Session 5: Functions, Iteration and Branching (continued) Classic Models & Simulation

Foundations of Quantitative Ecology (EEOB 8896.11)

Paul J. Hurtado
(hurtado.10@mbi.osu.edu)

Mathematical Biosciences Institute (MBI)
The Ohio State University

Last compile: September 18, 2013

BBS Summary Data

Exercise:

Download the BBS Data from the course website, and start a new project directory.

Write code that iterates through bird species, and if the name contains "Warbler", for example, plot the trend data.

Modify your code to write the plots to files.

For coding hints, see the latter half of `week4_in-class-scratch.R`

Hints...

```
# Contents of week4_in-class-scratch.R
# Load the data
aoucodes <- read.csv("BBS_aou_codes.csv")
BBSdata <- read.csv("BBS_trends_ALL.csv")
BBSdata <- merge(aoucodes, BBSdata); names(BBSdata)

## [1] "aou"      "name"     "binomial" "Year"     "Index"    "CI.025"
## [7] "CI.975"

which(BBSdata$name=="Northern Cardinal")

## integer(0)
```

The problem?

1. BBSdata names are padded by extra "white space" (spaces/tabs)
2. the names aren't strings in a big matrix, but are instead `factors` (?factor for details). R encodes species names with numbers, to save memory, so proceed carefully!

Data Frames, Factors, and Levels

Let's take a closer look at the data frame BBSdata:

```
class("Bird Name")  # Strings are of class 'character'

## [1] "character"

class(BBSdata$name)  # Names are character strings, right? WRONG!

## [1] "factor"

typeof(BBSdata$name)  # Memory storage type used?

## [1] "integer"

apply(BBSdata, 2, class)[1:5]  # Now they're all 'character'?

##         aou        name    binomial        Year       Index
## "character" "character" "character" "character" "character"
```

Data Frames, Factors, and Levels

Let's take a closer look at the data frame BBSdata:

```
class("Bird Name")  # Strings are of class 'character'

## [1] "character"

class(BBSdata$name)  # Names are character strings, right? WRONG!

## [1] "factor"

typeof(BBSdata$name)  # Memory storage type used?

## [1] "integer"

apply(BBSdata, 2, class)[1:5]  # Now they're all 'character'?

##         aou        name    binomial        Year       Index
## "character" "character" "character" "character" "character"
```

What's going on here!?

Data Frames, Factors, and Levels

R stores *strings* in data frames as *numbers*, along with a look-up table of which strings correspond to which numbers. `factor()` formats data this way. `levels()` accesses the list of strings encoded by integers.

Data Frames, Factors, and Levels

R stores *strings* in data frames as *numbers*, along with a look-up table of which strings correspond to which numbers. `factor()` formats data this way. `levels()` accesses the list of strings encoded by integers.

```
set.seed(1); data=sample(1:3,10,replace=T); data

##  [1] 1 2 2 3 1 3 3 2 2 1

fdata <- factor(data, labels=c("A","B","C")); fdata

##  [1] A B B C A C C B B A
## Levels: A B C

levels(fdata)

## [1] "A" "B" "C"

levels(fdata) <- c("a","b","c"); fdata

##  [1] a b b c a c c b b a
## Levels: a b c
```

Most stats/data routines prefer factors.

Data Frames, Factors, and Levels

So `levels()` lets us access the "lookup table" of string values

```
levels(BBSdata$name)[263:266]

## [1] "  Nelson's Sparrow              " "  Northern Bobwhite             "
## [3] "  Northern Cardinal             " "  Northern Goshawk              "
```

otherwise, R *usually* converts things to the appropriate output type

```
BBSdata$name[263:266]

## [1]   Glaucous-winged Gull            Glaucous-winged Gull
## [3]   Glaucous-winged Gull            Glaucous-winged Gull
## 428 Levels:   Abert's Towhee                 ...
```

BBS Summary Data

So to remove the white space before and after names...

```
head(levels(BBSdata$name), 2)  # same as levels(BBSdata£name)[1:2]

## [1] "  Abert's Towhee                 " "  Acadian Flycatcher           "

levels(BBSdata$name) <- gsub("^\\s+", "", levels(BBSdata$name))
levels(BBSdata$name) <- gsub("\\s+$", "", levels(BBSdata$name))
levels(BBSdata$binomial) <- gsub("^\\s+", "", levels(BBSdata$binomial))
levels(BBSdata$binomial) <- gsub("\\s+$", "", levels(BBSdata$binomial))
```

yields names without the extra padding...

```
head(levels(BBSdata$name), 3)  # head(x,n)

## [1] "Abert's Towhee"     "Acadian Flycatcher" "Acorn Woodpecker"

levels(BBSdata$binomial)[1:3]  # same as x[1:n]

## [1] "Accipiter cooperii" "Accipiter gentilis" "Accipiter striatus"
```

BBS Summary Data

Example *pseudocode* for creating figures for subsets of birds

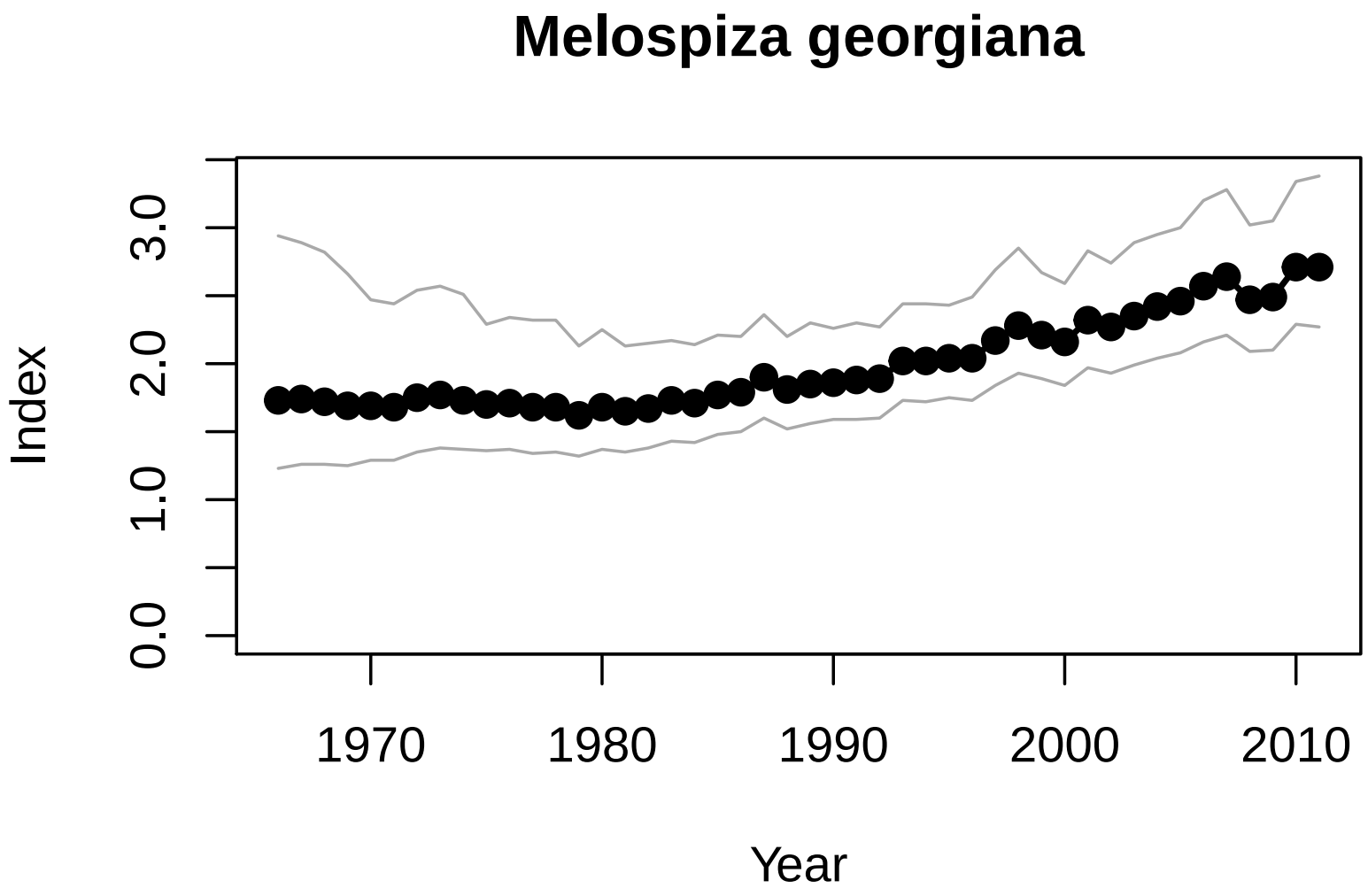
```
###### Option A ######
# Load Data
# birdlist = subset of birds
# Iterate over birdlist
#    Plot(bird)

###### Option B ######
# Load Data
# Iterate over all birds
#    if(plot this bird ? == Yes)
#        Plot(bird)
```

Let's do both!

BBS Summary Data: Option A

```
# Data already loaded.
# birdlist = subset of birds # Use unique() or levels()?
birdlist = grep("Melospiza",levels(BBSdata$binomial), value=TRUE)
# Iterate over birdlist
#    Plot(bird)
for(i in 1:length(birdlist)) { # for(bird in birdlist) works too!
  bird = birdlist[i];
  plot(Index~Year, data=subset(BBSdata,binomial==bird),
      pch=19, type="o",lwd=2, ylim=c(0,max(CI.975)), main=bird)
  points(CI.025~Year, data=subset(BBSdata,binomial==bird),
        pch=19, type="l",lwd=1, col="darkgray")
  points(CI.975~Year, data=subset(BBSdata,binomial==bird),
        pch=19, type="l",lwd=1, col="darkgray")
}
```

BBS Summary Data: Option A



BBS Summary Data: Option B

```
# Iterate over all birds
#    if(plot this bird ? == Yes)
#        Plot(bird)
for(bird in levels(BBSdata$binomial)) {
  if(grepl("Melospiza",bird)) {
    print(bird) # which birds get plotted? Output to screen
    plot(Index~Year, data=subset(BBSdata,binomial==bird),
        pch=19, type="o",lwd=2, ylim=c(0,max(CI.975)), main=bird)
    points(CI.025~Year, data=subset(BBSdata,binomial==bird),
          pch=19, type="l",lwd=1, col="darkgray")
    points(CI.975~Year, data=subset(BBSdata,binomial==bird),
          pch=19, type="l",lwd=1, col="darkgray")
  } # end if(grepl(...))
}
```

BBS Summary Data

Writing output to files can be done a number of ways.

Exercise: Modify the code above so that one writes the output figures to PDFs, and the other to PNGs.

Matrices

For the rest of class, go through the chapter on Matrices (and work through the exercises) in the Lab Manual (PDF) for the book Dynamic Models in Biology. (Please download this to your own computer instead of just viewing it in your browser!)