2. Installing R Studio

There are some additional features in R Studio that you may wish to take advantage of. To install R Studio, complete the following.

- To download Rstudio navigate to http://www.rstudio.com/products/rstudio/download/
- Click the RStudio Desktop Open Source License link;
- Installers for supported Platforms will be displayed;
- Click the "RStudio *Z.Z.Z – XXXXX*" where Z.Z.Z is the most recent version of R Studio, e.g. 1.0.143, and XXXXX is the appropriate operating system for your computer, e.g. Windows Vista/7/8/10.
- An executable installation file will be downloaded. You may be prompted to run it or you may need to click on the download to execute the installation.
- Select the desired locations for the install files and the shortcut, and click Finish when the install is completed.
- R Studio is a great tool to work with R, but the following instructions are for R.

3. Using R

- Open R and examine the interface:
  1. The menu bar: contains several menu items.
  2. R Console: You can enter commands in this window and output from the program will be shown here. Clicking enter after typing a command at the > submits the command. The “R Console” window is a command line prompt interface that can be used to run any R analysis. As no code is stored in the command line prompt interface, running analyses using just command line can be cumbersome. To store code used in an analysis, one must open a script.
  3. From the File Menu, select New Script. The script window (R Editor) allows code to be written, submitted for execution, and stored for later use. A set of instructions created in such a file is called a script. Anything written in this window can be saved as a “.R” file that can be opened by any R software program.



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

Figure 1. Layout of the R program

- To open an existing script select File > Open script.
- To submit code from the Script window, place your cursor on the line to be submitted and click the “Run line” icon on the menu bar or just press Ctrl+R.
- To submit multiple lines, highlight the desired lines and click the Run icon or press Ctrl+R.

4. Basic commends - R Console

- This symbol “ > ” is used to indicate that R is ready to receive a next code.
- Some incorrect input returns an error message such as: Error: unexpected input in "x".
- Use Ctrl+L to clear the R Console.
- Press Enter at the end of each command you type in the R Console. The R will return the result on the next line, prefaced by an index [1].
- Some basic mathematical operations in the R Console are shown in Table 1 and 2.

Table 1. R Mathematical operators and functions

Operator/Function	Code Example - R Console	
Addition	> 2 + 2 [1] 4	
Subtraction	> 2 - 2 [1] 0	
Multiplication	> 2 * 2 [1] 4	
Division	> 2 / 2 [1] 1	
Square-root of a value	> sqrt(4) [1] 2	
Natural Logarithm	> log(10) [1] 2.302585	
Exponentiation	> 2**3 [1] 8	> 2^3 [1] 8
Eulers number exponentiation	> exp(2) [1] 7.389056	

Table 2. Comparison operators in R

Operator/Function	Code Example - R Console
Equality	> 3 == 3 [1] TRUE
Inequality	> 3 != 3 [1] FALSE
Greater than	> 5 > 3 [1] TRUE
Less than	> 3 < 5 [1] TRUE
Greater than or equal to	> 3 >= 1 [1] TRUE
Less than or equal to	> 3 <= 3 [1] TRUE
Is within	> x <- c(2,4,6,8,10) > 8%in%x [1] TRUE

If	> x <- 2
	> x
	[1] 2
	> if(x==2) {
	+ "Hello World"
	+ } else {
	+ "Goodbye World"
	+ }
	[1] "Hello World">

Vector based if	> x <- c(1,2,3,4,5,6) > ifelse (x/1==1, "Hello World", "Goodbye World") [1] "Hello World" "Goodbye World" "Goodbye World" "Goodbye World" [5] "Goodbye World" "Goodbye World"

5. Syntax in R

- " <- " is used to denote an = sign, although in most cases an = sign will also work. Whereas 10*(2-3) returns a value, x = 10*(2-3) or x <- 10*(2-3) return: > |
  The object "x", then holds the value: > x
  [1] -10
  Almost any object can be assigned a name in R and often, assigning names is useful for analyses.
- File path names which specify a path with subdirectories must use a / to denote those directories rather than the normal \:
  - "C:\My Documents\Data\mydata.csv" becomes
  - "C:/My Documents/Data/mydata.csv"
- A # is used to denote a comment within the code. The line of code starting with # will be echoed to the console but will not be executed. For example, the code segment: "Hello World", when entered into R Console returns:
  ```
  > "Hello World"
  [1] "Hello World"
  ```
  R will ignore some code. To comment out code in R, preface the line of code with a "#". The statement (#"Hello World") will now return no output
  ```
  > # Hello World"
  >
  ```
- As noted above, the R console uses one color of text for code received and another color for outputted values returned. These colors may vary across operating systems and CRAN mirrors.
- A ">" symbol indicates that the console is reading the line as a new line of code, and a "+" symbol indicates that the current line being read was not completed on the previous line. For example, consider the statement: 10*(2-3), this returns:
  ```
  > 10*(2-3)
  [1] -10
  >
  ```
  Now, lets assume you forgot the ")" :
  ```
  > 10*(2-3
  +
  ```
  In this case, a ")" is needed to finish the line of code and send it through the processing console.
- In R, Press "Esc" to cancel any command input.

6. Vectors in R

- R is a vectorized language that is specialized for handling long sets of objects. To assign a vector we use the syntax:
  ```
  > y <- c(0,1,2,3,4,5,6,7,8,9,10)
  >
  ```
  where "c(" is an indication to R that the following string of objects, separated by commas, are going to make up the vector y. If we output y, we get:
  ```
   > y
  [1]  0  1  2  3  4  5  6  7  8  9 10
  ```

Mathematical operations on the elements in a vector can be performed based on a static value:

```
> y/2
 [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

- Simple vector in R:

  write this code in the R script x <- 1:10
  send it to R Console (highlight and Ctrl+R)
  write again the name (in our case x) to see data and press Enter

  ```
  > x <- 1:10
  > x
   [1]  1  2  3  4  5  6  7  8  9 10
  ```

  write this code in the R script y <- 2:12
  send to R Console (highlight and Ctrl+R)
  write again the name (in our case y) to see data and click Enter

  ```
  > y <- 2:11
  > y
   [1]  2  3  4  5  6  7  8  9 10 11
  ```

  display the second value

  ```
  > x[2]
  [1] 2
  > y[2]
  [1] 3
  ```

  mathematical operations on the elements in a vector

  ```
  > x/y
  [1] 0.5000000  0.6666667  0.7500000  0.8000000  0.8333333  0.8571429  0.8750000
  [8] 0.8888889  0.9000000  0.9090909
  ```

  statistics

  ```
  > max(x)
  [1] 10
  > sum(x)
  [1] 55
  > mean(x)
  [1] 5.5
  ```

7. R Data Import

Files can be read based on a full file path or from the document name (if a working directory is specified).

Table 3. If you have some local files for R to read

Description	Code	Example R Console
To see your current working directory	getwd()	> getwd() [1]"C:/Users/veridi7/Documents/Rdataset"
Set your new working directory	setwd ("C: ")	or go to the menu bar < File < Change working directory to
Import data from a Space Delimited file	data < read.table("Filename.txt", header=TRUE)	> data <- read.table("NRC2001.txt", header=TRUE) > data
Import data from a CSV file	data < read.csv("Filename.csv")	> data <- read.csv("NRC2001.csv") > data
Import data from a Excel file *Before, please install the package xlsx*	library (xlsx) data <-read.xlsx("Filename.xlsx", sheetName= "sheetname")	> data <- read.xlsx("NRC2001.xlsx", sheetIndex=1) > data
Comma Delimited Files	> data <- read.table("CommaDelimited.txt", sep=",", header=TRUE)	> data <- read.table("NRC2001.txt.txt", sep=",", header=TRUE)

Table 3.1 Frequent errors

Action	Error	Orientation
To change the working directory	Error in setwd("C:/Users/robin_white/Documents/Projects"): cannot change working directory	Translation: the path you specified does not exist, check the path name or create the folder you were attempting to reference.
	Error: '\U' used without hexdigits in character string starting ""C:\U"	Translation: you (like any sane, computer literate person) used backslashes in your file path but R would prefer that you either use double backslashes (“C:\\Users\\robin_white\\Documents”) or forward slashes (“C:/Users/robin_white/Documents”).
Space Delimited Files	Error in file(file, "rt") : cannot open the connection, In addition: Warning message: In file(file, "rt") : cannot open file 'SpaceDelimite.txt': No such file or directory	Translation: You spelled the filename wrong. Check filename and try again.
	No error is returned but data are uploaded as a string with commas separating numbers.	You have failed to specify that the values were separated by commas, add a “sep” term into your read.table statement.
	Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings, : line 8 did not have 7 elements	Translation: Your tab-delimited read.table() command called a file with missing values in line 8. This import format cannot handle missing values. Enter a NA into the space where the missing value should be and try again.
Importing CSV files	Error in file(file, "rt") : cannot open the connection. In addition: Warning message: In file(file, "rt") : cannot open file 'CSVExample.csv': No such file or directory	
Importing Data from Excel Files	Error in read.xlsx("XlsxFileExample.xlsx") : Please provide a sheet name OR a sheet index.	You need to specify a sheet to read either through the sheetIndex or sheetName arguments.
The xlsx package	Error in loadWorkbook(file) : Cannot find XlsxFileExampl.xlsx	You did not spell the file name right, check the filename/file path and try again.

8. Download Packages

Several packages are available on the Comprehensive R Archive Network (CRAN) at the R home page, *http://www.r-project.org/* . In your computer, you can check which packages are currently loaded using the function path.package() . If you need a new package, there are 2 ways to get it:

8.1. Loading a package from your hard drive

To install an R package

- Select “Packages” from the R menu bar > Set CRAN mirror > Secure CRAN mirrors (typically the nearest location; select 0 - cloud) > OK;
- Select “Packages” from the R menu bar > Install packages > select your “Package” from the alphabetical list of packages and Click OK;
- Check the Console messages to make sure if the package was downloaded;
- Any problems with the process ... highlight the error message and paste into Google with R at the end and you should find help. Another way to install the packages is to use the install.packages("nameofpackage") function.

8.2 Downloading a package from the website

- Go to https://cran.r-project.org/ and select packages;
- Table of available packages, sorted by name ;
- Available CRAN Packages By Name;
- Download and salve it (package.zip);
- install.packages(file_name_with_path, repos = NULL, type = "source") or go to the menu bar > Packages> Install the packages from local files

Once a package has been installed, you do not need to reinstall it. Each time that you need to use the package, before use it you have to load the package into R using the function: library(*package name*)

9. Data frames

Data tables are stored as two primary types in R, matrices and data frames. For most animal nutrition applications, data frames are the more useful utility. A data frame is a collection of vectors. Each column in the data frame represents a named vector of data. The data frame can contain a mix of vector data types, i.e. one vector can be text, another integers, and another real numbers. Each vector can be referred to in its entirety by the name of the data frame followed by the name of the vector separated by a $ sign: dfname$vectorname

9.1 Creating a basic data frame - R Console

```
> # Create 2 vectors of data
> animal <- c("cow1","cow2","cow3")
> production <- c(32,40,20)
> # Create a dataframe containing the 2 vectors
> d <- data.frame(animal, production,stringsAsFactors=FALSE)
```

9.2 Accessing the Data frame

a. The entire data frame:

```
> d
  animal production
1   cow1        32
2   cow2        40
3   cow3        20
>
```

b. Only the first column.

```
> d[,1] or d$animal
[1] "cow1"  "cow2"  "cow3"
```

c. Basic statistics on the data frame.

```
> summary(d$production)
   Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
  20.00   26.00   32.00    30.67   36.00   40.00

> summary(d$animal)
   Length     Class       Mode
        3           character  character
```

d. We also can refer to the column using [, column number]. For example, the production is in column 2 , thus:

```
> head( d [,2] )
[1] 32 40 20
```

e. Finding data

```
> which(d$production <= 39)
[1] 1 3

> d [1,]
        animal  production
1       cow1        32

> d [3,]
        animal   production
3       cow3        20
```

10. Working with data frames

10.1 Create a data frame containing 2 vectors (y and x)

```
> y <- c(0,1,2,3,4,5,6,7,8,9,10)
> x <- c(2,4,6,8,10,12,14,16,18,20,22)
> data <- data.frame(y, x)
> data
    y  x
1   0  2
2   1  4
3   2  6
4   3  8
5   4 10
6   5 12
7   6 14
8   7 16
9   8 18
10  9 20
11 10 22
```

First calling the data frame name and then identifying the column name. The two items must be separated by "$":

```
> data$x
 [1]  2  4  6  8 10 12 14 16 18 20 22
```

Mathematical operations on data within an R data frame can be run based on calling the columns directly:

```
> data$y/data$x
 [1] 0.0000000 0.2500000 0.3333333 0.3750000 0.4000000 0.4166667 0.4285714
 [8] 0.4375000 0.4444444 0.4500000 0.4545455
```

General mathematical operations using 2 vectors (a and b)

```
> a <- c(0,1,2,3,4,5,6)
> b <- c(2,4,6,8,10,12,14)
```

Mathematical operations

```
> c <- c(a+1)
> d <- c(a+b)
> e <- c(a-b)
> f <- c(a*b)
> g <- c(a/b)
> data <- data.frame(a,b,c,d,e,f,g)
> data
  a  b c  d  e  f         g
1 0  2 1  2 -2  0 0.0000000
2 1  4 2  5 -3  4 0.2500000
3 2  6 3  8 -4 12 0.3333333
4 3  8 4 11 -5 24 0.3750000
5 4 10 5 14 -6 40 0.4000000
6 5 12 6 17 -7 60 0.4166667
7 6 14 7 20 -8 84 0.4285714
```

10.2 Add a new row and data to an existing data frame

Table 4. We can use rbind() to add a new row to an existing data frame

Dataset 1	Dataset 1 + new row

```
> d
  animal production    breed
1   cow1        32     Holstein
2   cow2        40     Jersey
3   cow3        20     Holstein
```
	```
	> rbind(d,list("cow4",19, "Jersey"))
	  animal production    breed
	1   cow1        32     Holstein
	2   cow2        40     Jersey
	3   cow3        20     Holstein
	4   cow4        19     Jersey
	>
	```

10.3 Merging dataset by row

Merging the rows of two data frames together can be done with the rbind command.
#Merge the rows of two data columns, say you have two data frames from two different trials:

```
data1 <- read.table("TabDelimited.txt")
data2 <- read.table("TabDelimited.txt")
merged <- rbind(data1, data2)
```

When merging rows of data using the rbind command, the names of columns and the number of columns must match exactly.

Frequent Errors

a. Error in rbind(data1, dat) : object 'dat' not found
   R does not recognize the name of one of the data frames. Check the name spelling and make sure the data frame exists in R.
b. Error in rbind(deparse.level, ...) : numbers of columns of arguments do not match
   The number of columns does not match between the data frames. Use names() or summary() commands to check what the variables in each data frame are. Adjust the dataframes and continue.

Table 5. We can use rbind to bind rows from different data frames. The names and number of columns should be similar

Dataset 1	Dataset 2	New dataset (merged)

```
> d1 <- data.frame(a = 1:5,b =
1:5)
> d1
  a b
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
```
	```
	> d2 <- data.frame(a = 6:10,b =
	6:10)
	> d2
	   a  b
	1  6  6
	2  7  7
	3  8  8
	4  9  9
	5 10 10
	```
		```
		> merged <- rbind(d1,
		d2)
		> merged
		    a  b
		1   1  1
		2   2  2
		3   3  3
		4   4  4
		5   5  5
		6   6  6
		7   7  7
		8   8  8
		9   9  9
		10 10 10
		```

10.4 Merging data by column

To merge columns of two data frames by an identifier variable, you will want to use the merge statement from the plys package.

```
#Merge the columns of two data frames by a common variable:
library(plyr)
data1 <- read.table("TabDelimited.txt")
data2 <- read.table("TabDelimited.txt")
merged <- merge(data1, data2, by=c("Paper", "Treatment"))
```

Note that unlike SAS, R is smart enough to merge data that are not sorted first.
Frequent Errors

a. Error in fix.by(by.x, x) : 'by' must specify a uniquely valid column
   You have misspelled the name of a by variable. Alternatively, the by-variable you specified is not unique within at least one of the data frames.

b. Error in merge(data1, data2, by = c("Paper", "Treatment")) : object 'data1' not found
   R cannot find one of the data frames you referenced. Checked the spelling of the data fame and make sure it exists in the R environment.

Table 6. We can merge the data frames x and y

Dataset x	Dataset y	New dataset

```
> x <- data.frame(animals
= 1:5, number= 6:10)
> x
  animals number
1       1      6
2       2      7
3       3      8
4       4      9
5       5     10
```
	```
	> y <- data.frame(ages = 3:7,
	animals = 1:5)
	> y
	  ages animals
	1    3       1
	2    4       2
	3    5       3
	4    6       4
	5    7       5
	```
		<pre>
		> merged <- merge(x,y)
		> merged
		  animals number ages
		1 1 6 3
		2 2 7 4
		3 3 8 5
		4 4 9 6
		5 5 10 7</pre>

10.5 Add a new column to an existing data frame

Table 7. We can use cbind to add columns to an existing data frame

Dataset 1	Dataset 2	Merged (columns dataset 1 and 2)

```
> d1 <- data.frame(a =
1:5,b = 6:10)
> d1
  a  b
1 1  6
2 2  7
3 3  8
4 4  9
5 5 10
```
	```
	> d2 <- data.frame(c = 11:15,d =
	16:20)
	> d2
	   c  d
	1 11 16
	2 12 17
	3 13 18
	4 14 19
	5 15 20
	```
		```
		> d12 <-cbind(d1,d2)
		> d12
		  a  b  c  d
		1 1  6 11 16
		2 2  7 12 17
		3 3  8 13 18
		4 4  9 14 19
		5 5 10 15 20
		>
		```

10.6 Sorting

```
> #  Sorting
> x <- c(47,1,23,7,24)
> sort(x)
[1]  1  7 23 24 47
```

10.7 Order

a. Order a vector

If you are using a vector, then use the order() function.

```
> x <- c(47,1,23,7,24)
> order(x)
[1] 2 4 3 5 1  # position
```

This means that x=2 is the smallest value in x, x=4 is the second smallest, and so on.

b. Order by column

In our data base (x) we will use the order() function on the second column (age). Consequently, we will yield another data frame (x1) where the dataset is organized by age.

Table 8. Order by age

Dataset x	
> x <- data.frame(animal = 1:5, age = 7:3,	> x1 <- x[order(x$age),]
bw = 650:654)	> x1
> x	animal age bw
animal age bw	5 5 3 654
1 1 7 650	4 4 4 653
2 2 6 651	3 3 5 652
3 3 5 652	2 2 6 651
4 4 4 653	1 1 7 650
5 5 3 654	>

Data frames can be ordered by the bracketed order command:

```
# Order data within a dataframe
data <- read.table("TabDelimited.txt")
ordered <- data[order(data$Intake),]
```

Note that a comma must be added after the order statement.

Frequent Errors

- Error in `[.data.frame`(data, order(data$Intake)) : undefined columns selected
  You misspelled the name of the ordering column or you failed to include a comma after the order statement.

10.8 Melting or Casting Data in R

Melting is used to convert wide-format data to long-format data. Casting is used for the opposite process. When melting or casting, you want to use some identification variables. In meta-analyses, treatment and publication IDs are often our variables of interest.

The reshape2 package is needed to melt or cast data frames. To melt a dataset by publication and treatment ID:

```
#Melting and Casting Data in R
library(reshape2)
data <- read.table("TabDelimited.txt")
melted <- melt(data, id=c("Paper", "Treatment"))
```

The returned data frame has four columns, Paper, Treatment, Variable and Value. The Variable contains the previous column names and the value holds the numerical value associated with the variable, paper and treatment combination. This data is now in long format. Once the data is in long format, you can cast it into differing formats or apply various functions by groups within the data. For example, you can take the mean of the response variables reported in each study:

```
#Melting and Casting Data in R
library(reshape2)
data <- read.table("TabDelimited.txt")
melted <- melt(data, id=c("Paper", "Treatment"))
means <- dcast(melted, Paper ~ variable, fun.aggregate=mean)
```

Or you can sum up the response variables included within each study:

```
means <- dcast(melted, Paper ~ variable, fun.aggregate=sum)
```

When dealing with feed data, it is useful to understand these functions as they allow you to calculate important parameters like the quantity of a nutrient provided by the diet given the diet feed composition information and inclusion rates.

10.9 Join in R

a) Full outer join (all rows from both data frames, specify all=True)
```
# outer join
z <- merge(x, y, all=TRUE)
View(z)
```
b) Inner join (only rows that match from the data frames)
```
# inner join
merge <- merge(x, y)
View(merge)
```
c) Left outer join (to include all the rows of your data frame x, and only those from y that match, specify all.x=True)
```
  # left join com R base
left.join.r.base <- merge(x, y, all.x = TRUE)
```
d) Right outer join (to include all the rows of your data frame y, and only those from x that match, specify all.y=True)
```
right.join.r.base <- merge(x, y, all.y = TRUE)
```

10.10 Exporting R data

```
# Exporting CSV
write.csv(dataname, file = "dataname.csv")
```

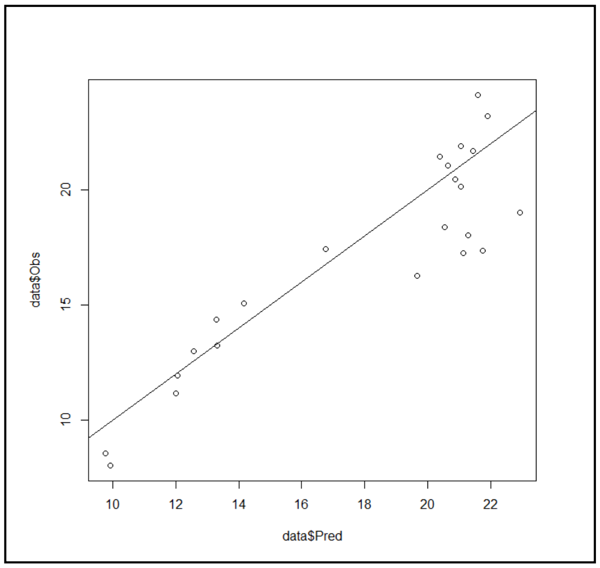
Example:

```
> getwd()
[1] "C:/Users/veridi7/Documents"
> setwd ("C:/Users/veridi7/Documents/RWork")
> library(readxl)
> NANP <- read_excel("~/RWork/NANP.xlsx", sheet = "Studies")
> View(NANP)
> write.csv(NANP, file = "NANPt.csv")
>
```

Open the CSV file in your current working directory, in our example RWork.

11. Data visualization tools

```
> data = read.csv(file = "Lesson 4 exercise data file.csv")
> names(data) = c("Obs", "Pred")
> data
     Obs  Pred
1   8.03  9.90
2   8.56  9.75
3  16.26 19.68
4  17.25 21.13
5  17.38 21.76
6  20.14 21.05
7  12.99 12.57
8  11.18 11.99
9  14.37 13.28
10 15.09 14.17
11 18.03 21.28
12 19.01 22.93
13 11.96 12.04
14 13.24 13.32
15 18.37 20.55
16 17.42 16.76
17 24.14 21.60
18 21.06 20.64
19 21.46 20.38
20 20.47 20.88
21 21.90 21.05
22 21.68 21.45
23 23.22 21.91
> #Always plot your data first to get a feel for the data
> plot(data$Pred, data$Obs, type='p')
> abline (0, 1)
```



Reference List

The R Core Team (2017). R: A Language and Environment for Statistical Computing. Version 3.4.1. R Foundation for Statistical Computing, URL https://cran.r-project.org/doc/manuals/r-release/fullrefman.pdf

J.F. Wiley, L.A. Pace 2015. Beginning R. An Introduction to Statistical Programming. ISBN: 978-1-4842-0374-3 (Print) 978-1-4842-0373-6 (Online). DOI 10.1007/978-1-4842-0373-6.

N. Matloff 2011. The art of R programming: A tour of statistical software design. No Starch Press; 1 edition. ISBN-10: 1593273843.

Document prepared by Veridiana L Daley