**The University of British Columbia**

**Wildlife Institute of India**

**R Workshop**

***Course Handout***

> storks=c(2,6,3,6,2,8)

> babies=c(6,8,5,9,4,9)

> par(mfrow=c(1,1),mai=c(1.4, 1.4, 0.9, 0.9),cex=1.5)

> plot(storks, babies)

**February 25 and 26, 2013**

**CAWP 2942**

**Instructors:**

**Richard Schuster, PhD candidate**

**Prof. Valerie LeMay**

# Table of Contents

[Table of Contents ii](#_Toc349239432)

[Workshop Schedule 1](#_Toc349239433)

[DAY 1: Introduction to R 2](#_Toc349239434)

[Topic 1: Introductions and Logistics for the 2-day workshop 2](#_Toc349239435)

[Topic 2: The basics of R 4](#_Toc349239436)

[Topic 3: Manipulating Data 7](#_Toc349239437)

[Exercise: Practice Using R to Manipulate Data 7](#_Toc349239438)

[DAY 2: Graphing and Fitting Models Using R 18](#_Toc349239439)

[Topic 4: Graphs 18](#_Toc349239440)

[Exercise: Graphs Using R 18](#_Toc349239441)

[Topic 5: Linear regression using least squares 23](#_Toc349239442)

[Exercise: Basic Statistics and Regression Analysis Using R 23](#_Toc349239443)

[Exercise: Multiple Linear Regression Using R 35](#_Toc349239444)

[Using R and Stepwise Methods to Select Predictor Variables in a Regression Model 39](#_Toc349239445)

[Topic 6: Generalized Linear Models 43](#_Toc349239446)

[Example Logistic Regression (y-variable is binomial): Species presence. 48](#_Toc349239447)

[Example: The y-variable is count, with or without extra zeros: Bird counts. 57](#_Toc349239448)

[Another example of application of a generalized linear model: Large Mammals 62](#_Toc349239449)

[R Books 64](#_Toc349239450)

# Workshop Schedule

|  |
| --- |
| Day 1, February 25: Introduction to R, 1500 to 1700 |
| Topic 1: Introduction and logistics for the 2-day workshop  Topic 2: The basics of R. What is it, how does it work.  Topic 3: Manipulating data |
|  |
|  |
| Day 2, February 25: Graphing and Fitting Models, 1200 (noon) to 1600 |
| Topic 4: Graphs  Topic 5: Linear models using least squares  Topic 6: Generalized linear models: Introduction using examples |
|  |

# DAY 1: Introduction to R

## Topic 1: Introductions and Logistics for the 2-day workshop

*Logistics*

* Instructors
* Name tags and finding a computer
* Washrooms
* Safety Issues
  + Exits
  + Earthquake response
  + Telephone in lobby of Forest Sciences building
* No food or liquids in the computer labs, PLEASE!
* Breaks

*Course Content*

* Day 1: Using R: General introduction to R and to manipulating data using R
* Day 2: Graphing and fitting models

**Computer Login: ­­­­­­­­­­­­­­­­wii2013**

**Password: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Log in with the login id and the password provided by the instructor
2. Open Windows Explorer (Windows Key+e)
3. Navigate in the left panel of Windows Explorer to Courses:\WII Visitors
4. Find the sub-folder for the R workshop, Right click on this, and then Ctrl + C to copy these files
5. Right mouse click on C:\temp in the left panel of Windows Explorer and then Ctrl + V to paste a copy of the workshop materials into a temporary folder.

## Topic 2: The basics of R

**(These notes were summarized for classroom presentation along with some short R exercises.)**

*Background:*

R is a free software package that has been designed to analyze and graph data. A collection of people worldwide have developed libraries of functions that you can use to analyze data. Because this is freeware, there may be “bugs” in the software. However, many of the library functions have now been tested by many users, and compared to other commercially available software packages such as SAS and SPSS.

*Installing R on your home or laptop computer:*

To run R, first you need to load the software. Generally, to load the software, you can go to R website directly. There are instructions for downloading software on the site. You will need to find a Cran near to your location, for faster service. A list of these can be found on the R website. This loads the standard package, with a number of libraries.

Here are the steps you would follow:

1. Go to the R website: <http://www.r-project.org/>
2. Select CRAN, which is under Download, on the left hand side of the screen
3. From the list (centre and right side of the screen), select the closest location
4. Then, select Download and Install R, and select Windows
5. Under Subdirectories, select Base
6. R-2.7.0-win32.exe [or whatever the latest version of R is]

You will be asked if you want to run or to save this file. If you are ready to install the package, click on Run.

You will then be prompted for where you wish to save the file (pick a simple path on you home or laptop computer e.g. C:/R ). Use the default settings.

When you are done, there should be the R icon on your desktop and R should be ready for your use.

*Running R:*

After you install R for Windows, you will have an R icon on your desktop. To run R, click on the icon . You get a work session window. You could type in your commands here and they would run as you enter them. Instead, you can enter your commands into a separate file called *script*. The script is just R commands, organized and put into a text file. Then, you can open your script file while you are running R. You will get a separate window with the script. You can then run this all at once (only if you are very confident), or in segments (preferred).

To run the R commands in the session window, simple press Enter on your keyboard.

To run the R commands in your script, simply highlight the parts you want to run and press Ctrl and R at the same time OR, click on Edit and find Run in the list.

The outputs from running your commands will also appear in the session window. You can also highlight any part of the session window you wish to keep, and then press Ctrl and C at the same time to copy this, and then open a Word file and copy this in using Ctrl and V at the same time.

If part of your script involves getting a graph, another window opens up if you get a graph using R. If you run another graph, the original graph disappears and you get the new graph.

*Useful things to know:*

* R is case sensitive. This means that the variable Trees is not the same as the variable trees for example.
* R does not like spaces nor special characters. Instead, use a ‘.’ For example, trees.pine identifies a variable.
* R uses two slashes instead of one to indicate a subfolder. For example, if your data in Windows are in: E:\measurements\trees.txt then in R you would use E:\\measurements\\trees.txt since the single slash has a different meaning in R
* Any R commands that start with # are just *comments* that you can add to explain what the script does.

*Saving and cleaning up:*

You can save the work session, the script, and the graph window anytime you wish by using File and Save for the session window when that window is active, or to save the script when the script window is active, or to save the graph when the graph window is active.

To save the objects you have created, you can use File and Save Workspace to save all objects (ie all the data and outputs you have put into objects). You can then later on use File and Load Workspace to bring the objects back in and continue your work. You can also cut and paste any of the outputs from your session window into WORD or other files.

If you have created a graph, you can use File and Copy to Clipboard and then As a Metafile (gives a better graph then as a Bitmap). You can also save the file as a picture using File and Save As..

At any time, you can use Edit and Clear Console to clean out the session window. However, the data you brought in, and any variables and objects wil still be there. To remove all of these, click on Misc and then select Remove all objects, OR type rm(list=ls(all=TRUE)) in the session window. When you begin an R session with new data, it is always a good idea to start with no objects.

*Graphs:*

R is very good at graphs. The main way to make a graph is to use the function plot( ), where there are a number of arguments in the brackets (i.e., the x variable, the y variable, labels, type of graph, etc. However, only one graph appears at once, in a separate window. When you graph in R, you should save the graph (e.g., as a .jpg file or as a metafile), before moving to the next graph. R can also do multiple graphs in the same graph window.

*Types of Objects (more on this later in the course):*

Factor: a class variable, often represented as letters but maybe represented as numbers

Vector: a “column” of numbers

Matrix: several columns of numbers

Dataframe: Like a matrix, but can have columns of numbers and columns of letters

List: Can be several objects all stored together such as regression outputs, matrices, etc.

*Help:*

The R website has a number of manuals that you might find useful, including an introduction: <http://cran.r-project.org/doc/manuals/R-intro.pdf> and <http://cran.r-project.org/doc/contrib/usingR.pdf> There are also a number of very useful books published by Springer, and Chapman and Hall publishers that are very useful. Dr. Andrew Robinson, University of Melbourne, Melbourne, Australia, produced an introduction to R that is very useful and can be found at: <http://www.ms.unimelb.edu.au/~andrewpr/r-users/icebreakeR.pdf>

At any time, you can also use help( ) where the function is given in the brackets. This help is a bit hard to follow, and is really meant to tell you the specific options for a function. However, there are also a few examples with the help that you might find useful as you are using R.

*Expanding the R package:* When you run R, only some of the functions are brought into the work session automatically to save memory. To add others, you can use require( ) where the package is given in brackets. Also, there are many other parts of R that are extra to the main package. To bring these in, you will need to access the website and get the software package. This then can be downloaded to the R directory in a sub-folder under library. For example, if you installed R in: C:\Program Files\R\R-2.8.0\, then you can add more software into C:\Program Files\R\R-2.8.0\library\ You can then use library( ) to bring in these other packages for your analysis.

*Learning R:*

Many people have put documentation and examples using R code or script on the web. Also, Springer book publishers and Chapman Hall/CRC book publishers have published many books on using R in the last 2 years (See list of e-books at UBC Library included in these notes). Examples are very helpful for reducing the time you spend in getting R to do what you would like. However, the best way to learn R is really to use it. The course materials provided by Dr. Andrew Robinson are excellent to help you practice and learn R and become more comfortable with using it for your analyses. The exercises provided here are very brief and just give you a taste of using R for forestry problems.

## Topic 3: Manipulating Data

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

### Exercise: Practice Using R to Manipulate Data

*R code:* 03.Data.Manipulation.r

###################################################

### Creation and Assignment

###################################################

a <- 1 # Create an object "a" and

# assign to it the value 1.

a <- 1.5 # Wipe out the 1 and make it 1.5 instead.

class(a) # What class is it?

class(a) <- "character" # Make it a character

class(a) # What class is it now?

a

a <- "Richard" # Wipe out the 1.5 and make it "Richard" instead.

b <- a # Create an object "b" and assign to it

# whatever is in object a.

a <- c(1,2,3) # Wipe out the "Richard" and make it a vector

# with the values 1, 2, and 3.

# Never make c an object!

b <- c(1:3) # Wipe out the "Richard" and make it a vector

# with the values 1, 2, and 3.

b <- mean(a) # Assign the mean of the object a to the object b.

ls() # List all user-created objects

rm(b) # Remove b

###################################################

### Numeric

###################################################

a <- 2 # create variable a, assign the number 2 to it.

class(a) # what is it?

is.numeric(a) # is it a number?

b <- 4 # create variable b, assign the number 4 to it.

(a + b) ^ 3 # basic math

a == b # test of equality (returns a logical)

a < b # comparison (returns a logical)

max(a,b) # largest

min(a,b) # smallest

###################################################

### String

###################################################

a <- "string" # create variable a, assign the value "string" to it.

class(a) # what is it?

is.numeric(a) # is it a number?

is.character(a) # is it a string?

b <- "spaghetti" # create variable b, assign the value "spaghetti" to it.

paste(a, b) # join the strings

paste(a, b, sep="") # join the strings with no gap

d <- paste(a, b, sep="")

substr(d, 1, 4) # subset the string

###################################################

### Factor

###################################################

a <- c("A","B","A","B") # create vector a

class(a) # what is it?

is.character(a) # is it a string?

is.factor(a) # is it a factor?

a <- factor(a) # make it so

levels(a) # what are the levels?

table(a) # what are the counts?

a <- factor(c("A","B","A","B"), levels=c("B","A"))

# create a factor with different levels

###################################################

### Logical

###################################################

a <- 2 # create variable a, assign the number 2 to it.

b <- 4 # create variable b, assign the number 4 to it.

d <- a < b # comparison

class(d) # what is it?

e <- TRUE # create variable e, assign the value TRUE to it.

d + e # what should this do?

d & e # d AND e is True

d | e # d OR e is also True

d & !e # d AND (NOT e) is not True

###################################################

### Missing Data

###################################################

a <- NA # assign NA to variable A

is.na(a) # is it missing?

class(a) # what is it?

a <- c(11,NA,13) # now try a vector

mean(a) # agh!

mean(a, na.rm=TRUE) # Phew! We've removed the missing value

is.na(a) # is it missing?

###################################################

### Vector

###################################################

a <- c(11,12,13) # a is a vector

a[1] # the first object in a

a[2] # the second object in a

a[-2] # a, but without the second object

a[c(2,3,1)] # a, but in a different order

a + 1 # Add 1 to all the elements of a

length(a) # the number of units in the vector a

order(c(a,b)) # return the indices of a and b in increasing order

c(a,b)[order(c(a,b))] # return a and b in increasing order

a <- c(11,NA,13) # a is still a vector

a[!is.na(a)] # what are the elements of a that aren't missing?

which(!is.na(a)) # what are the locations of the non-missing elements of a?

###################################################

### Exercise

###################################################

# Create a vector d with elements 12,5,NA,6,9,2,NA,17

# Remove NA's and save results in vector e

# Extract all values of e that are bigger than 5

###################################################

###################################################

### Vectorization

###################################################

diameters <- rgamma(n=1000000, shape=2, scale=20)

basal.areas <- rep(NA, length(diameters))

system.time(

for (i in 1:length(diameters)) {

basal.areas[i] <- diameters[i]^2 \* pi / 40000

}

)

system.time(

basal.areas <- diameters^2 \* pi / 40000

)

###################################################

### Dataframe

###################################################

setwd("C:\\temp\\")

ufc <- read.csv("ufc.csv") # ufc is a dataframe

is.data.frame(ufc) # we hope

dim(ufc) # the size of the dimensions (r,c)

names(ufc) # the labels of the columns

ufc$height[1:5] # first 10 heights

ufc$species[1:5] # first 10 species

ufc[1:5, c(3,5)] # first 5 species and heights

ufc[1:5, c("species","height")] # first 5 species and heights again

table(ufc$species)

###################################################

### Drop NA's

###################################################

ufc <- ufc[ufc$species != "",]

ufc$species <- factor(ufc$species)

table(ufc$species)

###################################################

### Create new variables

###################################################

ufc$dbh.cm <- ufc$dbh/10 # Dbh now in cm

ufc$height.m <- ufc$height/10 # Height now in metres

str(ufc)

###################################################

### Create new Dataframe with subset

###################################################

temp <- data.frame(my.species=ufc$species,

my.dbh=ufc$dbh.cm)

temp[1:5,]

###################################################

### Extracting data

###################################################

ufc$height.m[ufc$species=="LP"] # Heights of lodgepole pine

mean(ufc$height.m[ufc$species=="LP"], na.rm=TRUE)

# Average height of lodgepole pine

###################################################

### Sapply

###################################################

sapply(ufc[,4:7], mean, na.rm=TRUE)

sapply(ufc, class)

###################################################

### provides the indices of the ob-servations in order of decreasing height

###################################################

ufc$species[order(ufc$height.m, decreasing = TRUE)][1:3]

###################################################

### tapply

###################################################

tapply(ufc$height.m, ufc$species, mean) # Average height by species

tapply(ufc$height.m, ufc$species, mean, na.rm=TRUE) # Average height by species

format(tapply(ufc$height.m, ufc$species, mean, na.rm=TRUE), dig=3)

##################################################

### Exercise

##################################################

# What are the mean diameters by species?

# What are the three species with the largest median slenderness (height/diameter)

# ratios?

##################################################

###################################################

### How would we pull out the identity of the median height tree of the species

### that was eighth tallest on average?

###################################################

(ht.bar.by.species <- tapply(ufc$height.m, ufc$species, mean, na.rm=TRUE))

(species.order.by.ht <- order(ht.bar.by.species, decreasing = TRUE))

(species.by.ht <- levels(ufc$species)[species.order.by.ht])

(sp.8 <- species.by.ht[8])

(m.ht.8 <- median(ufc$height.m[ufc$species == sp.8], na.rm=TRUE))

ufc[which(ufc$height.m==m.ht.8 & ufc$species == sp.8),]

# Single operation

ufc[which(ufc$height.m==median(ufc$height.m[ufc$species ==

levels(ufc$species)[order(tapply(ufc$height.m, ufc$species, mean,

na.rm = TRUE), decreasing = TRUE)][8]], na.rm = TRUE) &

ufc$species==levels(ufc$species)[order(tapply(ufc$height.m,

ufc$species, mean, na.rm = TRUE), decreasing = TRUE)][8]),]

##################################################

### Exercise

##################################################

# What is the identity of the tallest tree of the species that was the

# fattest on average?

##################################################

###################################################

### Matrices

###################################################

(mat.1 <- matrix(c(1,0,1,1), nrow=2))

(mat.2 <- matrix(c(1,1,0,1), nrow=2))

solve(mat.1) # This inverts the matrix

mat.1 %\*% mat.2 # Matrix multiplication

mat.1 + mat.2 # Matrix addition

t(mat.1) # Matrix transposition

det(mat.1) # Matrix determinant

apply(ufc[,4:7], 2, mean, na.rm=TRUE)

###################################################

### Lists

###################################################

(my.list <- list("one", TRUE, 3))

my.list[[2]]

my.list[2]

(my.list <- list(first = "one", second = TRUE, third = 3))

names(my.list)

my.list$second

names(my.list) <- c("First element","Second element","Third element")

my.list

my.list$`Second element`

###################################################

### Merging

###################################################

(params <- data.frame(species = c("WP", "WL"),

b0 = c(32.516, 85.150),

b1 = c(0.01181, 0.00841)))

(trees <- ufc[ufc$species %in% params$species & !is.na(ufc$height.m),][1:3,])

(trees <- merge(trees, params))

(trees$volume <-

with(trees, b0 + b1 \* (dbh.cm/2.54)^2 \* (height.m\*3.281))\*0.002359737)

###################################################

### Reshaping

###################################################

(trees <- data.frame(tree = c(1, 2), species = c("WH", "WL"),

dbh.1 = c(45, 52), dbh.2 = c(50, 55),

ht.1 = c(30, 35), ht.2 = c(32, 36)))

(trees.long <- reshape(trees,

direction = "long",

varying = list(c("dbh.1","dbh.2"),

c("ht.1","ht.2")),

v.names = c("dbh","height"),

timevar = "time",

idvar = "tree"

))

# direction tells R to go wide or go long,

# varying is a list of vectors of column names that are to be stacked,

# v.names is a vector of the new names for the stacked columns,

# timevar is the name of the new column that differentiates between successive

# measurements on each object, and

# idvar is the name of the existing column that differentiates between

# the objects.

###################################################

### Sorting

###################################################

ufc[order(ufc$height.m, decreasing=TRUE),][1:5,]

ufc[order(ufc$plot, ufc$species, ufc$height.m),][1:5,]

# 

# DAY 2: Graphing and Fitting Models Using R

## Topic 4: Graphs

### Exercise: Graphs Using R

*Background:*

R has some very useful graphics functions. These can be very helpful for conveying information to audiences in presentations and papers.

*Files:*

We will use the tree data found in trees.txt for this exercise. There are 250 *Populus* trees and 250 *Abies* trees in this dataset. We will run some simple plots to visualize this fairly large dataset. The script can be found in graphs.R.

*Running the R Script:*

1. Start R
2. Use File and Open script to bring in the graphs.R script.
3. For graphs, a number of lines of the R script must be run together, to set up the graph, and then add data to the graph. These lines of R script are separated by blank lines. Run the R Script in parts by: 1) highlighting a part of the script and then 2) Using Ctrl+R to run the script.
4. As you run the script in parts, write down what each section does.
5. Also, click on the graph window and then File and Save As to save one or more of your graphs.

*For discussion:*

Which plot(s) did you find useful in visually describing these data?

################################################################################

# bring in data from an outside tab delimited text file, can be converted from a EXCEL file

# (shown in this example)

#

################################################################################r

rm(list=ls(all=TRUE)) # removes anything that might be restored from previously running R

trees<- read.table("E:\\R\_materials\_workshops\\workshop\_2013\\data\\trees.txt",header=TRUE)

attach(trees)

names(trees)

dim(trees)

# change species name to a class variable, rather than a number #

speciesname<-factor(trees$species)

trees2<- data.frame(trees,speciesname)

detach(trees)

rm(trees,speciesname)

ls()

attach(trees2)

names(trees2)

# get a simple summary of all variables in dataframe trees2 #

summary(trees2)

################################################################################

# get simple scatterplots for volume with all variables in trees2

# notice that the plot changes to a box plot for the class variable,

# speciesname (shown in this example)

################################################################################

plot(volume~.,data=trees2)

rank<- order(dbh) # get the order of trees by dbh

sortedtrees<- trees2[rank,] # use the order of trees to order the data

plot(volume~dbh,type="p") # points

plot(volume~dbh,type="l",data=sortedtrees) # lines -- data must be sorted by x

plot(volume~dbh,type="b",data=sortedtrees) # both -- data must be sorted by x

plot(volume~dbh,type="h",data=sortedtrees) # histogram

##########################################################################

# normality plots and histograms

#########################################################################

qqnorm(dbh) # normality plot

qqline(dbh,col=2)

qqnorm(height) # normality plot

qqline(height,col=2)

hist(dbh, breaks =20 , density=10,col="green", border="black") # draws a histogram

hist(height, breaks =10 , density=10,col="black", border="black") # draws a histogram

################################################################################

# plots from regression -- standard set. NOTE: you will need to click on the

# graph window to bring up each plot (i.e., "Waiting to confirm page change..."

################################################################################

model1<-lm(volume~height,data=trees2)

plot(model1)

##########################################################################

# Box plots: volume and logarithm of volume

#########################################################################

boxplot(volume~speciesname,data=trees2,col='pink')

boxplot(volume~speciesname,data=trees2,log="y",col='pink')

##########################################################################

# Multiple plots

#########################################################################

require(lattice)

xyplot(volume~height+dbh,data=trees2,scales="free",groups=speciesname,auto.key=TRUE,layout=c(1,2),corner=c(0,0))

##########################################################################

# Another way to get multiple plots: 2 box plots, one graphics image

#########################################################################

par(mfrow=c(1,2),mai=c(1.4, 1.4, 0.2, 0.2),cex=1.5)

boxplot(volume~speciesname,data=trees2,col='pink')

boxplot(volume~speciesname,data=trees2,log="y",col='pink')

par (mfrow=c(1,1), mai=c(1.0,1.0,1.0,1.0),cex=1.0)

################################################################################

# Clean up all of your files, or shut down R before doing another exercise

################################################################################

rm(list=ls(all=TRUE)) # removes anything that might be restored from previously running R

ls()

## Topic 5: Linear regression using least squares

### Exercise: Basic Statistics and Regression Analysis Using R

*Objectives:*  In this exercise you will use R to get basic statistics and a linear regression representing an allometric relationship for trees.

*Files Needed*: You will need the files: ht\_dbh.xlsand ht\_dbh.txt (the tree data) and ht\_dbh.R (R commands, called R script).

*Exercise:* A forest land owner measures the outside bark diameters at 1.30 m above ground (dbh) and total tree height from ground to tree tip for a sample of 20 trees on a small piece of land. The trees are equally spaces over the land area. The measures are:

|  |  |  |
| --- | --- | --- |
| Tree Number | Dbh (cm) | Height (m) |
| 1 | 10.1 | 14.2 |
| 2 | 11.2 | 15.1 |
| 3 | 19.7 | 25.3 |
| 4 | 20.5 | 21.2 |
| 5 | 17.8 | 21.5 |
| 6 | 17.0 | 18.0 |
| 7 | 11.0 | 12.1 |
| 8 | 4.1 | 5.2 |
| 9 | 6.0 | 6.3 |
| 10 | 8.0 | 9.1 |
| 11 | 2.3 | 10.1 |
| 12 | 20.1 | 19.2 |
| 13 | 18.0 | 16.0 |
| 14 | 22.1 | 26.3 |
| 15 | 16.3 | 17.3 |
| 16 | 20.5 | 19.8 |
| 17 | 17.0 | 20.1 |
| 18 | 18.0 | 22.3 |
| 19 | 17.0 | 19.5 |
| 20 | 19.7 | 18.6 |

Before we can do any analysis, we need to bring these data into the R environment. We can do this by:

1. Typing the data right into the R script (Parts I to IV of this Exercise)
2. Entering the data into EXCEL (e.g., ht\_dbh.xls) and then saving this as a tab delimited text file (e.g.,ht\_dbh.txt) or comma delimited file (e.g., ht\_dbh.csv) (Part V of this Exercise).

Once the data are in the R environment, we can get basic statistics, fit models, get graphs, etc.

For this exercise, R script was provided as ht\_dbh.R. The script is organized in parts using comments (the # denotes comments). To learn what the script is doing, you should run this in pieces and determine what the R code is doing before you move on to the next step.

To run this in segments, you can copy and paste a part of the R script into the work session, and then running that part. Another way that we will use is to highlight a part of the script and using Ctrl+R to run that part of the script**.**

The work session window will include the R commands, and the outputs. At any time, you can copy and paste any part of the session window into a WORD file, or store the entire work session window.

1. First, start R, and bring the script in by using File and then Open Script. Browse until you find the ht\_dbh.R file and click on it to bring it into R. You will see that there are comments added to the script to explain what each line of code does. Remember, comments begin with # .
2. Part I: Using the R script provided as ht\_dbh.R, highlight Part I of the code that brings the data into R. This is done by 1) highlighting that part of the code, and 2) using Ctrl+R to run the code. You should see results in the “session” window.

*What did each line do? Try to understand how each line of code was used to bring the data into the R environment.*

1. Part II. Run the next part of the R code provided to calculate simple statistics for the heights. For each item in this list, 1) find the R code, highlight the code, and use Ctrl+R to run the code. *Write down the answers you obtain.* 
   1. The sample mean
   2. The variance
   3. The standard error of the mean
   4. The mode
   5. The median
   6. The coefficient of variation as a percent
   7. A 95% confidence interval for the true mean (all of the trees).
   8. Given the sample data, and no assumptions about the probability distribution, what is the estimated probability that a tree will be more than 10.0 cm in dbh?
   9. Given the sample data, and the assumption that it follows a normal distribution, what is the estimated probability that a tree will be more than 10.0 cm in dbh?
2. Before running more of the provided R code, modify this to obtain the same statistics for dbh. To do this, use File and New Script to open a new window for your script that you will create. Then, copy and paste the code for the height basic statistics into the file, save it, and modify it for dbh instead of height. *Again, write down the answers as you get them* OR *copy and paste them from the console to a WORD file.*
3. Parts III and IV. Now, we would like a model to predict height from dbh, since height is harder to measure. The fitted model can then be used where only dbh was measured. Using the R code provided, locate and run the part of the code fits the model. *Run this in parts, as before and write down your answers as you go.* 
   1. Graph the height versus dbh for these sample data. NOTE: This will appear in a Graph window. Save the graph as picture for future reports.
   2. Since this is not a linear relationship, transformations are needed to linearize the relationship before using linear regression. NOTE: Part III does height versus dbh (no transformations) whereas Part IV uses transformations.
   3. Fit a simple linear regression of height versus your transformed dbh NOTE: There is no need to change units to be the same for both variables. Write down the answers that you get as you use the script to get:
      1. The estimated intercept and slope. Use the estimated slope and intercept and overlay your equation over the selected graph in part c.
      2. Calculate the standard errors and 95% confidence intervals for the intercept and for the slope.
      3. The coefficient of determination (r2) and the standard error of the estimate (SEE), also called the root mean squared error (Root MSE). What do these mean?
      4. Graph the fitted line over the original points.
      5. Based on the graph, are the assumptions that the line fits the data and that variances of y’s around the x’s are equal met for your selected equation? (i.e., you need the residual plot).
      6. Are errors normally distributed?
      7. How would you check the assumption that the observations are independent for these data?
4. Before going to Part V, clean up your all of your work and remove all objects. This is done by using Edit and Clear console and also Misc and then Remove all objects. This allows you to start fresh, getting rid of any variables and data you brought in, and any outputs you have created. This can prevent errors, but you must bring in new data after clearing out all the objects.
5. Part V: In this part, the data come from an EXCEL file instead of being entered into the R code itself. These data were entered into EXCEL and then saved as a tab delimited text file to be used in R (ht\_dbh.txt ). You must give the full path for your data, and NOTE that the folders are given after \\ instead of the usual \ used by Microsoft Windows. *Run this other script, and again write down your answers as with Question 4 c.*

# Part I. FIRST WAY TO BRING DATA INTO R. Type the numbers directly into R, each variable separately, and

# then append these together. Each variable will be a column in a matrix called treedat.

# Then, put this into a dataframe for use later (another way to store the data) called treedat2.

################################################################################

# enter in the tree numbers first

treeno<-c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)

treeno

# list the first entry of trees

treeno[1:1]

#

# enter in the dbh data

dbh<-c(10.1,11.2,19.7,20.5,17.8,17.0,11.0,4.1,6.0,8.0,2.3,20.1,18.0,22.1,16.3,20.5,17.0,18.0,17.0,19.7)

dbh

#

# enter in the height data

height<-c(14.2,15.1,25.3,21.2,21.5,18.0,12.1,5.2,6.3,9.1,10.1,19.2,16.0,26.3,17.3,19.8,20.1,22.3,19.5,18.6)

height

#

treedat<-data.frame(treeno,dbh,height)

treedat

# to select the dbh column of the dataframe, only, use

col2<- treedat[,"dbh"]

col2

################################################################################

# Clean up the workspace before going to the next part

################################################################################

ls() # This lists all of the objects in the workspace

rm(col2,treeno,dbh,height) # Remove these since they are now part of treedat and treedat2 objects.

ls()

################################################################################

# Part II. Basic statistics. for dbh values, get a. the mean; b. the variance;

# c the standard error of the mean; d. the mode; e. the median; f. The CV as a percent;

# g. A 95% CI for the true mean; h. proportional of obs > 10.9 cm dbh;

# i. Prob. of > 10.0 cm dbh assuming a normal distribution;

##############################################################################

n<-length(treedat[,"dbh"])

n # no. of observations

dbhbar<- sum(treedat[,"dbh"])/n

dbhbar # a. mean dbh

ssdbh<- sum((treedat[,"dbh"]-dbhbar)^2) # corrected sums of squares, dbh

vardbh<-ssdbh/(n-1)

vardbh # b. variance of dbh

sterrordbh<- (vardbh/n)^0.5

sterrordbh # c. standard error of the mean dbh

sorteddbh<-sort(treedat[,"dbh"])

mediandbh<-sorteddbh[n/2] # find the dbh at n/2, the middle

mediandbh #d. median dbh

quantile(treedat[,"dbh"]) # to get all the quantiles

cvdbh<-100\*((vardbh^0.5)/dbhbar)

cvdbh #e. cv as a percent

df<-(n-1) # degrees of freedom is no of samples - 1

tvalue<-qt(0.975,df) # get the tvalue for 1-alpha/2 to get a (1-alpha)\*100 confidence interval

lowerCI<- (dbhbar-(tvalue\*sterrordbh)) # g. A 95% CI for the true mean

upperCI<- (dbhbar+(tvalue\*sterrordbh))

df; tvalue; lowerCI; upperCI

dbh<- treedat[,"dbh" ]

nobs<- length(dbh [dbh > 10.9])

proplarge<- (nobs/n)\*100

sorteddbh

proplarge # h. proportional of obs > 10.9 cm dbh

sddbh<- (vardbh^0.5)

zvalue<- (10.0 - dbhbar)/sddbh

zvalue

problarge<- 1-pnorm(zvalue)

problarge # i. Prob. of > 10.0 cm dbh assuming a normal distribution

################################################################################

# statistics using built in functions, useful for data in general

################################################################################

# means and other stats of all variables using built in stats functions

mean(treedat[,"dbh"])

mean(treedat[,"height"])

sd(treedat[,"dbh"])

sd(treedat[,"height"])

# histograms of variables

hist(treedat[,"dbh"]) # note that this graph will be replaced by the next one, unless it is saved

# The histogram will appear in a separate window. Can use File, and Save As, to save the histogram

# as a .jpg or other picture file

hist(treedat[,"height"]) # The dbh histogram disappears, and the height histogram appears

hist(treedat[,"height"],plot=FALSE) # calculates frequencies but no histogram is drawn

# Sort the trees by dbh first and then by height. First step is to calculate the indexes for this order.

indexsort<-order(treedat$dbh,treedat$height)

# use these indices to then sort the trees

sorttreedat<-treedat[indexsort,]

sorttreedat # data sorted by dbh and height afterward

# summary statistics for each of the files-- all the same data just stored differently

require(stats) # This is another R pack with other functions in it.

summary(treedat)

summary(sorttreedat)

################################################################################

# Clean up the workspace before going to the next part

################################################################################

ls() # This lists all of the objects in the workspace

rm(cvdbh, dbh,dbhbar,df,indexsort,lowerCI,mediandbh,n,nobs,problarge,

proplarge,sddbh,sorteddbh,sorttreedat,ssdbh,sterrordbh,tvalue,upperCI,vardbh,

zvalue) # Remove these since they are now part of treedat and treedat2 objects.

ls()

################################################################################

Part III. linear regression of height vs. dbh

################################################################################

# Obtain the regression using the dataframe, treedat2. lm calculates the linear regression.

# Summary prints out some of the results obtained.

summary(lm(treedat$height~treedat$dbh))

# plot the original data and the fitted line.

plot(treedat$height~treedat$dbh)

abline(lm(treedat$height~treedat$dbh))

# since treedat was created as a dataframe, the names can be attached for simpler commands

attach(treedat) # this allows you to use the dataframe, treedat2, with shorter names for variables

names(treedat)

cor(treedat) # correlations first

plot(height~dbh) # simple plot of height (y) versus dbh (x)

lm.height=lm(height~dbh) # fit the linear model and store it as an object

lm.height

summary(lm(height~dbh)) # can use the fitted linear model that was stored as an object instead

summary(lm.height)

anova(lm.height)

plot(height~dbh)

abline(lm.height) # plot the height versus dbh and overlay the regression line

yhat<-fitted(lm.height) # store the predicted values in an object, yhat.

resid<-resid(lm.height) # store the observed - predicted values, called the residuals, in resid

cbind(height,yhat,resid) # list measured and predicted height and differences (i.e. residuals)

plot(yhat~height)

abline(a=0,b=1) # plot a reference line where yhat equals height

plot(resid~yhat) # residual plot

plot(resid~dbh) #different residual plot

qqnorm(resid) # normality plot

qqline(resid,col=2)

hist(resid, freq=FALSE,breaks = 8, density=10,col="green", border="black") # draws a histogram

################################################################################

# Clean up your files, etc.

################################################################################

detach(treedat) # This just detaches the dataframe, treedat, but it can be reattached

ls() # This lists all of the objects in the workspace

# keep the treedat, lm.height, resid,and yhat objects

################################################################################

# Part IV. Fixing problems using transformations. Do the transformations using the dbh and

# height objects, and then put this into a new dataframe.

################################################################################

attach(treedat) # reattach treedat dataframe

dbhsq<-dbh^2 # get dbh squared

logdbh<-log(dbh) # get log of dbh

treedat2<-data.frame(treedat,dbhsq,logdbh) # add these to treedat and save in treedat3

treedat2

detach(treedat)

rm (dbhsq,logdbh)

attach(treedat2) # This attaches treedat2 to the session, meaning that you can now use the variable names

names(treedat2)

# regression using dbh squared. This gives a parabola shape.

lm.height2<-lm(height~dbhsq)

lm.height2

summary(lm.height2)

anova(lm.height2)

yhat2<-fitted(lm.height2)

resid2<-resid(lm.height2)

cbind(height,yhat2,resid2) # list measured and predicted height and differences (i.e. residuals)

plot(yhat2~height)

abline(a=0,b=1) # plot a reference line where yhat equals height

plot(resid2~yhat2) # residual plot

plot(resid2~dbh) #different residual plot

qqnorm(resid2) # normality plot

qqline(resid2,col=2)

hist(resid2, freq=FALSE,breaks = 8, density=10,col="green", border="black") # draws a histogram

################################################################################

# Clean up your files, etc.

################################################################################

detach(treedat2) # This just detaches the dataframe, treedat2, but it can be reattached

ls() # This lists all of the objects in the workspace

rm(lm.height,lm.height2,resid,resid2,treedat,treedat2,yhat,yhat2) # remove all objects

ls()

################################################################################

# Part V. SECOND and THIRD WAY to bring in data, and CI's for predicted heights

# 1. bring in data from an outside tab delimited text file, can be converted from a EXCEL file

# (shown in this example)

# 2. If EXCEL saved as comma delimited, then change this to:

# treedat4<-read.csv("E:\\R\_workshop\\workshop 2009\\data\\ht\_dbh.csv",header=TRUE)

################################################################################

treedat3<- read.table("E:\\R\_materials\_workshops\\workshop\_2013\\data\\ht\_dbh.txt",header=TRUE)

attach(treedat3)

names(treedat3)

treedat3

dbhsq<-dbh^2 # dbh squared

logdbh<-log(dbh) #log dbh

treedat4<-data.frame(treedat3,dbhsq,logdbh)

treedat4

detach(treedat3)

rm(treedat3,dbhsq,logdbh)

attach(treedat4)

lm.height<-lm(height~dbhsq)

lm.height

summary(lm(height~dbhsq))

anova(lm(height~dbhsq))

yhat<-fitted(lm.height)

resid<-resid(lm.height)

cbind(height,yhat,resid) # list measured and predicted height and differences (i.e. residuals)

plot(yhat~height)

abline(a=0,b=1) # plot a reference line where yhat equals height

plot(resid~yhat) # residual plot

plot(resid~dbh) #different residual plot

qqnorm(resid) # normality plot

qqline(resid,col=2)

hist(resid, breaks =8 , density=10,col="green", border="black") # draws a histogram

# get 95% confidence intervals for the average height, given each of the dbh's in the data, using the equation

predCI<- predict(lm.height, treedat4,interval="confidence")

predCI

# plot the orginal data, the predicted values and 95% confidence interval bands

rank<- order(dbh)

sortedpredCI=predCI[rank,]

dbhsort<-dbh[rank]

plot(dbh,height)

matlines(dbhsort,sortedpredCI,lty=c(1,2,2),color="black") #overlay the predicted values and CI using linetypes given

# get confidence intervals for the average heights for dbh=20 cm

dbhnew<-20.0

dbhsqnew<-dbhnew^2

new <- data.frame(dbh=dbhnew,dbhsq=dbhsqnew)

# default is for a 95 percent confidence interval for mean y given the x. Can change the percent by including the level

pred.w.clim <- predict(lm(height~ dbhsq), new, interval="confidence",level=0.90)

pred.w.clim

pred.w.plim <- predict(lm(height ~dbhsq), new, interval="prediction",level=0.90)

pred.w.plim

################################################################################

# Clean up all of your files, or shut down R before doing another exercise

################################################################################

rm(list=ls(all=TRUE))

### Exercise: Multiple Linear Regression Using R

*Background:*

Multiple linear regression uses more than one x-variable to predict the variable of interest, the y-variable. The x’s can be several different variables that we have measured, or can be the originally measures variables, plus transformations of these variables. For example, we may use dbh and dbh squared to predict height, rather than just dbh or just dbh squared. In the case of the transformed variables, we are trying to meet the assumption that the linear model is correct.

*Objective:*  Practice bringing in data that originally in an EXCEL file, and practice using R to get an equation with more than one predictor variable (x-variable) in a multiple linear regression.

*Files:* For this, you will use data gathered for a few African trees (provided by Dr. Akindele). The data can be found inafrican\_trees.xls*.* There is also R script provided as mlr.R

1. *Getting the data into R:* 
   1. In EXCEL, bring up the data file.
   2. Save this as a tab delimited text file called african\_trees.txt*.*
   3. Start R.
   4. Bring in the R script found in mlr.R.
   5. Modify the R script by correcting the path for the datafile.
2. Use the script to run a multiple linear regression to predict height (Ht) from dbh (Dbh) and transformations of dbh. As with Exercise 1, run this in segments and relate what happens to the R code that you have run (i.e., highlight a part of the code and use Ctrl+R to run that part). There are blank lines in the code to indicate a “part” of the code that should be run at the same time.

NOTE: In R, the variable *dbh* is different from the variable *Dbh* – capitals matter.

################################################################################

# bring in data from an outside tab delimited text file, can be converted from a EXCEL file

# (shown in this example)

#

################################################################################

rm(list=ls(all=TRUE)) # remove anything that might be remaining from prior analyses

# change this to read in the correct data file with the full path ############################

treedat<- read.table("E:\\R\_materials\_workshops\\workshop\_2013\\data\\african\_trees.txt",header=TRUE)

attach(treedat)

names(treedat)

Dbhsq<-Dbh^2 # Dbh squared

logDbh<-log(Dbh) #log Dbh

loght<- log(Ht) # log height

treedat2<- data.frame(treedat,Dbhsq,logDbh,loght)

detach(treedat)

rm(treedat,logDbh,loght,Dbhsq)

attach(treedat2)

names(treedat2)

ls()

treedat2[1:5,]

lm.height<-lm(Ht~Dbh+Dbhsq) # parabola shape

lm.height

summary(lm(Ht~Dbh+Dbhsq))

anova(lm(Ht~Dbh+Dbhsq))

yhat<-fitted(lm.height)

resid<-resid(lm.height)

cbind(Ht,yhat,resid) # list measured and predicted height and differences (i.e. residuals)

plot(yhat~Ht)

abline(a=0,b=1) # plot a reference line where yhat equals height

plot(resid~yhat) # residual plot

plot(resid~Dbh) #different residual plot

qqnorm(resid) # normality plot

qqline(resid,col=2)

hist(resid, breaks =6 , density=10,col="green", border="black") # draws a histogram

# get 95% confidence intervals for the average height, given each of the Dbh's in the data, using the equation

predCI<- predict(lm.height, treedat2,interval="confidence")

predCI

# plot the orginal data, the predicted values and 95% confidence interval bands

rank<- order(Dbh)

sortedpredCI=predCI[rank,]

Dbhsort<-Dbh[rank]

plot(Dbh,Ht)

matlines(Dbhsort,sortedpredCI,lty=c(1,2,2),color="black") #overlay the predicted values and CI using linetypes given

# get confidence intervals for the average heights for Dbh=20 cm

Dbhnew<-20.0

Dbhsqnew<-Dbhnew^2

new <- data.frame(Dbh=Dbhnew,Dbhsq=Dbhsqnew)

# default is for a 95 percent confidence interval for mean y given the x. Can change the percent by including the level

pred.w.clim <- predict(lm(Ht~ Dbh+Dbhsq), new, interval="confidence",level=0.90)

pred.w.clim

pred.w.plim <- predict(lm(Ht ~ Dbh+Dbhsq), new, interval="prediction",level=0.90)

pred.w.plim

lm.height2<-lm(Ht~Dbhsq)

anova(lm(Ht~Dbhsq))

anova(lm.height2,lm.height) # partial F test to compare the two nested models

################################################################################

# Clean up all of your files, or shut down R before doing another exercise

################################################################################rm(list=ls(all=TRUE))

### Using R and Stepwise Methods to Select Predictor Variables in a Regression Model

**Using R and Stepwise Methods to Select Predictor Variables in a Regression Model**

*Background:*

Stepwise methods can be helpful for selecting some x-variables for predicting the y-variable. Methods can be forward (in only), backward (out only) or both (in and out). The resulting subset of x variables can be different, depending upon the method used. Once subsets of x variables are obtained using these selection methods, a full regression can be run, and the assumptions checked, etc.

*Files*: We will use the plot data found in volha\_landsat.csv for this exercise. Each observation is a cluster plot (ground) on which volume per ha of trees was calculated, along with Landsat reflectances and indices from an equivalent spatial window. The script can be found in stepwise.R.

*Exercise:*

Run the script in sections, as before, to be able to understand what the R code does. Then, using one of the subsets of selected variables, run a full regression analysis and check assumptions, etc.

*For discussion:*

How useful were these selection methods for choosing x variables to predict volume per ha? Did you obtain a good result with your full regression using the subset of x variables?

################################################################################

# bring in data from an outside comma separated (.csv) file, can be converted from a EXCEL file

# (shown in this example)

#

################################################################################

require(MASS)

standdat<- read.csv("E:\\R\_materials\_workshops\\workshop\_2013\\data\\volume\_ha\_landsat.csv",

header=TRUE)

attach(standdat)

names(standdat)

edit(standdat)

ls()

################################################################################

# Using step to do stepwise methods to help in choosing x variables. Starts with all

# x variables, as shown by volha ~ . and the specific data frame. First one uses

# forward (in only), second one uses backward (out only), and last allows

# variables to be dropped and later to enter back in (both).

################################################################################

# model1 has all x-variables

model1 <- lm(tvolha ~ .-X\_coordmn+Y\_coordmn, data = standdat)

summary(model1)

# model2 has all but the X spatial coordinate

model2<-update(model1, .~.-X\_coordmn)

summary(model2)

# model3 has all but the X and Y spatial coordinates

model3<-update(model2, .~.-Y\_coordmn)

summary(model3)

step.model1 <- stepAIC(model3,direction=c("forward"))

summary(step.model1)

step.model2 <- stepAIC(model3,direction=c("backward"))

summary(step.model2)

step.model3 <- stepAIC(model3,direction=c("both"))

summary(step.model3)

################################################################################

# Once you have a model, do a full analysis, as you need to check assumptions, etc.

################################################################################

lm.volume<-lm(tvolha~b1mn+b7mn+srmn+ndvimn+tvimn+vc2mn)

lm.volume

summary(lm.volume)

anova(lm.volume)

yhat<-fitted(lm.volume)

resid<-resid(lm.volume)

cbind(tvolha,yhat,resid) # list measured and predicted height and differences (i.e. residuals)

plot(yhat~tvolha)

abline(a=0,b=1) # plot a reference line where yhat equals volha

plot(resid~yhat) # residual plot

qqnorm(resid) # normality plot

qqline(resid,col=2)

hist(resid, breaks =6 , density=10,col="green", border="black") # draws a histogram

# get 95% confidence intervals for the average vol per ha, given each of the observed x's in the data, using the equation

predCI<- predict(lm.volume, standdat,interval="confidence")

predCI

################################################################################

# Clean up all of your files, or shut down R before doing another exercise

################################################################################

## Topic 6: Generalized Linear Models

## 

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

### Example Logistic Regression (y-variable is binomial): Species presence.

*R code:* 06\_GLM\_birds..r *Data:* PtCnt\_FRST532B.csv

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

#############################################################################################

# Generalized linear models - loistic regression to fit point count data

# 06\_GLM\_birds.r UBC R workshop

#

#############################################################################################

library(MASS)

library(ROCR)

# read in data and create data frame 'brddata' and examine names and dimensions

setwd("C:\\temp")

brddata <- read.csv(file="PtCnt\_FRST532B.csv")

dim(brddata)

names(brddata)

# Z indicates standardized variables

# URB\_100Z - Urban/industrial area within a 100m buffer

# RUR\_100Z - Rural area within a 100m buffer

# FOR0\_100Z - Forest cover in a 100 m buffer structural stages open and shrubby

# FOR1\_100Z - Forest cover 100 m; structural stages closed and young forest

# FOR2\_100Z - Forest cover 100 m; structural stages mature and old forest

# SAV\_100Z - Savannah area within a 100m buffer

# SHR\_100Z - Shrub area within a 100m buffer

# CR\_CL\_2Z - Crown closure (26-60%) area within a 100m buffer (Landsat)

# CR\_CL\_3Z - Crown closure (>60%) area within a 100m buffer (Landsat)

# Is\_sizeZ - Island size of plot location

# rd\_up\_100Z - Road length unpaved within a 100 m buffer

# rd\_p\_100Z - Road length paved within a 100 m buffer

###OLS fit. This model does not make sense, but it shows that a

# generalized linear model is the same as the usual linear model

# when a normal distribution (gaussian) is specified.

ols1 <- glm(CR\_CL\_2Z ~ Easting+Northing, family=gaussian, data=brddata)

summary(ols1)

###Check residual and other plots

###Set up four plots per page

par(mfrow=c(2,2))

plot(ols1)

## GLM: use glm() to fit mortality model for DF

brcr\_m1 <- glm(BRCR ~ CR\_CL\_2Z, family=binomial, data=brddata)

##Examine the plot diagnostics. Plots will have the same interpretation as do those from OLS

par(las=1, mfrow=c(2,2), cex=0.75, mar=c(4,4,4,1))

plot(brcr\_m1)

summary(brcr\_m1)

### Classification table

### Set threshold in this case the mean value is chose

### not run because ROCR used instead

### threshold<-mean(fitted(brcr\_m1))

### table(brcr\_m1$y,fitted(brcr\_m1)>threshold)

### show ROCR plot and calculate AUC value

par(mfrow=c(1,1))

brcrpr <- prediction(fitted(brcr\_m1), brcr\_m1$y)

brcrpe <- performance(brcrpr, measure = "tpr", x.measure = "fpr")

brcrauc <- performance(brcrpr, measure = "auc")

plot(brcrpe, colorize=T)

brcrauc@y.values[[1]]

### Concordance table ###

### requires custom function: "Concordance.function.r" ###

### not run

### Concordance(brcr\_m1)

###Additional Predictors####

###Nested comparison of models [M(1) versus M(2)]

brcr\_m2 <- glm(BRCR ~ CR\_CL\_2Z + rd\_up\_100Z + Is\_sizeZ, family=binomial, data=brddata)

summary(brcr\_m2)

##Generalized likelihood ratio test (uses Deviance)

##Note the order is important

anova(brcr\_m2, test= "Chisq")

brcr\_m3 <- glm(BRCR ~ CR\_CL\_2Z + rd\_up\_100Z, family=binomial, data=brddata)

anova(brcr\_m3, test= "Chisq")

##Wald test and coefficients

summary(brcr\_m3)

#Extract AICs and compare Models 1, 2 and 3

AIC(brcr\_m1)

AIC(brcr\_m2)

AIC(brcr\_m3)

## Proportion of deviance explained by each model

## (null-residual deviance)/null.deviance

r1 <- (brcr\_m1$null.deviance-deviance(brcr\_m1))/brcr\_m1$null.deviance

r2 <- (brcr\_m2$null.deviance-deviance(brcr\_m2))/brcr\_m2$null.deviance

r3 <- (brcr\_m3$null.deviance-deviance(brcr\_m3))/brcr\_m3$null.deviance

(rsq <- cbind(r1, r2,r3))

##Estimate dispersion parameter estimates for Model 3

###

deviance(brcr\_m3)

summary(brcr\_m3)$dispersion ## 1 (by definition)

dfr <- df.residual(brcr\_m3)

deviance(brcr\_m3)/dfr

d3 <- sum(residuals(brcr\_m3,"pearson")^2)

disp3 <- d3/dfr

gg3 <- update(brcr\_m3,family=quasipoisson)

summary(gg3)$dispersion

#pchisq(d3,df=dfr,lower.tail=FALSE)

all.equal(coef(brcr\_m3),coef(gg3)) ## TRUE

se1 <- coef(summary(brcr\_m3))[,"Std. Error"]

se2 <- coef(summary(gg3))[,"Std. Error"]

se2/se1

sqrt(disp3)

#str(brcr\_m3)

par(mfrow=c(2,2))

plot(brcr\_m1)

#1-pchisq(599.18, 2199)

##Deviance or likelihood ratio test

##Note the order is important

brcr\_m3 <- glm(BRCR ~ CR\_CL\_2Z + rd\_up\_100Z, family=binomial, data=brddata)

anova(brcr\_m2, test= "Chisq")

anova(brcr\_m3, test= "Chisq")

#Generalized likelihoof ratio test

anova(brcr\_m1,brcr\_m2, brcr\_m3, test= "Chisq")

summary(brcr\_m3)$coef

##Model fit using quasibinomial

brcr\_m4 <- glm(BRCR ~ CR\_CL\_2Z + rd\_up\_100Z, family=quasibinomial, data=brddata)

summary(brcr\_m4)

anova(brcr\_m4, test= "Chisq")

### Example: The y-variable is count, with or without extra zeros: Bird counts.

*R code:* 07\_Other\_GLMs.r *Data:* bird\_cnt.csv

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

library ("MASS")

library(lmtest)

library(pscl)

library(sandwich)

setwd("C:/temp")

data <- read.csv(file="bird\_cnt.csv")

Creeper <- data$BRCR

Rural <- data$RUR\_100Z

Crown\_cl <- data$CR\_CL\_2Z

Road\_up <- data$rd\_up\_100Z

#Creeper Histogram

tmp <- hist (Creeper, breaks=0:(max(Creeper)+1), main="Brown Creeper point count data",

xlab="Number of individuals identified", ylab="Frequency", ylim=c(0,400), right=FALSE, axes=F)

axis(2)

axis(1, at=tmp$mids, labels=0:max(Creeper))

bird <- data.frame(cbind(Creeper, Rural, Crown\_cl, Road\_up))

pois<-glm(Creeper ~Rural + Crown\_cl + Road\_up,family = poisson, data = bird)

summary(pois)

coeftest(pois, vcov = sandwich)

qpois<-glm(Creeper ~Rural + Crown\_cl + Road\_up,family = quasipoisson, data = bird)

summary(qpois)

nb<-glm.nb(Creeper ~Rural + Crown\_cl + Road\_up,link = "log", data = bird)

summary(nb)

llhNB = logLik(nb)

llhPoisson =logLik(pois)

d <- 2 \* (llhNB - llhPoisson)

pval <- pchisq(as.numeric(d), df=1, lower.tail=FALSE)/2

pval

lrtest(pois,nb)

##ZERO INFLATION

f1 <- formula(Creeper ~ Rural + Crown\_cl + Road\_up

| Rural + Crown\_cl + Road\_up)

Zip1 <- zeroinfl(f1, dist = "poisson", link = "logit",

data = bird)

summary(Zip1)

f4 <- formula(Creeper ~ Rural + Crown\_cl + Road\_up

| Rural + Crown\_cl + Road\_up)

Nb4<- zeroinfl(f4, dist = "negbin", link = "logit",

data = bird)

summary(Nb4)

lrtest(Nb4,Zip1)

llhZNB = logLik(Nb4)

llhZPoisson =logLik(Zip1)

d <- 2 \* (llhZNB - llhZPoisson)

pvalz <- pchisq(as.numeric(d), df=1, lower.tail=FALSE)/2

pvalz

#estimates

fm <- list("ML-Pois" = pois, "Quasi-Pois" = qpois, "NB" = nb,

"ZI-Pois" = Zip1, "ZI-NB" = Nb4)

sapply(fm, function(x) coef(x)[1:8])

#stderrors

cbind("ML-Pois" = sqrt(diag(vcov(pois))),

"Adj-Pois" = sqrt(diag(sandwich(pois))),

sapply(fm[-1], function(x) sqrt(diag(vcov(x)))[1:8]))

#-2\*LL

rbind(logLik = sapply(fm, function(x) logLik(x)),

Df = sapply(fm, function(x) attr(logLik(x), "df")))

#AIC

rbind(AIC = sapply(fm, function(x) AIC(x)))

#expected to observed zero counts

round(c("Obs" = sum(Creeper < 1), "ML-Pois" = sum(dpois(0, fitted(pois))),

"NB" = sum(dnbinom(0, mu = fitted(nb), size = nb$theta)),

"ZI-Pois" = sum(predict(Zip1, type = "prob")[,1]),

"ZI-NB" = sum(predict(Nb4, type = "prob")[,1])))

#Vuong tests (see Hilbe p. 176+)

vuong(Nb4,nb)

vuong(Zip1,pois)

### Another example of application of a generalized linear model: Large Mammals

*:*

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

# R Books

Albert, J. 2007. Bayesian analysis with R. Springer, New York.

Bivand, R.S., E.J. Pebesma, and V. Gómez-Rubio. 2008. Applied spatial data analysis with R. Springer, New York.

Dalgaard, P. 2002. Introductory statistics with R. Springer, New York.

Everitt, B.S. 2005. An R and S-Plus companion to multivariate analysis. Springer-Verlag, London.

Everitt, B. S. , and T. Hothorn. 2006. A handbook of statistical analyses using R. Chapman & Hall/CRC, New York.

Faraway, J.J. 2006. Extending the linear model with R: Generalized linear, mixed effects, and nonparametric regression models. Chapman & Hall/CRC, New York.

Gentlemen, R., K. Hornik, G. Parmigiani. 2008. Data manipulation with R. Springer, New York.

Jones, O. R. Maillardet, and A. Robinson. 2009. Introduction to scientific programming and simulation using R. CRC Press, New York.

Muenchen, R.A. 2009. R for SAS and SPSS users. Springer, New York.

Murrell, P. 2006. R graphics. Chapman & Hall/CRC, New York.

Pfaff, B. 2006. Analysis of integrated and cointegrated time series with R. Springer, New York.

Shumway, R. H. and D. S. Stoffer. 2006. Time series analysis and its applications with R examples, 2nd edition. Springer, New York.

West, B.T., K.B. Welch, and A. T. Galecki. 2007. Linear Mixed Models: A Practical Guide Using Statistical Software. Chapman & Hall/CRC, New York.

Wood, S.N. 2006. Generalized additive models with R. Chapman & Hall/CRC, New York.

Zuur, A. F., E. N. Ieno and E.H.W.G. Meesters. 2009. A beginner’s guide to R. Springer, New York.

Zurr, A.F., E. N. Ieno, and G.M. Smith. 2007. Analyzing ecological data. Springer, New York.

Zurr, A.F., E. N. Ieno, N. J. Walker, A.A. Saveliev, and G.M. Smith. 2009. Mixed dffects models and extensions in ecology with R. Springer, New York.