Routine fisheries analysis in R

Michael E. Colvin

28-29 July 2009

Part I
Course overview

1 Objectives

The content of this continuing education course will focus on fisheries applications of the R program to: import and export, manipulate, visualize, and analyze data. These topics are geared to provide fisheries professionals with the tools to use the R program to provide analysis for reporting and publishing fisheries data and analysis. In particular this course will emphasize uses of the R program that will potentially be useful for visualization and analysis of data that are periodically updated and incorporated in annual reports.

1.1 About this document

This document is intended to demonstrate the use of the R program for data summarization, visualization, and analysis. Interpretations of analysis are limited and should be supplemented by a good statistics text book. A number of applied fisheries examples are taken from Guy and Brown (2007) and modified so that the analysis runs in R. In depth information regrading the interpretation of analysis results can be found in Guy and Brown (2007). When applicable this document references box numbers used in Guy and Brown (2007) to facilitate further interpretation of results.

1.2 Why R?

- R is free!
- Platform independent (works on Windows, Linux, Mac)
- Publication quality graphics
- Can be a common analysis software among collaborators
- Requires only a fraction of disc space compared to other proprietary softwares
- Additional packages extend the capabilities of R (e.g., GIS capabilities)

1.3 Helpful R web resources

A number of useful web resource are available for the R program and can be found in Table 1.

Table 1: Useful web resource for using the R program.

Site	URL
The R Project	http://www.r-project.org/
Intro to R	http://cran.r-project.org/doc/manuals/R-intro.pdf
IcebreakR	http://www.ms.unimelb.edu.au/~andrewpr/r-users/icebreakeR.pdf
Quick R: R for SAS, Stata users	http://www.statmethods.net/index.html
The R Wiki	http://wiki.r-project.org/rwiki/doku.php
Stats R Us	http://pj.freefaculty.org/R/Rtips.html
R Graphics showcase	http://bm2.genes.nig.ac.jp/RGM2/index.php?clear=all
R in Ecology listserv	https://stat.ethz.ch/mailman/listinfo/r-sig-ecology
R Graph Gallery	http://addictedtor.free.fr/graphiques/
Mike's R stuff	http://www.public.iastate.edu/~mcolvin/r.html
FishR	http://www.ncfaculty.net/dogle/fishR/index.html

2 Getting started with R

2.1 Downloading and installing R

The R program is freely available from the R Project website (www.r-project.org). The base download is relatively small (36 megabytes), compared to other programs (e.g., SAS, STATA, S-PLUS). To download the program follow the following steps:

1. click the CRAN link on the left hand side of the web page
2. Select the Iowa State University CRAN Mirror from the list of potential mirrors
3. Select the Windows link from the download options
4. Click the hyperlink entitled "base"
5. Click "Download R 2.9.1 for Windows" and save to your workstation
6. The appropriate windows install program for R is also included on your packet CD
7. Double click R-2.9.1-win32.exe to install
8. Follow the prompts from the install shield, a typical install will be a "User Installation" with default startup options
9. Allow the program to finish installation

2.2 Is R just a big fancy calculator?

Opening the the R program will result in a window that has a few drop down menus and a window that is called the R Console (Figure 1).

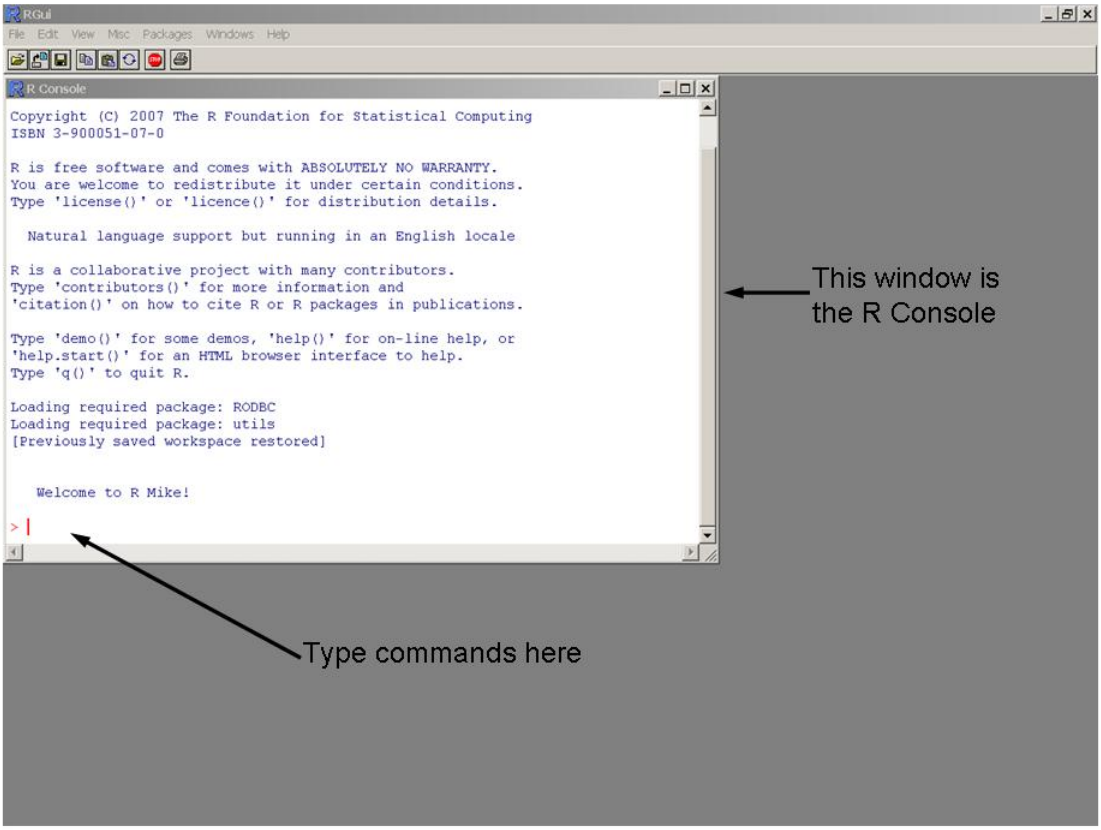

Figure 1: R Console

After opening one can use the command line of the console like a big calculator (R Code Chunk 1).

R code chunk 1

```
2+2 # R WILL RETURN 4
2*2 # R WILL RETURN 4
2*2+6 # R WILL RETURN 10
6+2*2 # R WILL RETURN 10, IT KNOWS THE ORDER OF OPERATIONS
6+(2*2) # R ALSO RECOGNIZES BRACKETS
"AFS CONTINUING EDUCATION COURSES R FUN!" # R WILL RETURN THE TEXT
```

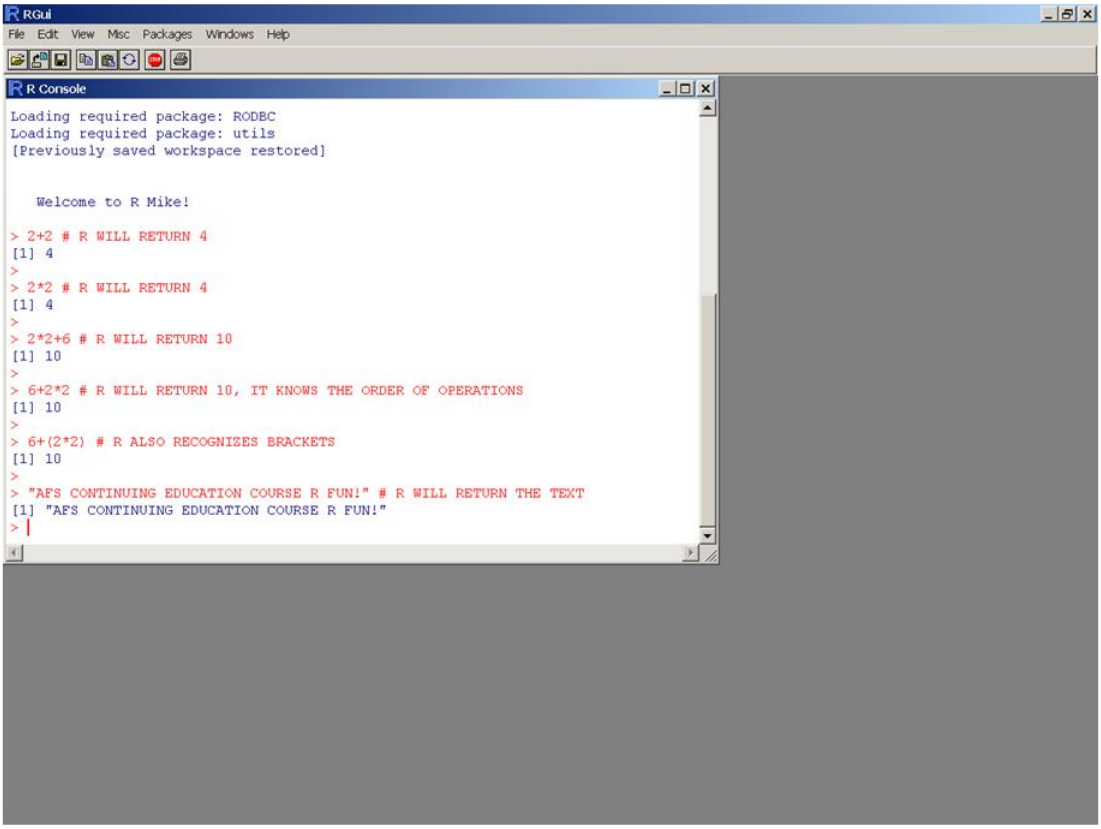

Figure 2: Output from the R Console from R Code Chunk 1.

2.3 Scripts

As illustrated in R Code Chunk 1, the R program is command driven. The beginning of an R session typically starts with opening a new or existing script. Scripts are where an analyst can write, annotate and save analysis code for future use. There are a number of ways of working with R code and scripts. The R program has a built in script editor that is a basically a bare bones text editor. Tinn-R is an open source script editor that specifically interfaces with the R console and has some nice features such as color coding of parentheses (`http://www.sciviews.org/Tinn-R/`). Emacs is another popular open source script editor that offers syntactic indentation and coloring for S dialects (i.e., R, S), using the ESS package (Emacs speaks statistics) (`http://www.gnu.org/software/emacs/`). The aforementioned packages offer some advantages particularly in syntactical tabbing and coloring which may aid novice R programmers in identifying code bugs. However I have found that in practice that the built in R script editor is just as convenient and will be the baseline that is provided for this continuing education workshop.

NOTE: all R scripts should be saved as scriptName.R

2.4 Script annotation

In a script, annotation can be accomplished by the use of the pound sign (#) (R Code Chunk 2). R will ignore everything to the right of allowing an analyst to add comments and meta data for future

use. The start of a typical annotated R script usually starts with some annotation of the reason for the analysis, date, and the person performing the analysis. To open a new script go to *File* > *new script*. Note that when you save a script be sure to add a *.R file extension since the R program does not do this automatically.

R code chunk 2

```
##### THIS IS MY FIRST R SCRIPT
##### ANYTHING I TYPE TO THE RIGHT OF THE # SIGN IS IGNORED BY R
##### I CAN ANNOTATE AS MUCH OR AS LITTLE IS NEEDED
#####
##### ANALYST: JOE BIOLOGIST, JOE.BIOLOGIST@STATEAGENCY.GOV
##### DATE: 1 JULY 2008
```

Annotation becomes increasingly important as an analysis gets more sophisticated and/or when scripts are shared among collaborators. Additionally, a well annotated script is useful if there are periodic breaks during the analysis (e.g., revisiting the analysis during manuscript revisions).

2.5 R help

R comes with a number of built in help commands, the code below demonstrates the use of those commands (R Code Chunk 3).

R code chunk 3

```
?mean # Works!  mean is an R command
help(mean) # Works!  mean is an R command
help(regression) # Fails!  regression is not an R command
help.search("regression") # Well that works alright
RSiteSearch("regression") # We can also look at the R archive on the web
```

2.6 Packages

R is a very lean program requiring very little space on your hard drive. To increase the functionality of R, package have been created by various people who work with R frequently and are distributed through CRAN. It is easy to install commonly used packages to your computer and packages can be called as needed for use in analysis (Table).

R code chunk 4

```
install.packages("package")
require(package)
update.packages() # UPDATES INSTALLED PACKAGES
```

Code breakdown

1. `install.packages("package")`: this bit of code installs a package that may be needed, it only needs to be run once to install the package

2. `require(package)`: after installation a package can be used by loading the package, `load(package)` can be used as well

3. `update.packages()`: allows R to update packages, as they do change periodically

The packages required for this continuing education workshop can be installed to your computer by running R Code Chunk 5.

R code chunk 5

```
install.packages("car")
install.packages("RODBC")
install.packages("gplots")
install.packages("maptools")
install.packages("pda",repos="http://R-Forge.R-project.org")
```

Table 2: Commonly use R packages.

Name	Description
car	Companion to applied regression
pda	Practical data analysis
agricolae	Statistical procedures for agricultural research
vegan	Community ecology package
spatstat	Spatial point pattern analysis, model-fitting, simulation, tests
RODBC	ODBC database access: communicating with excel and access
gplots	Various R programming tools for plotting data

3 Data structures and indexing

There are three primary data structures that are used in the analysis of fisheries data: vector, dataframe, and lists. Each data structure has its own different primary use. Vectors are used to hold a single list of values, while a dataframe is used to hold rows and columns of data, and a list can hold several vectors that are allowed to vary in length. Learning how to index these data structures will be very important for simulations that will be required throughout this continuing education course.

3.0.1 Vectors

R code chunk 6

```
x<- c(1,2,3,4,5,6,7,8)
X<- c(2,3,4,5,10,12,14)# R is case sensitive
x<- as.vector(c(1,2,3,4,5,6,7,8)) # Make a vector named x
x[1] # Get the first value from the vector
x[1:3] # Get values 1 to 3 from the vector
str(x)
str(X)
summary(x)
```

Code breakdown

1. `c()`: used to make a vector of data (`c()` can be thought of as "combine")
2. `as.vector()`: performs similarly to `c()`
3. `x[1]`: gets the first value from the vector `x`
4. `x[1:3]`: Get values 1 to 3 from the vector
5. `str(x)`: returns a summary of the data types
6. `dim(x)`: returns the number of rows and columns
7. `summary(x)`: returns a simple summary of the data

3.0.2 Dataframes

R code chunk 7

```
##### Dataframes are probably the most similar to an excel database
##### Dataframes are required to have the same number of
##### rows for each column
mydataframe<- data.frame(
   x= c(1,2,3,4,5),
   lake=c("Clear","Clear","Spirit","Spirit","Spirit"),
   y= c(2.2,3.1,5.3,6.9,3.1))
mydataframe[c(1:5),] # returns the first 5 rows of the dataframe
mydataframe[c(1:5),c(3)] # first 5 rows of the 3rd field
names(mydataframe) # returns the field names
mydataframe$group # returns all the values for the group column
str(mydataframe)
dim(mydataframe)
summary(mydataframe)
```

Code breakdown

1. `lake=c("Clear","Clear","Spirit","Spirit","Spirit")`: note the text characters are enclosed by parentheses
2. `mydataframe[c(1:5),]`: within the bracket, the rows are indexed before the comma and the columns after the comma
3. `names(mydataframe)`: this returns the field names of the dataset
4. `mydataframe$group`: note the use of the `$` to specify which field to return, any value returned by `names()` is fair game to go after the `$`

3.0.3 Lists

R code chunk 8

```
##### Lists offer more flexibility since they can be "ragged"
##### they are useful for holding simulation and summary data
mylist<- list(lake = c("Wall Lake"), secchiTransparency= c(0.2,0.3,0.25),
fishLength=c(11,12,14,11,30,22.3,40,29))
names(mylist) # returns names of elements
mylist$lake
mylist[[2]] # returns second vector in the list secchiTransparency
mylist$fishLength[1] # Get the first value from fishLength
mylist[[3]][1] # Same as previous
str(mylist)
summary(mylist)
```

Code breakdown

1. `list()`: creates a list
2. `names(myList)`: returns the names of the fields contained in the list
3. `$`: dataset$field, the `$` sign tells R the field name

3.1 Quick in class exercise

1. What is the value for the 4th row and the 6 column for the `mydataframe` dataset?
2. Make a new object x that has the value of the 5th row and 2nd column from the `mydataframe` dataset.

4 Importing data

R can import data from text files such as comma separated files and tab delimited files. Importing data from Microsoft Excel worksheets and Microsoft Access is supported through the RODBC package. A number of other file formats (e.g., Open Office, mySQL) are supported as well through additional packages.

4.1 Importing text files

R code chunk 9

```
read.delim("C:/contEdCourseData/part1/book1.txt", header = TRUE, sep =
"\t")
read.csv("C:/contEdCourseData/part1/book1.csv", header=T)
```

Code breakdown:

1. `"C:/contEdCourseData/part1/book1.txt"`: Specifies the location of the file to be read into R. Note that that "\" are backwards from a typical windows pathway. This is an R quirk that has been set primarily for interoperability of operating systems (e.g., linux, unix, mac). File paths are always enclosed parenthetically either with double (`"c:/filepath"`) or single (`'c:/filepath'`) quotation marks.

2. `header = T`: Specifies that the first row of the imported dataset contains the column names and not actual data

3. `sep="\t"`: Specifies how the data columns are separated. In the case of `sep="\t""`, a tab separation is present. One could also read a comma separated file in the same way by specifying `sep=","`, however the `read.csv` command eliminates the need for this.

4.1.1 Tab delimited files

Since R is an object oriented program we can import a dataset and create an object.

R code chunk 10

```
tabdelim_dataset<-read.delim("C:/contEdCourseData/part1/book1.txt", header =
TRUE, sep = "\t")
tabdelim_dataset # RETURNS THE ENTIRE DATASET TO THE CONSOLE
head(tabdelim_dataset) # SHOWS THE FIRST FIVE ROWS OF THE DATASET
summary(tabdelim_dataset) # SUMMARIZES THE DATASET
```

Code breakdown:

1. `read.delim`: reads a tab delimited file into the R console and makes an data frame entitled tabdelim_dataset to hold the data.

2. `tabdelim_dataset`: is the object that holds the imported data as identified by `<-` (Note = can be used however it is used syntactically in other R functions, so use at your own risk).

3. `header = TRUE`: same as before that the first row contains column names

4. `tabdelim_dataset`: returns the entire dataset to the R console

5. `head(tabdelim_dataset)`: shows the first 5 lines of the dataset

6. `summary(tabdelim_dataset)`: summarizes all the fields of the dataset

4.1.2 Comma separated files

R code chunk 11

```
csv_dataset<-read.csv("C:/contEdCourseData/part1/book1.csv", header=T)
csv_dataset
str(csv_dataset)
dim(csv_dataset)
head(csv_dataset)
summary(csv_dataset)
```

Code breakdown

1. The above code does the same thing as `read.delim` however using a comma separated file.

4.2 Importing from MS Excel or Access

The use of spreadsheet and databases as repositories for data collected during ecological research and monitoring has been increasing over the years. These types of repositories offer a number of advantages over storing data as text files and are now common place. These programs allow the user to save a datasheet, table, or query as tab or comma separated files for import to R and analysis. This additional step is at times problematic with large complex datasets frequently encountered in ecology. Problems often arise when the analyst may update the primary data repository and forget to export the data again or many versions of the exported data exist. The annotation of the steps required to process the data from Excel or Access are at times lost without meticulous record keeping leading the analyst to recreate the analysis dataset. A work around for this is to read the data directly from the data repository and perform any needed cleaning, filtering, sub setting in R. This is beneficial as the commands used in the R script are explicit and allow the analyst to replicate the analysis dataset at will as well as a well defined audit trail (sometimes required by regulatory agencies). The RODBC package works by creating a communication channel between R and Excel or Access.

R code chunk 12

```
# install.packages("RODBC") # already run previously to install
require(RODBC)
```

Code breakdown

1. `install.packages("RODBC")`: installs the required package from the CRAN mirror.
2. `require(RODBC)`: calls a package from the package library for use by R, alternatively `load(RODBC)` can be used

4.2.1 Microsoft Excel

R code chunk 13

```
channel<-odbcConnectExcel("C:/contEdCourseData/part1/book1.xls", readOnly=T)
sqlFetch(channel, "Sheet1")
excel_dataset<- sqlFetch(channel,"Sheet1")
head(excel_dataset)
summary(excel_dataset)
```

Code breakdown

1. `odbcConnectExcel`: creates an open communication channel to the excel workbook

2. `sqlFetch`: imports an entire worksheet from the Excel workbook with the open communication channel

3. `head(excel_datasheet)`: returns the first five rows of a dataset

4. `summary(excel_datasheet)`: summarizes all the fields of the dataset

4.2.2 Microsoft Access

R code chunk 14

```
channel_access<- odbcConnectAccess("C:/contEdCourseData/part1/book1.mdb")
sqlFetch(channel_access,"Book1")
access_dataset<- sqlFetch(channel_access,"Book1")
head(access_dataset)
str(access_dataset)
summary(access_dataset)
```

Code breakdown

1. `odbcConnectAccess("C:/contEdCourseData/part1/book1.mdb")`: creates an open communication channel to the excel workbook

2. `sqlTable`: returns an entire table from a access database

3. `access_dataset<- sqlTable(channel_access,"Book1")`: returns the dataset to an object names access_dataset

4.3 Working directory

Setting a working directory allows us to specify a filepath by typing a period.

R code chunk 15

```
##### Setting a new working directory
getwd() # Gets the current working directory
setwd("C:/contEdCourseData/part1/") # sets the new working directory

##### We can now read in a file using a period
read.csv("./book1.csv") # this saves lots of typing
```

Code breakdown

1. `setwd("C:/contEdCourseData/part1/")`: note the backslashes are "/" and not "\" as with a normal windows pathway. [1]

[1] This allows for platform independence. Code written on a windows operating system will run on a linux or macintosh machine.

4.4 Getting data out

R code chunk 16

```
###### lets save our analysis file
write.csv(excel_dataset, "./filename.csv")
```

Code breakdown

1. `write.csv(excel_dataset, "./filename.csv")`: pretty straight forward, just the R object (excel_datatset) and the path and filename to save the dataset as

4.5 In class exercise

1. Change your current working directory to `"c:"`
2. Use the getwd() function to make sure you working directory has changed
3. Import the trawlData.csv from the folder `C:/contEdCourseData/part1` as an object named trawlData_csv.
4. Import the same dataset from trawlData.xls (worksheet = data) as an object named trawlData.
5. Briefly look at the data using: `head()`, `str()`, and `dim()` functions.

5 Data manipulation and summarization

The following section will use a dataset of length and weight data for fishes captured over two years of trawling. Some things to note are that each fish was measured for total length (mm), however only a subset of fish were measured for weight. Over the two years approximately 45 trawl runs were conducted for a total of approximately 90 runs and the dominant fish captured were walleye, yellow bass, common carp, and black bullhead Colvin et al. (2008, 2009) (Figure 3).

R code chunk 17

```
# IMPORT TRAWL CATCH DATA
trawlData_csv<-read.csv("./trawlData.csv")
channel<- odbcConnectExcel("./trawlData.xls")
trawlData<- sqlFetch(channel, "data")

# HOW BIG IS THE DATASET?
dim(trawlData)
# WHAT ARE THE DIMENSIONS OF THE DATASET?
str(trawlData)
# LETS LOOK AT THE FIRST 5 ROWS OF THE DATASET
head(trawlData)
# WHAT ARE THE NAMES OF THE FIELDS IN THE DATASET?
names(trawlData)
```

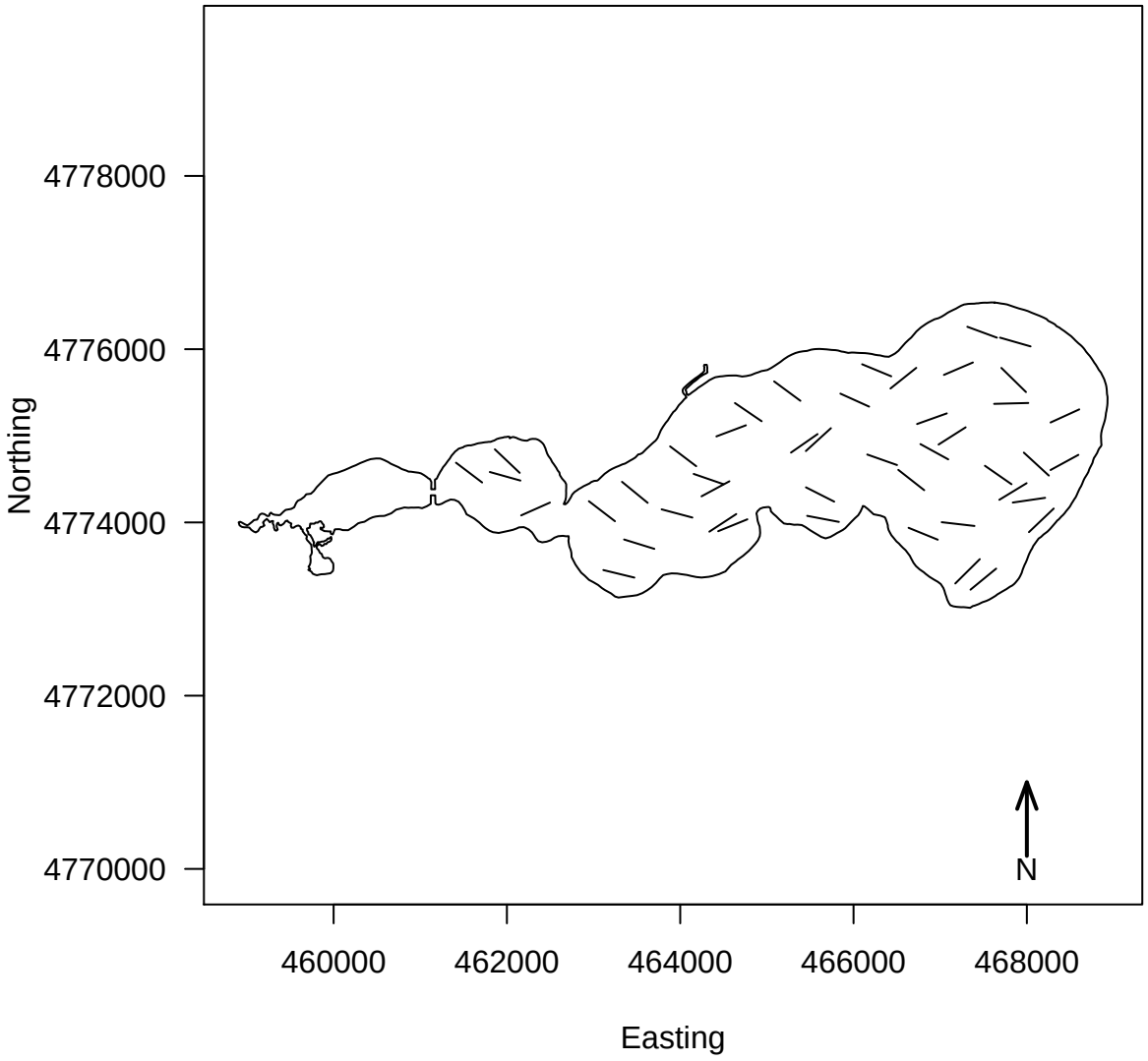

Figure 3: Trawling routes randomly selected for Clear Lake used in fiscal years 2007 and 2008 (n=45).

5.1 Commonly used builtin functions for basic statistics

The R program has a number of useful builtin functions for generate basic statistics. R Code Chunk 18 will use these functions to generate some basic statistics for the lengths of fish captured during two years of trawling.

R code chunk 18

```
# SOME BASIC STATISTICS FOR FISH LENGTHS
mean(trawlData$fishLength) # mean of fish lengths
min(trawlData$fishLength) # min of fish lengths
max(trawlData$fishLength) # maximum of fish lengths
range(trawlData$fishLength) # range of fish lengths
var(trawlData$fishLength) # variance of fish lengths
sd(trawlData$fishLength) # standard deviation of fish lengths
length(trawlData$fishLength) # how many observations?
summary(trawlData$fishLength) # summarize fish lengths
```

5.1.1 Handling missing values

In the case of fish weights in this dataset there are a number of missing values. In general in R there are always several ways to handle this and R Code Chunk 19 will demonstrate how to calculate summary statistics and dealing with missing values.

R code chunk 19

```
summary(trawlData$fishWeight)
# SOME BASIC STATISTICS FOR FISH LENGTHS
mean(trawlData$fishWeight, na.rm=TRUE) # mean of fish weight
min(trawlData$fishWeight, na.rm=TRUE) # min of fish weight
max(trawlData$fishWeight, na.rm=TRUE) # maximum of fish weight
range(trawlData$fishWeight, na.rm=TRUE) # range of fish weight
var(trawlData$fishWeight, na.rm=TRUE) # variance of fish weight
sd(trawlData$fishWeight, na.rm=TRUE) # standard dev.  of fish weight
```

Code breakdown

1. `na.rm=TRUE`: calculates the summary statistic by removing any missing values `na.rm=T` does the same thing

5.2 Converting characters and numbers to factors

The R program by default attempts to classify data fields (columns) as numeric, integer, factor, or characters. This can be problematic if the contents of a data field are numeric and represent a lake or site. NOTE: R is case sensitive

R code chunk 20

```
# WHAT FACTOR LEVELS ARE THERE FOR FISH SPECIES?
summary(trawlData$species)
levels(trawlData$species)
# LETS LOOK AT SITES
summary(trawlData$sites) # it is a integer and is should be a factor
trawlData$sites<- as.factor(trawlData$sites) # convert numeric to a factor
```

5.3 Tabulating data

There are several ways to create simple summaries of data in R using builtin functions `table()` and `xtabs()`that are illustrated in R Code Chunk 21.

R code chunk 21

```
# TABLULATE COUNTS OF FISH BY SPECES
table(trawlData$species) # species count
table(trawlData$basin) # count by lake basin
# TABULATE BASIN BY SPECIES
xtabs(~basin + species, trawlData)
```

The `table()` and `xtabs()` functions are useful for summarizing counts, however cannot handle more complex summaries such as a mean or variance. The `tapply()` function can summarize data based on an index variable such as lake basin or species or even both (R Code Chunk 22).

- `table(variable)`
- `xtabs(rowVariable, columnVariable)`
- `tapply(variableToSummarize, indexVariable, function,na.rm=T)`

R code chunk 22

```
# SUMMARIZE MEAN FISH LENGTH BY BASIN
tapply(trawlData$fishLength, trawlData$basin, mean)
# SUMMARIZE FISH LENGTH VARIANCE BY BASIN
tapply(trawlData$fishLength, trawlData$basin, var)
# SUMMARIZE MEAN FISH LENGTH BY SPECIES
tapply(trawlData$fishLength, trawlData$species, mean)
# WHAT ABOUT WEIGHT
tapply(trawlData$fishWeight, trawlData$species, mean)
tapply(trawlData$fishWeight, trawlData$species, mean, na.rm=TRUE)

# WHAT ABOUT A SPECIES BY BASIN BY YEAR SUMMARY?
# WE NEED TO MAKE A UNIQUE BASIN AND SPECIES INDEX
trawlData$index<- paste(trawlData$year, trawlData$basin, trawlData$species)
# USING THE NEW INDEX VARIABLE WE CAN SUMMARIZE AS BEFORE
tapply(trawlData$fishLength, trawlData$index, mean)
```

Code breakdown

1. `na.rm=TRUE`: Adding the `na.rm=TRUE` to the `tapply()` function tells R to remove any missing values prior to calculating the summary statistic.
2. `paste()`: this function concatenates 2 or more variables into a single variable. In this case we use it to make a unique id to summarize over.

The summaries presented in R Code Chunk 22 are sufficient for looking at summaries of data in R, however in most cases we need to summarize data and export the summarized data for a report, presentation or a manuscript. R Code Chunk 23 will demonstrate how to create a dataframe of summary statistics and export that summary to you working directory.

- `data.frame(columnName1=columnData1, columnName2=columnData2, ...)`

R code chunk 23

```
# MAKING A SUMMARY TABLE FOR AN ANNUAL REPORT
summaryTable<- data.frame( # lets use a dataframe for this
   year= tapply(trawlData$year, trawlData$index, unique),
   basin= tapply(as.character(trawlData$basin), trawlData$index, unique),
   species= tapply(as.character(trawlData$species), trawlData$index,
unique),
   meanLength= tapply(trawlData$fishLength, trawlData$index, mean),
   minLength= tapply(trawlData$fishLength, trawlData$index, min),
   maxLength= tapply(trawlData$fishLength, trawlData$index, max))

# NOW WE CAN SAVE THIS TO PUT IN OUR REPORT
write.csv(summaryTable, "./summaryTable.csv")
# THIS SCRIPT CAN NOW BE RERUN EACH YEAR!
```

Code breakdown

1. `data.frame()`: the use of data.frame allows us to make an excel like spreadsheet of data. The usage is simple `data.frame(name=column1, name=column2, name=column3...)` just make sure all your columns are the same length!
2. `as.character(trawlData$basin)`: this is a nuance of R. Since basin is a factor, internally R references to these as numbers, so we need to convert it to a character to actually get the basin name, not the number!
3. `unique`: this function grabs the unique values from a list of values

5.4 Conditional expressions

There are times when we need to know if a value passes as test. An example of this would be if a fish is a carp or not or if a fish is greater than a certain length. This can be done using the ifelse and the logical operators from Table 3 and is demonstrated in R Code Chunk 24.

- `ifelse(variable, logical argument, True, False)`

R code chunk 24

```
ifelse(trawlData$fishLength > 500 , 1, 0)
trawlData$aggBasins<- ifelse(trawlData$bassin %in% c("little",
"middle"),"Western","Eastern")
trawlData$carp<-ifelse(trawlData$species=="CRP", 1,0)
mean(trawlData$carp) # percent of the catch that was carp
trawlData$carp_bullhead<-ifelse(trawlData$species=="CRP"|trawlData$species=="BBH",
1,0)
# percent of the catch that was carp or bullhead
mean(trawlData$carp_bullhead)
mean(ifelse(trawlData$species%in%c("CRP","BBH"), 1,0))# same as above
```

Code breakdown

1. `ifelse(trawldata$fishLength > 500 , 1, 0)`: this command evaluates whether fishLength is greater than 500 and if this is true a 1 is returned, if not a 0 is returned, this could also be "yes", "no", "true", "false", anything you want however using 1 and 0 allows for easy calculation of percentages.
2. `trawlData$carp<-`: adds a field (column) to our dataframe to hold the results of this test
3. `mean(trawlData$carp)`: by calculating the mean of this we see what percentage of the catch was carp
4. `%in%`: this is used commonly to get several values in a list it works similar to a series of or statements

5.5 Subsetting a dataset

There are many ways to manipulate data post import. The benefit of manipulating data after importing allows the analyst to track manipulations, leaving an audit trail and the original analysis datasets unmanipulated. This is increasingly important when working for agencies with regulatory capacity, collaboration with fellow analysts, and maintaining reproducible analysis.

- `subset(dataset, subsetting arguments)`

R code chunk 25

```
subset(trawlData, species==c("CRP"))
subset(trawlData, fishLength >250)
subset(trawlData, (species==c("CRP") & fishLength >250))
subset(trawlData, species %in% c("CRP","BBH","BUF"))
subset(trawlData, species != c("CRP"))

##### If the dataset is subset by a factor,
##### we should to drop the unused factor
##### or else it R will give summaries, ect for that factor level
trawlNoCarp<- subset(trawlData, species != c("CRP"))
tapply(trawlNoCarp$fishLength, trawlNoCarp$species, mean)
# WHY THE NA FOR CARP?
trawlNoCarp$species<- trawlNoCarp$species[,drop=T]
tapply(trawlNoCarp$fishLength, trawlNoCarp$species, mean) # carp are gone!
```

Code breakdown

1. `subset`: subset returns the rows of the dataset where the group is equal to d
2. `&`: subset returns the rows of the dataset where the group is equal to d and x1 is greater than 5 (see table 3 for a list of legal operators)
3. `drop=T`: this portion of code within the brackets drops unused levels of the factor, which can be problematic when summarizing data

Table 3: Legal operators for subsetting dataframes, vectors, and matrices

Operator	Action
==	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal
%in%	within
&	and
|	or

5.6 Merging datasets

Often in fisheries type data there are values that need to be added to a dataset based on an identification. An example of this would be values of stock and quality length to calculate PSD for a fish population. This can be completed in R using the `merge()` function (R Code Chunk 26).

- `merge(dataset1, dataset2, by.x=mergeVarDataset2, by.y=mergeVarDataset2, all.x=T)`

R code chunk 26

```
# READ IN QUALITY LENGTH AND STOCK LENGTH TABLE FOR EACH FISH SPECIES
psdData<-read.csv("./psdData.csv")
psdData # look at the data
# THE FISH SPECIES CODE IS UNIQUE FOR EACH SPECIES AND
# MATCHES THE SPECIES CODES USED IN psdData
trawlData<- merge(trawlData, psdData, by.x="species", by.y="species",
all.x=TRUE)
head(trawlData) # dataset has the appropriate values needed to calculate PSD

trawlData$stockLength<-ifelse(trawlData$fishLength>trawlData$stockLength,1,0)
trawlData$qualityLength<-ifelse(trawlData$fishLength>trawlData$qualityLength,1,0)
overallSummary<- data.frame(
   spp = tapply(as.character(trawlData$commonName), trawlData$index,unique),
   year= tapply(trawlData$year, trawlData$index,unique),
   n_Qual = c(tapply(trawlData$qualityLength, trawlData$index,sum,na.rm=TRUE)),

   n_Stock = c(tapply(trawlData$stockLength, trawlData$index,sum,na.rm=TRUE)))
```

Code break down:

1. `by.x`: specifies the column name in `trawlData` that contains the values to merge on
2. `by.y`: specifies the column name in `psdData` that contains the values to merge on
3. `all.x=TRUE`: this lets R know that you want all the values of `trawlData` retained. Species where there is not a quality length and stock length specified in `psdData` will be assigned NA. If FALSE is specified then the resulting dataset will only contain values where there is a matching `lakeId` in `trawlData` and `psdData`.

5.7 In class exercise

1. Import lakeData.csv and catchData.csv from C:\contEdCourseData\part1 (this should already be your working directory, use `getwd()` to make sure).
2. Merge the individual lake data from lakeData.csv to catch data from catchData.csv based on lakeId
3. Make a summary table that includes the following: lakeId, lakeName, lakeArea, and the mean and variance of fish length and weight.
4. Export the table you your working directory as a *.csv file.

6 Visualizing data

R has a number of base graphics that can be used to explore your data, such as boxplots, scatterplots, and histograms. Plots can easily be saved as publication quality file formats (e.g., jpeg, wmf, eps), eliminating the need for graphics software (e.g., Sigma plot).

6.1 Boxplots

- `boxplot(Yvariable~Xvariable, dataset, col=" ", main=" ", xlab=" ", ylab=" ")`

R code chunk 27

```
boxplot(fishLength~lake, catchData)
boxplot(fishLength~lake, catchData, main="Boxplot of fish lengths")
boxplot(fishLength~lake, catchData, main="Boxplot of fish lengths",
xlab="Lake",ylab="Fish Length (mm)")

# ADD SOME COLOR
boxplot(fishLength~lake, catchData, main="Boxplot of fish lengths",
xlab="Lake",ylab="Fish Length (mm)",col="red")
# CAN WE SUBSET?
boxplot(fishLength~lake, catchData, main="Boxplot of fish lengths",
xlab="Lake",ylab="Fish Length (mm)",col="red", subset=lake !="Lake c")
```

Code breakdown

1. `col="red"`: fills each boxplot with red
2. `subset=lake !="Lake c"`: allows us to subset the dataset without having to make a new object

6.2 Histograms

- `histogram(variable, breaks, include.lowest=T`

R code chunk 28

```
##### A histogram of all the fish lengths
hist(trawlData$fishLength)
savePlot("./lengthFreq",type="jpg")

# SOME INTERVALS TO BIN FISH LENGTHS INTO
bins<- seq(0,600, by=25)
bins
# LENGTH FREQUENCY HISTOGRAM
hist(trawlData$fishLength, breaks=bins, include.lowest=T)
# A DENSITY PLOT
hist(trawlData$fishLength, breaks=bins, include.lowest=T, freq=F)
# LETS ADD A TITLE
hist(trawlData$fishLength, breaks=bins, include.lowest=T, freq=F,
main="Lenght frequency plot", xlab="Length (mm)",ylab="Density")
# LETS MAKE THE BARS BLACK
hist(trawlData$fishLength, breaks=bins, include.lowest=T, freq=F,
main="Length frequency plot", xlab="Length (mm)",ylab="Density",
col="black")
savePlot("./lengthFrequency",type="jpg")
```

Code breakdown

1. `seq()`: makes a sequence of numbers from 0 to 600 by a interval of 25
2. `include.lowest=T`: uses the sets up the bins such that a value goes into the interval if it is greater than or equal to.
3. `freq=f`: converts the plot from a frequency to a density plot
4. `col="black"`: fills the bars with black
5. `savePlot("./figure",type="jpg")`: saves the current plot to your working directory as a figure.jpg ("wmf", "pdf", "eps" are also allowed)

6.3 Barplots

- `barplot(height, width, names, col="")`
- `barplot(height, width, names, besides=T, col="")`

R code chunk 29

```
summaryTable
sum_2007<- subset(summaryTable, year==2007)
sum_2008<- subset(summaryTable, year==2008)
barplot(sum_2007$meanLength, names.arg=sum_2007$species)
barplot(sum_2007$meanLength, names.arg=sum_2007$species, col="blue")
# THE X LABELS ARE NOT THAT USEFUL, LETS ROTATE THEM
barplot(sum_2007$meanLength, names.arg=sum_2007$species, col="blue", las=2)
barplot(sum_2007$meanLength, names.arg=sum_2007$species, density=20, las=2)
```

Code breakdown

1. `las=2`: this will rotate the axis labels (can be between 1 and 4)

6.4 Scatterplots

A number of point and line types can be used in R graphics allowing the user great flexibility in producing figures of data (Figure 4).

R code chunk 30

```
plot(fishWeight~fishLength, trawlData)
# add a title
plot(fishWeight~fishLength, trawlData, main="Weight-Length for fish")
# add better axis labels
plot(fishWeight~fishLength, trawlData, main="Weight-Length for fish",
xlab="Length (mm)", ylab="Weight (g)")
# We can copy and paste this directly into MS Word

##### Lets add some grouping
plot(fishWeight~fishLength, trawlData, main="Weight-Length for fish",
xlab="Length (mm)", ylab="Weight (g)",type="n")
points(fishWeight~fishLength, trawlData,subset=species=="CRP",col="red")
points(fishWeight~fishLength, trawlData,subset=species=="YLB", col="blue")
points(fishWeight~fishLength, trawlData,subset=species=="WAL", col="green")
points(fishWeight~fishLength, trawlData,subset=species=="BBS", col="black")

##### Lets add some grouping with different points plot(fishWeight~fishLength,
trawlData, main="Weight-Length for fish", xlab="Length (mm)",
ylab="Weight (g)",type="n") points(fishWeight~fishLength,
trawlData,subset=species=="CRP",col="red",
   pch=1)
points(fishWeight~fishLength, trawlData,subset=species=="YLB", col="blue",
   pch=2)
points(fishWeight~fishLength, trawlData,subset=species=="WAL", col="green",
   pch=3) points(fishWeight~fishLength, trawlData,subset=species=="BBS",
col="black",
   pch=4)
```

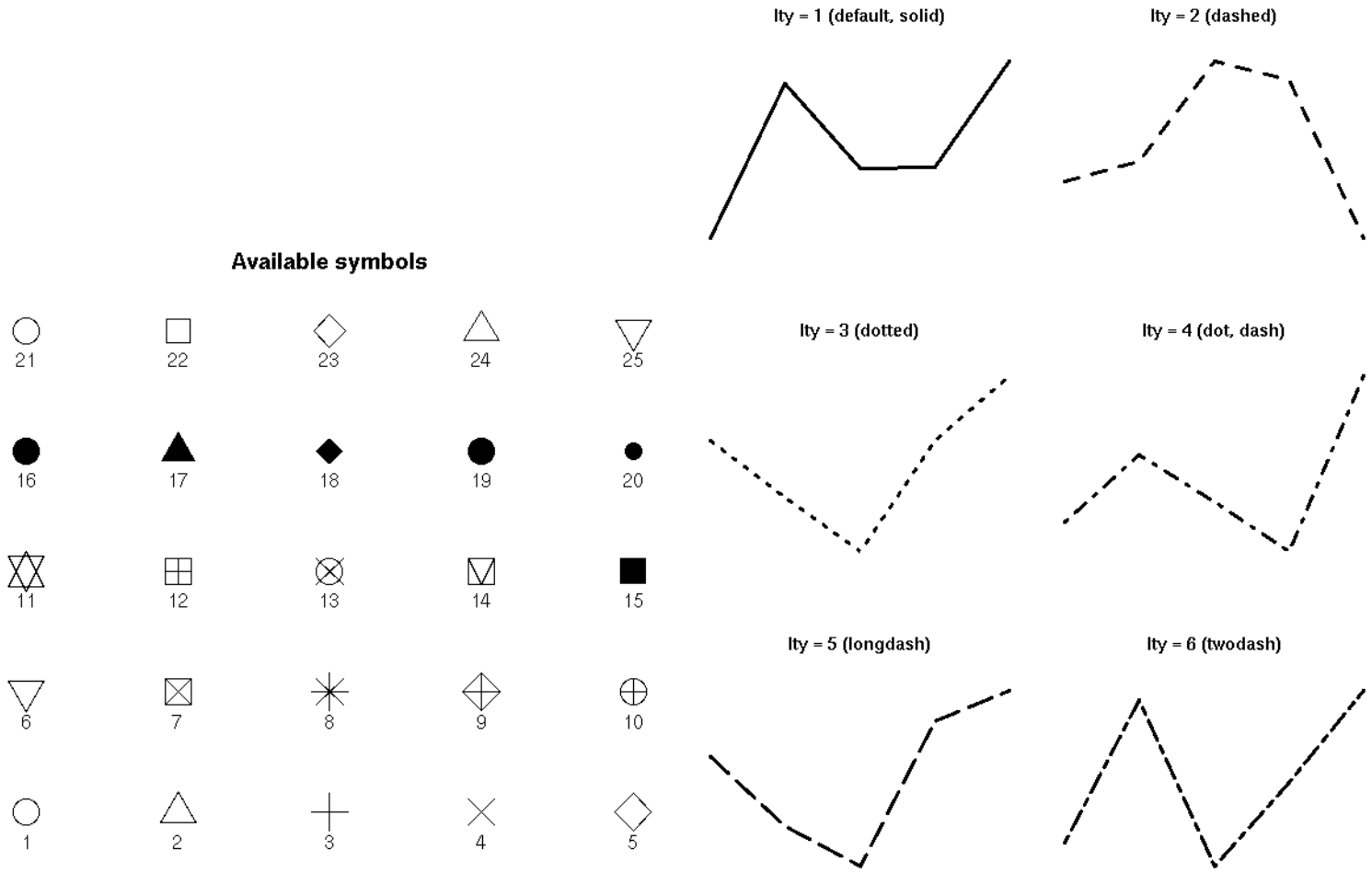

Figure 4: Available point and line types used in R.

Code breakdown

1. `plot(trawlData)`: plots the dataset as matrix of scatterplots, sometimes useful for initially evaluating a dataset
2. `main=" ", xlab=" ", ylab=" "`: adds customizable labels for the plot and axes
3. `type="n"`: plots all the data without points, this allows the plot to have the appropriate x and y limits
4. `points( )`: adds the a group to the empty plot

Code breakdown

6.4.1 Fine tuning scatter plots

R code chunk 31

```
# THE POINTS NEED TO BE BIGGER AND FILLED
# CEX IS A MAGNIFICATION FACTOR AND PCH 1-25 ARE ALL SORTS OF SYMBOLS
plot(fishWeight~fishLength, trawlData,cex=2,pch=19)
# MY BOSS WANTS BIGGER FONT FOR THE LABELS
# CEX.AXIS IS A MAGNIFICATION FOR THE AXIS FONT
# CEX.LAB IS A MAGNIFICATION FOR THE LABEL FONT
plot(fishWeight~fishLength,trawlData,cex=1.5,pch=19,
   cex.axis=1.1,cex.lab=1.2,xlab="Fish length(mm)")
# PERSONALLY, I DISLIKE HAVING THE NUMBERS ON THE YAXIS
# ORIENTED PARALLEL TO THE AXIS
# LAS=1 FIXES THAT
plot(fishWeight~fishLength,trawlData,cex=1.5,pch=19,
   cex.axis=1.1,cex.lab=1.2,xlab="Fish length(mm)",
   ylab="Weight (g)",las=1)
```

6.5 Legends

- `legend("location", listOfNames, listOfPoints,listOfColors)`

R code chunk 32

```
##### Lets add some grouping with different points plot(fishWeight~fishLength,
trawlData, main="Weight-Length for fish", xlab="Length (mm)", ylab="Weight
(g)",type="n")
points(fishWeight~fishLength,trawlData,subset=species=="CRP",col="red",
   pch=1)
points(fishWeight~fishLength, trawlData,subset=species=="YLB", col="blue",
   pch=2)
points(fishWeight~fishLength, trawlData,subset=species=="WAL", col="green",
   pch=3) points(fishWeight~fishLength, trawlData,subset=species=="BBS",
col="black",
   pch=4)
legend("topleft", c("Common carp", "Yellow bass", "Walleye", "Black
bullhead"),
   pch=c(1,2,3,4),col=c("red","blue", "green", "black"))
```

Code breakdown

1. `"topleft"`: other options can be: topleft, bottomright, bottomleft

7 Functions

R code chunk 33

```
add<-function(var1,var2,var3){var1+var2+var3}
add(4,6,20)

# A SUMMARY FUNCTION
var_summary<- function(var1,var2){
par(mfrow=c(2,1))
hist(var1);savePlot("./hist1",type="pdf")
hist(var2);savePlot("./hist2",type="pdf")
mean(var1, na.rm=T)
mean(var2, na.rm=T)
}

##### Use the function to summarize a dataset
var_summary(trawlData$fishLength,trawlData$fishLength)
```

Code breakdown

1. `function(var1,var2,var3)`: this specifies the variables needed for the function
2. `par(mfrow=c(2,1))`: sets a 2 row by 1 column plot
3. `{}`: the brackets contain the operations for the function
4. `var_summary(csv_dataset$y,csv_dataset$x1)`: applies the function

8 Some statistical functions

8.1 Linear model

The `lm()` function in R fits a linear model to data. We can fit a linear model to some data and call the output whatever we want, in this case I named the fitted model, "`fit`." In this case, `fit` is a list that contains fitted values ($\hat{Y}_i$), residuals (ε_i), and estimates for β_j . Once the model is fit was can get a summary of the model using `summary()` and an anova table using the `anova()` and `Anova()` functions.

R code chunk 34

```
require("car")
fit<- lm(fishWeight~fishLength ,trawlData) # fits a linear regression
plot(fit)# diagnostics

# LOG TRANSFORM VARIABLES
trawlData$lweight<- log(trawlData$fishWeight)
trawlData$llength<- log(trawlData$fishLength)
fit<- lm(lweight~llength ,trawlData)
plot(fit)
summary(fit) # returns information about the model fit
anova(fit)# returns an anova table
Anova(fit, type="III")
Y_hats<- fit$fitted # extracts the predicted values
residuals<- fit$resid # extracts the residuals
confint(fit)

# ADD A SPECIES INTERACTION
fit<- lm(lweight~llength*species ,trawlData)
plot(fit)
summary(fit) # returns information about the model fit
anova(fit)# returns an anova table
Anova(fit)
confint(fit)
```

Code breakdown

1. `lm()`: fits a linear model
2. `summary()`: summarizes the model
3. `anova()`: produces type I sums of squares anova table
4. `Anova()`: produces type II or III sums of squares anova table
5. `fit$fitted`: model predicted values
6. `fit$resid`: model residuals
7. `confint(fit)`: extracts 95% confidence intervals for the parameter estimates

8.2 Non linear models

R code chunk 35

```
# FIT A NON LINEAR REGRESSION
fit<- nls(fishWeigth~a*fishLength^b ,trawlData, start=list(a=0.0001, b=3))
summary(fit)
# A WEIGHT-LENGTH RELATIONSHIP FOR CARP
fit<- nls(fishWeigth~a*fishLength^b ,trawlData, start=list(a=0.0001,
b=3),subset=species=="CRP")
summary(fit)
```

8.3 In class exercise

1. Import the dataset walleye.csv from C:\contEdCourseData\part1
2. Plot a histogram of the length frequencies
3. Plot a scatterplot of the weight to length relationship
4. Log transform weight and length
5. Fit a linear model of log(length) and log(weight)
6. Using the same data fit a non linear model of weight-length assuming the form $W = a * L^b$, where W is weight and L is length.

8.4 Closing

The previous functions are but a few of the many functions that R has for statistical analysis. Other potentially useful functions include `glm()` for logistic and Poisson regression, `lme()` for mixed effects models, and `nlme()` for non linear mixed effects models.

Part II

Routine fisheries analysis

The following examples are largely from Guy and Brown (2007). The scope of this continuing education workshop is to present a potential tool for practicing fisheries professionals to use for routine fisheries analysis. Therefore interpretation is limited. For in depth interpretation of analysis results can be found in the chapter sections of Guy and Brown (2007).

9 Estimation of Z from population metrics

R code chunk 36 Estimate Z from population metrics (Hayes et al., 2007)

```
setwd("C:/contEdCourseData/part2")
lengths <- read.csv("./box6_8.csv")
head(lengths)
hist(lengths$len/10,xlab="Length (cm)")
bins<- seq(9,53, by=1)
h<-hist(lengths$len/10,breaks=bins, right=FALSE, xlab="Length (cm)")
Linf <- 636
K <- 0.226
meanLength <- mean(lengths$len)
minLength <- min(lengths$len)
Z_bevertonHolt <- K * (Linf - meanLength)/(meanLength - minLength)
Z_bevertonHolt
medLength <- median(lengths$len)
Ymedian <- -log(1 - medLength/Linf)
Ymin <- -log(1 - minLength/Linf)
Z_Hoenig <- 0.693 * K/(Ymedian - Ymin)
Z_Hoenig
```

10 Age and length

10.1 Age length key

R code chunk 37 Age length key (Isely and Grabowski, 2007).

```
setwd("C:/contEdCourseData/part2")
key <- read.csv("./box5_1.csv")
lengths <- read.csv("./box5_1_lengthData.csv")
str(lengths)
key$tlint<- ifelse(key$tl< 100, 90, 0)
key$tlint<- ifelse(key$tl>=100 & key$tl< 120, 100, 0)
key$tlint<- ifelse(key$tl>=120 & key$tl< 140, 120, 0)
key$tlint<- ifelse(key$tl>=140 & key$tl< 160, 140, 0)
key$tlint<- ifelse(key$tl>=160 & key$tl< 180, 160, 0)
key$tlint<- ifelse(key$tl>=180 & key$tl< 200, 180, 0)
key$tlint<- ifelse(key$tl>=200 & key$tl< 220, 200, 0)
key$tlint<- ifelse(key$tl>=220 & key$tl< 240, 220, 0)
key$tlint<- ifelse(key$tl>=240 & key$tl< 260, 240, 0)
key$tlint<- ifelse(key$tl>=260 & key$tl< 280, 260, 0)
key$tlint<- ifelse(key$tl>=280 & key$tl< 300, 280, 0)
key$tlint<- ifelse(key$tl>=300 & key$tl< 320, 300, 0)
key$tlint<- ifelse(key$tl>=320 & key$tl< 340, 320, 0)
key$tlint<- ifelse(key$tl>=340 & key$tl< 360, 340, 0)
key$tlint<- ifelse(key$tl>=360 & key$tl< 380, 360, 0)
key$tlint<- ifelse(key$tl>=380 & key$tl< 400, 380, 0)
key$tlint<- ifelse(key$tl>=400 & key$tl< 420, 400, 0)
key$tlint<- ifelse(key$tl>=420 & key$tl< 440, 420, 0)
key$tlint<- ifelse(key$tl>=440 & key$tl< 460, 440, 0)
key$tlint<- ifelse(key$tl>=460 & key$tl< 480, 460, 0)
key$tlint<- ifelse(key$tl>=480 & key$tl< 500, 480, 0)
key$tlint<- ifelse(key$tl>=500 & key$tl< 520, 500, 0)
key$tlint<- ifelse(key$tl>=520 & key$tl< 540, 520, 0)
# AN EASIER WAY, THIS DOES THE SAME THING AS THE PREVIOUS 23 LINES
bins<- c(0,seq(100,540, by=20))
labels.x<- c(90, seq(100,520, by=20))
key$tlint<-cut(key$tl, breaks = bins, right=FALSE, include.lowest=TRUE,labels=labels.x)
summaryTable <- table(key$tlint, key$age)
ageLengthKey <- prop.table(summaryTable , margin = 1)
ageLengthKey[which(is.na(ageLengthKey ))]<-0
ageLengthKey <- round(ageLengthKey, digits=3)
lengths$tlint<-cut(lengths $tl, breaks = bins, right=FALSE,
include.lowest=TRUE,labels=labels.x)
lengths<- subset(lengths, tl<520)
len.freq <- table(lengths$tlint)
dim(ageLengthKey)
length(len.freq)
# A LIST OF NUMBER OF FISH ASSIGNED TO EACH AGE GROUP
ageFrequency<-len.freq %*% ageLengthKey
ageFrequency
```

10.2 Back calculated length

R code chunk 38 Back calculated length (Isely and Grabowski, 2007).

```
setwd("C:/contEdCourseData/part2")
backCalc <- read.csv("./box5_2.csv")
str(backCalc)
head(backCalc)
backCalc$Si2 <- backCalc$Si*10/24
# convert radial measurements
backCalc$Sc2 <- backCalc$Sc*10/24
# convert total scale radius
backCalc$Li1 <- backCalc$Lc * backCalc$Si2/backCalc$Sc2

# Fraser-Lee Method
fit <- lm(Li1 ~ Si2, backCalc)
anova(fit)
a <- coef(fit)[1] # get the correction factor (first coefficient in lm1)
backCalc$Li2 <- a + (backCalc$Lc-a)*(backCalc$Si2/backCalc$Sc2)
```

11 Sampling and experimental design

11.1 Simple random sample

R code chunk 39 Estimating the age composition form a random sample of brown trout (Page 63 Hansen et al., 2007).

```
ageData<- data.frame(
   age=c(0,1,2,3,4,5,6,7),
   n=c(55,22,10,18,6,3,1,1))
totalNumberOfTrout<-sum(ageData$n)
ageProportion<- ageData$n/totalNumberOfTrout
# CALCULATED PROPORTION
seAgeProportion<- sqrt(ageProportion*(1-ageProportion)/(totalNumberOfTrout-1))

# STANDARD ERROR OF THE ESTIMATE
# FUNCTIONS TO CALCULATE CONFIDENCE INTERVALS
lowerCI<- function(ageCounts, alpha){
N<- sum(ageCounts)
lowerF<-qf(alpha,2*(N-ageCounts+1),2*ageCounts,lower.tail = FALSE)
ageCounts/(ageCounts+(N-ageCounts+1)*lowerF) }
upperCI<- function(ageCounts, alpha){
# TO RUN THIS FUNCTION INPUT A VECTOR OF AGE COUNTS
# AND AN ALPHA VALUE
N<- sum(ageCounts)
upperF<-qf(alpha,(2*ageCounts)+2,(2*(N-ageCounts+1)-2),lower.tail=FALSE)
((ageCounts+1)*upperF)/(N-ageCounts+(ageCounts+1)*upperF)
}# END FUNCTION
# END FUNCTIONS TO CALCULATE CONFIDENCE INTERVALS
# ENTER A VECTOR OF AGE COUNTS AND AN ALPHA TO CALCULATE CONFIDENCE LIMITS
lowerCL<-lowerCI(ageData$n,0.05)
upperCL<-upperCI(ageData$n,0.05)
# CREATE A SUMMARY TABLE
summaryTable<- data.frame(
age=ageData$age, proportion_age=ageProportion,
SE=seAgeProportion, LowerCL=lowerCL, UpperCL=upperCL)
summaryTable
write.csv(" ./summaryTableRandomSample.csv")
```

11.2 Simple stratified random sampling

R code chunk 40 Estimate the mean catch from a stratified random sample (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
srs<- read.csv("./box3_4.csv")
# CALCULATE MEAN AND VARIANCE OF THE STRATA
stratumSummary<- data.frame(
   stratum= tapply(as.character(srs$stratum),srs$stratum,unique),
   n= tapply(srs$Catch,srs$stratum,length),
   mean=tapply(srs$Catch,srs$stratum,mean),
   variance=tapply(srs$Catch,srs$stratum,var))
stratumSummary$weights<- c(0.2125,0.5375,0.250)
meanCatch<- sum(stratumSummary$weights*stratumSummary$mean)
meanCatachSE<- sqrt(sum((stratumSummary$weights^2*stratumSummary$variance)/
stratumSummary$n))
CI<- meanCatachSE*qt(0.975,df=nrow(srs)-3)
lowerCI<-meanCatch- CI
upperCI<-meanCatch+ CI
```

11.3 Cluster sample

R code chunk 41 Estimation of the mean weight of age 0+ bluegill from a cluster sample (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
cluster<- read.csv("./box3_5.csv")
cluster$countId<- ifelse(is.na(cluster$Weight)==TRUE,0,1)
summary<- data.frame(
   net= tapply(as.character(cluster$Net),cluster$Net,unique),
   catch= tapply(cluster$countId,cluster$Net,sum),
   clusterTotal=tapply(cluster$Weight,cluster$Net,sum,na.rm=T))
numberOfNets<- nrow(summary)
meanCatch<- sum(summary$catch)/numberOfNets
meanWeight<- sum(summary$clusterTotal)/sum(summary$catch)
meanWeightSE<- (1/(sqrt(numberOfNets)*meanCatch))* sqrt(
sum((summary$clusterTotal-meanWeight*summary$catch)^2)/(numberOfNets-1))
CI<- meanWeightSE*qt(0.975,df=numberOfNets-1)
lowerCI<- meanWeight-CI
upperCI<- meanWeight+CI
```

11.4 Systematic sample

R code chunk 42 Estimate the mean width of a stream based on a systematic sampling design (Hansen et al., 2007) .

```
setwd("C:/contEdCourseData/part2")
systematic<- read.csv("./box3_6.csv")
clusterTotals<- tapply(systematic$width,systematic$sample,sum)
samplesWithinCluster<- tapply(systematic$sample,systematic$sample,length)
nclusters<- length(tapply(systematic$sample,systematic$sample,unique))
meanWidth<- mean(systematic$width)
meanWidthSE<- 1/(sqrt(2)*15)* sqrt((sum((clusterTotals-meanWidth*15)^2))/
(nclusters-1))
```

11.5 Regression or double sampling

R code chunk 43 Estimate the mean coverage of woody debris based on calibrated visual estimates (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
regression<- read.csv("./box3_7.csv")
subset<- na.omit(regression)
measuredObs<- nrow(subset)
fit<- lm(Measured~Estimated,subset)
slope<- fit$coeff[2]
interecept<-fit$coeff[1]
xbar<- mean(subset$Estimated)
ybar<- mean(subset$Measured)
Xbar<- mean(regression$Estimated)
meanCoverage<- ybar+slope*(Xbar-xbar)
meanCoverageSE<- sqrt(1/(measuredObs*(measuredObs-2))*(sum((subset$Measured-ybar)^2)-
(sum((subset$Measured-ybar)*((subset$Estimated-xbar)^2))/
(sum((subset$Estimated-xbar)^2)))))
```

11.6 Completely randomized design

R code chunk 44 Analysis evaluates differences catches based on bag limits that were randomly assigned to lakes (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
crd<- read.csv("./box3_8.csv") # the data
require(car);require(pda) # packages needed for analysis
options(contrasts=c("contr.sum","contr.poly")) # this is needed to reproduce
SAS results
str(crd) crd$bag_limit<- as.factor(crd$bag_limit)# make bag limit a factor
fit <- lm(catch ~ bag_limit, crd) # model of the data
plot(fit) # plot model diagonstics
anova(fit) # provides type I sums of squares
Anova(fit, type="III") # provides type III sums of squares
lsmean(fit, level=0.05, pdiff=T) # evaluates differences in lsmeans
```

11.7 Diagnostics and assumptions

R code chunk 45 Diagnostics and assumptions Hansen et al. (2007).

```
setwd("C:/contEdCourseData/part2")
residuals<- residuals(fit) # extract residuals from the previous model
summary(residuals)
hist(residuals) # histgram of residuals
qqnorm(residuals) # plot of qqnorm
shapiro.test(residuals) # test for normality
ks.test(residuals, "pnorm") # ks test
boxplot(residuals~crd$bag_limit) # boxplot of residals by bag limit factor
abline(h=0,col="red") # add a red line to the previous plot
```

11.8 Randomized block design

R code chunk 46 Randomized block design (Hansen et al., 2007). Page 93

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda) ; require(gplots)
options(contrasts=c("contr.sum","contr.poly"))
rbd<- read.csv("./box3_10_2.csv")
rbd$bag_limit<- factor(rbd$bag_limit)
fit <- lm(catch ~ region*bag_limit, rbd)
Anova(fit , type="III")
lsmean(fit,factors="region")
lsmean(fit,factors="bag_limit")
# DROP THE INSIGNIFICANT INTERACTION TERM
fit <- lm(catch ~ region+ bag_limit, rbd)
Anova(fit, type="III")
lsmean(fit,factors="region", level=0.05, pdiff=T)
lsmean(fit,factors="bag_limit", level=0.05, pdiff=T)
interaction.plot(rbd$bag_limit,rbd$region, rbd$catch,type="b", xlab="Bag
Limit",ylab="Catch", trace.label ="Region")
# Plot Means with Error Bars
require(gplots)
plotmeans(catch~bag_limit,rbd, xlab="Walleye bag limit", ylab="Catch",
main="Mean Plot with 95% CI",las=1)
```

11.9 Analysis of covariance (ANCOVA)

R code chunk 47 Analysis of covariance (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda)
options(contrasts=c("contr.sum","contr.poly"))
ancova<- read.csv("./box3_11.csv")
plot(growth~egg_diameter, ancova)
points(growth~egg_diameter, ancova,subset=substrate=="Sand",col="red")
points(growth~egg_diameter, ancova,subset=substrate=="Gravel",col="blue")
points(growth~egg_diameter, ancova,subset=substrate=="Cobble",col="black")
fit<- lm(growth~egg_diameter*substrate, ancova)
summary(fit)
Anova(fit, type="III")
ancova$predicted<- fitted(fit)
# ADD FITTED LINES TO THE PREVIOUS PLOT
points(predicted~egg_diameter, ancova,subset=substrate=="Sand",
col="red",type="l")
points(predicted~egg_diameter, ancova,subset=substrate=="Gravel",
col="blue",type="l")
points(predicted~egg_diameter, ancova,subset=substrate=="Cobble",
col="black",type="l")
```

11.10 Factorial model

R code chunk 48 Factorial model anlaysis (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car); require(pda)
options(contrasts=c("contr.sum","contr.poly"))
factorial<- read.csv("./box3_13.csv")
factorial$survival<- asin(factorial$survival/100)
fit<- lm(survival~size*release, factorial)
Anova(fit, type="III")
lsmean(fit)
```

12 Contingency tables

R code chunk 49 Contingency table for differences in length frequency data (Neumann and Allen, 2007).

```
setwd("C:/contEdCourseData/part2")
meanLength <- read.csv("./box9_5.csv")
str(meanLength)
meanLength
x<-xtabs(num~year + lcat, meanLength)
chisq.test (x)
x<-xtabs(num~year + lcat, meanLength, subset=year %in% c(1996,1998))
summary(x)
x<-xtabs(num~year + lcat, meanLength, subset=year %in% c(1996,2000))
summary(x)
x<-xtabs(num~year + lcat, meanLength, subset=year %in% c(1998,2000))
summary(x)
bon.alpha<-0.05/3
```

13 Linear regression type analysis

13.1 Simple and multiple regression

R code chunk 50 Correlation simple regression and multiple regression (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest);require(agricolae);require(MPV)
options(contrasts=c("contr.sum","contr.poly"))
regression <- read.csv("./box4_5.csv")
regression str(regression)
plot(CPE0~WINSTAGE, regression)
plot(CPE0~WINRET, regression)
plot(CPE0~SPRSTAGE, regression)
cor(regression[,c(2,4,5,6)], method="pearson",use="pairwise.complete.obs")
fit1<- lm(CPE0~WINSTAGE +WINRET +SPRSTAGE, regression)
anova(fit1)
summary(fit1)
Anova(fit1,type="II")
vif(fit1)
fit2<- lm(CPE0~WINSTAGE+ SPRSTAGE, regression)
Anova(fit2,type="II")
fit3<- lm(CPE0~WINSTAGE, regression)
Anova(fit3,type="II")
```

13.2 Log linear model

R code chunk 51 Log linear model to test for year class abundance differences (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(agricolae)
options(contrasts=c("contr.sum","contr.poly"))
llinear<- read.csv("./box4_1.csv")
str(llinear)
llinear$YEARCL <- as.factor(llinear$YEARCL)
llinear$AGE<- as.factor(llinear$AGE)
fit<- lm(log(CATCH)~YEARCL + AGE, llinear)
anova(fit)
Anova(fit, type="III")
lsmean(fit, factors="YEARCL", pdiff=T, level=0.001786)
x<-pairwise.t.test(log(llinear$CATCH),llinear$YEARCL, paired=T,p.adj =
"bonf")
df<-df.residual(fit)
MSerror<-deviance(fit)/df
comparison <- LSD.test(log(llinear$CATCH),llinear$YEARCL,df,MSerror,
p.adj="none",alpha=0.00176, group=FALSE, main=" ")
```

13.3 Time series

R code chunk 52 Evaluation of time series trends in abundance (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest)
options(contrasts=c("contr.sum","contr.poly"))
timeSeries<- read.csv("./box4_2.csv")
str(timeSeries)
timeSeries<- subset(timeSeries, YEAR>1971)
plot(AGE0CPE~YEAR, timeSeries)
summary(timeSeries)
cor(timeSeries, method="pearson")
cor(timeSeries, method="kendall")
fit<- lm(AGE0CPE~YEAR, timeSeries)
summary(fit)
anova(fit)
x<-acf(residuals(fit))
plot(x)
x
dwtest(fit)# DURBIN WATSON TEST FOR AUTOCORRELLATION
```

13.4 Catch curve analysis

R code chunk 53 Catch curve regression to identify week and strong year classes (Hansen et al., 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest);require(agricolae);require(MPV)
options(contrasts=c("contr.sum","contr.poly"))
catchCurve <- read.csv("./box4_4.csv")
catchCurve$LNUM<- log(catchCurve$NUM+1)
catchCurve$LMEANRET<- log10(catchCurve$MEANRET)
fit<- lm(LNUM~AGE, catchCurve)
catchCurve$Weight<-fitted(fit)
fit<- lm(LNUM~AGE, catchCurve, weights=Weight)
summary(fit)
anova(fit)
catchCurve$PLUM<- fitted(fit)
catchCurve$PredictedSE<- predict(fit, se.fit =TRUE)$se.fit
catchCurve$residual<- residuals(fit)
catchCurve$rstudent<- rstudent(fit)
catchCurve$cookd<- cookd(fit)
catchCurve
sum(catchCurve$residual)
sum(catchCurve$residual^2)
plot(LNUM~AGE, catchCurve, xlab="Year-class",ylab="Natural log Number at
age", las=1)
points(PLUM~AGE, catchCurve, type="l")
```

13.5 Catch curve

R code chunk 54 Catch curve analysis (Miranda and Bettoli, 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest);require(agricolae);require(MPV)
options(contrasts=c("contr.sum","contr.poly"))
ccurve <- read.csv("./box6_4.csv")
str(ccurve)
ccurve
ccurve$lCatch<- log(ccurve$Catch)
fit<- lm(lCatch~Age, ccurve, subset=Age>=3)
anova(fit)
ccurve$w<- predict(fit,ccurve)
fit<- lm(lCatch~Age, ccurve, subset=Age>=3,weights=w)
anova(fit)
coef(fit)
Z <- -coef(fit)[2]
Z
confidenceIntervals<- -confint(fit, "Age")
# PLOT THE CATCH AND MODEL PREDICTED CATCH
plot(lCatch~Age, ccurve,xlab="Age",ylab="Log(Catch)")
points(fitted(fit)~c(3:10),type="l",lwd=2)
```

13.6 Catch curve variability

R code chunk 55 Catch curve variability (Miranda and Bettoli, 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest);require(agricolae);require(MPV)
options(contrasts=c("contr.sum","contr.poly"))
ccurve_ancova <- read.csv("./box6_6.csv")
str(ccurve_ancova)
ccurve_ancova$Lake<- as.factor(ccurve_ancova$Lake)
ccurve_ancova$lCatch<- log(ccurve_ancova$Catch)
fit<- lm(lCatch~ Age, ccurve_ancova, subset=(Age>=2 & Lake=="1"))
summary(fit)
anova(fit)
fit<- lm(lCatch~Age, ccurve_ancova, subset=(Age>=2 & Lake==2))
summary(fit)
anova(fit)
fit<- lm(lCatch~Age+ Lake + Age:Lake, ccurve_ancova, subset=Age>=2)
ccurve_ancova$pred<- predict(fit, ccurve_ancova)
summary(fit)
anova(fit)
Anova(fit, type="III")
plot(fit)

plot(lCatch~Age, ccurve_ancova, type="n",xlab="Age",ylab="Log(Catch)")
points(lCatch~Age, ccurve_ancova,subset=Lake==1, col="grey")
points(lCatch~Age, ccurve_ancova,subset=Lake==2, col="black")
points(pred~Age, ccurve_ancova,subset=(Lake==1 & Age>=2),
col="grey",type="l")
points(pred~Age, ccurve_ancova,subset=(Lake==2 & Age>=2),
col="black",type="l")
```

14 Non linear models

14.1 Condition

R code chunk 56 Weight-length data anlaysis (Pope and Kruse, 2007).

```
setwd("C:/contEdCourseData/part2")
condition <- read.csv("./box10_1.csv")
options(contrasts=c("contr.sum","contr.poly"))
str(condition)
head(condition)
plot(condition)
condition$lWT<- log10(condition$WT)
condition$lTL<- log10(condition$TL)
fit<- lm(lWT~lTL, condition, subset=pop=="a")
summary(fit);anova(fit);confint(fit)
fit<- lm(lWT~lTL, condition, subset=pop=="b")
summary(fit);anova(fit);confint(fit)
fit<- lm(lWT~lTL, condition, subset=pop=="c")
summary(fit);anova(fit);confint(fit)
fit<- nls(WT~a*TL^b, condition, subset=pop=="a", start=list(a=0.000001,
b=3))
summary(fit);confint(fit) summary(fit)$coefficients
fit<- nls(WT~a*TL^b, condition, subset=pop=="b", start=list(a=0.000001,
b=3))
summary(fit);confint(fit) summary(fit)$coefficients
fit<- nls(WT~a*TL^b, condition, subset=pop=="c", start=list(a=0.000001,
b=3))
summary(fit);confint(fit) summary(fit)$coefficients
fit<- lm(lWT~lTL*pop, condition)
summary(fit);anova(fit);confint(fit)
Anova(fit, type="III")
fit<- lm(lWT~lTL + pop, condition)
summary(fit);anova(fit);confint(fit)
Anova(fit, type="III")
```

14.2 Spawner recruit curve

R code chunk 57 Beaverton Holt spawner recruit curve (Page 151 Maceina and Pereira, 2007).

```
setwd("C:/contEdCourseData/part2")
require(car);require(pda);require(lmtest);require(agricolae);require(MPV)
options(contrasts=c("contr.sum","contr.poly"))
bvh <- read.csv("./box4_7.csv")
bvh$LRECRUIT<- bvh$RECRUIT
plot(RECRUIT~SPAWNER, bvh)
fit<- nls(RECRUIT~(a*SPAWNER)/(1+ b*SPAWNER), bvh, start=list(a=0.03,
b=0.002))
confint.default(fit)
fit1<- nls(RECRUIT~(SPAWNER)/(1+ b*SPAWNER), bvh, start=list( b=0.002 ))
confint.default(fit1)
```

Part III

Some additional useful R codes

15 AIC Model Selection

R code chunk 58

```
# WELL AIC IS POPULAR NOW WHAT ABOUT THOSE?
setwd("C:/contEdCourseData/extras")
coho<-read.csv("http://www.public.iastate.edu/~mcolvin/575x/coho.csv")
pairs(coho[,c(3:8)]) summary(coho)
coho.global<- lm(log(cohoDensity)~ streamId + maxTemp + naturalCover + slope
+ percentPool + riparianCover,coho)
# LETS EVALUATE SOME HYPOTHESES
M1<-lm(log(cohoDensity)~1,coho)
M2<-lm(log(cohoDensity)~1,coho)
M3<-lm(log(cohoDensity)~1,coho)
M4<-lm(log(cohoDensity)~1,coho)
M5<-lm(log(cohoDensity)~1,coho)
M6<-lm(log(cohoDensity)~1,coho)
M7<-lm(log(cohoDensity)~1,coho)
M8<-lm(log(cohoDensity)~1,coho)
M9<-lm(log(cohoDensity)~1,coho)
M10<-lm(log(cohoDensity)~1,coho)
# MAKE A DATAFRAME WITH MODEL AIC
selectionTable <- data.frame(
model=c("global",paste("M",1:10,sep="")),
aic=c(AIC(coho.global), AIC(M1), AIC(M2), AIC(M3), AIC(M4), AIC(M5),
AIC(M6), AIC(M7), AIC(M8), AIC(M9), AIC(M10)))
# CALCULATE THE AIC DIFFERENCE (DELTA AIC)
selectionTable$deltaAic<-selectionTable$aic- min(selectionTable$aic)
# CALCULATE THE MODEL LIKELIHOOD
selectionTable$lik<-exp(-0.5*selectionTable$deltaAic)
# CALCULATE THE RELATIVE MODEL WEIGHT
selectionTable$weight<-selectionTable$lik/sum(selectionTable$lik)
# REORDER THE DATAFRAME BY AIC (FOR EASE OF INTERPRETATION)
selectionTable<-selectionTable[order(selectionTable$aic),]
# LOOK AT THE SELECTION RESULTS selectionTable
```

16 Study area maps

R code chunk 59

```
# PLOTTING A SHAPEFILE USING THE MAPTOOLS PACKAGE
# install.packages("maptools")
setwd("C:/contEdCourseData/extras")
require(maptools)
clearLake<- readShapePoly("c:/gis/clearlakenad83.shp",proj4string=CRS("+proj=utm
+zone=15 +datum=NAD83"))
plot(clearLake,axes=T, las=1)
par(mar=c(5,5,1,1))
plot(clearLake,axes=T, las=1)
# LETS GET SOME POINTS TO PUT ON THE MAP
trawling<-read.csv("http://www.public.iastate.edu/~mcolvin/trawl.csv")
points( startUTMy~startUTMx, trawling)
# IT IS A LAKE, LETS MAKE IT BLUE
plot(clearLake,axes=T, las=1, col="blue")
points(startUTMy~startUTMx, trawling,pch=16)
savePlot("./trawlSites",type="pdf")
```

17 Matrix population model

R code chunk 60

```
# AN AGE OR STAGE MATRIX OF FECUNDITIES AND SURVIVALS
A<- matrix(c( 0,2,3,4, 0.1,0,0,0, 0,0.3,0,0, 0,0,0.4,0),nrow=4, byrow=T)
# A MATRIX TO HOLD THE FORECASTS OF THE POP
population<- matrix(0, nrow=4, ncol=100)
# SEED THE POPULATION
population[,1]<- c(400, 200, 100, 40)
# FORECAST THE STRUCTURED POPULATION
for(i in 2:100){
population[,i]<- A%*%population[,i-1]}
# CALCULATE THE TOTAL POPULATION AT EACH TIME STEP
totalPopulation<- apply(population, 2, sum)
plot(totalPopulation)
# LAMBDA FOR MATRIX A
lambda<-as.numeric(eigen(A)$values[1])
w<-as.numeric(eigen(A)$vectors[,1]);
w<-w/sum(w); # STABLE AGE DISTRIBUTATION FOR MATRIX A
v<-as.numeric(eigen(t(A))$vectors[,1]);
v<-v/v[1]; # REPRODUCTIVE VALUE FOR MATRIX A
s<-outer(v,w)/sum(v*w); # SENSITIVITY FOR MATRIX A
e<-(s*A)/lambda # ELASTICITIES FOR MATRIX A
```

18 For loops

R code chunk 61

```
##### For loops are good for doing repetitive calculations
values<- list(x=rep(0,100)) # makes a list to hold the values
names(values) # values within the list we can add values to
values # look at the list
for(i in 1:100){
values$x[i]<- values$x[i]+rnorm(1,0,0.2) # Add a random number with mean 0
}
values$x
```

Code breakdown

1. `for(i in 1:100)`: this specifies that the loop will do the operations for i being equal to 1 all the way to i being equal to 100
2. `values$x[i]<- values$x[i]+rnorm(1,0,0.2)`: the replaces the ith value of x with the value plus some
3. `{}`: the brackets contain the operations of the for loop

19 Solutions for in class examples

R code chunk 62

```
# Solutions # 3.1
mydataframe[4,6]
x<- mydataframe[5,2]
x

# 4.5
setwd("c:/")
setwd("C:/contEdCourseData/part1")
trawlData_csv<- read.csv("./trawlData.csv")
channel<- odbcConnectExcel("./trawlData.xls")
trawlData<- sqlFetch(channel, "data")
head(trawlData)
str(trawlData)

# 5.7
lakeData<- read.csv("./lakeData.csv")
catchData<- read.csv("./catchData.csv")
catchData<- merge(catchData, lakeData, by.x="lakeId", by.y="lakeId",
all.x=T)
summary<- data.frame(
lakeId= tapply(catchData$lakeId,catchData$lakeId, unique),
lakeName=tapply(catchData$lakeName,catchData$lakeId, unique),
lakeArea=tapply(catchData$lakeArea,catchData$lakeId, unique),
mean=tapply(catchData$fishLength,catchData$lakeId, mean),
var=tapply(catchData$fishLength,catchData$lakeId, var))
write.csv("./summary.csv")

# 8.3
walleye<- read.csv("./walleye.csv")
hist(walleye$length)
plot(weight~length, walleye)
walleye$loglength<- log(walleye$length)
walleye$logweight<- log(walleye$weight)
fit<- lm(logweight~loglength,walleye) fit
fit<-nls(weight~a*length^b,walleye, start=c(a=0.000002, b=3))
confint(fit)
```

References

Colvin, M., E. Katzenmyer, T. W. Stewart, and C. L. Pierce, 2008. The clear lake ecosystem simulation model (clesm) project. Technical report, Iowa State University.

Colvin, M., E. Katzenmyer, T. W. Stewart, and C. L. Pierce, 2009. The clear lake ecosystem simulation model (clesm) project. Technical report, Iowa State University.

Guy, C., and M. Brown. 2007. Analysis and interpretation of freshwater fisheries data. Analysis and interpretation of freshwater fisheries data, Bethesda, MD.

Hansen, M. J., T. D. B. Jr., and D. B. Hayes, 2007. Sampling and experimental design. Pages 51–120 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.

Hayes, D., J. Bence, T. Kwak, and B. Thompson, 2007. Abundance, biomass, and production. *in* C. Guy and M. Brown, editors. Analysis and interpreation of freshwater fisheries data. American Fisheries Society, Bethesda.

Isely, J. J., and T. B. Grabowski, 2007. Age and growth. Pages 187–228 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.

Maceina, M. J., and D. L. Pereira, 2007. Recruitment. Pages 121–186 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.

Miranda, L. E., and P. W. Bettoli, 2007. Mortality. Pages 229–278 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.

Neumann, R. M., and M. S. Allen, 2007. Size structure. Pages 375–422 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.

Pope, K. L., and C. G. Kruse, 2007. Condition. Pages 423–472 *in* C. Guy and M. Brown, editors. Analysis and interpretation of freshwater fisheries data. American Fisheries Society, Bethesda, MD.