big data from the internet with R

env/bio 665 Bayesian inference for environmental models

Jim Clark

2020-01-15

Contents

resources

data

- download Breeding bird survey, explanation and data, with supporting data, but only for North Carolina or get specific BBS files from `Sakai/Resources/data`:
    - `BBSexampleTime.rdata`
- setup of a directory structure `../dataFiles/dataBBS/`
    - place BBS files in `/dataBBS`

software

- R2jags, must be installed

```
source('../clarkFunctions2020.r')
```

readings

Tidy Data, Wickam on concepts that link models and data to algorithms to R, *J Stat Soft*.

Connectivity of wood thrush breeding, wintering, and migration sites, Stanley et al on tracking wood thrush populations combined with BBS *Cons Biol*.

objectives

- gain familiarity with R
  - syntax for files, directories, filenames, paths
  - Interacting with files through `list.file, read.table, read.csv, readLines, load, source, write.table, save`
  - Interacting with objects and storage modes `numeric, character, factor, list, date, formula`
  - Interacting with functions: arguments, default values
  - transform a data file into observations by variables (OxV)
- learn to model with factors
- survive a (semi) big data example, including input, data exploration, and application to the wood thrush decline
- write a formula and graph for a basic GLM
- basics of exploratory data analysis (EDA)
- implement a Bayesian model applied to count data

the migratory wood thrush

The migratory wood thrush population is declining. There are multiple threats, both here in the temperate zone and in their tropical wintering grounds in Central America. Stanley et al. (2015) quantified declines in wood thrush abundance in the breeding range by region, and they related it to forest cover. They used tracking data to determine if regional differences in the US might be explained by the fact that birds summer and wintering grounds were linked. I used this example to examine changes in abundance using the breeding-bird survey (BBS) data. Specifically, we will ask whether or not wood thrush populations could be in decline in NC and, if so, which variables in the BBS data might help us understand it.

Figure 1: Wood thrush populations may be declining

I introduced R without much explanation in unit 1. Here I'll say a bit more to get us started on the BBS data, then let the data themselves lead the transition to data exploration and modeling.

why R?

Data manipulation remains the most time-consuming element of analysis. The challenges posed by data can halt progress long before model analysis begins. The popularity of R begins with the flexibility it provides for data structures and management.

A user can interact with R at a range of levels. High-level interactions with transparent functions like `lm` (linear regression) require little more than specification of predictors and a response variable. At the same time, R allows for low-level algorithm development. In fact, functions written in C++ and Fortran can be compiled and used directly with R. This combination of high- and low-level interaction allows one to enter at almost any degree of sophistication, then advance, without requiring a change in software.

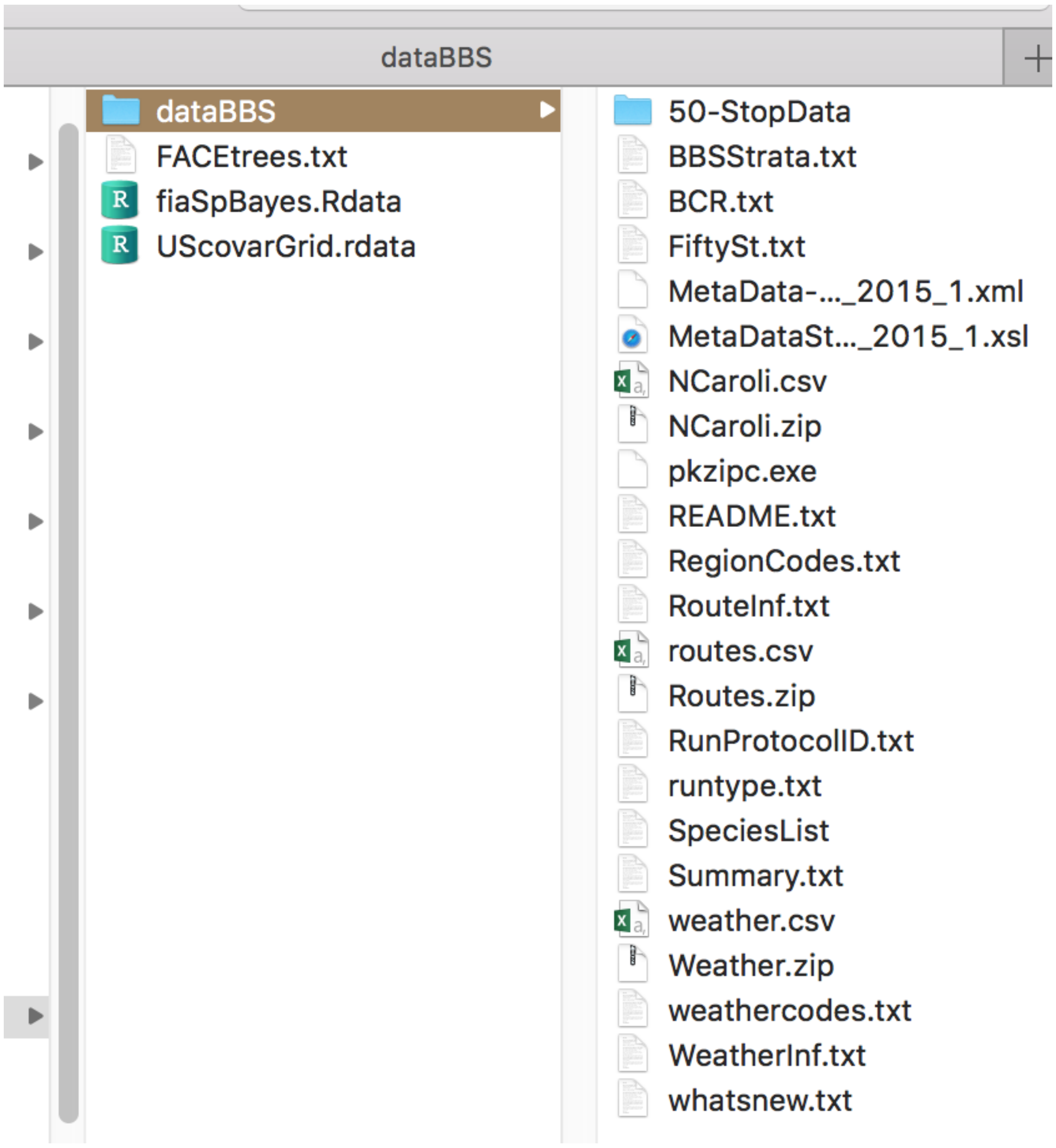
interacting with R

If you have not previously used R, this is a good time to work through the vignette on basicR. From there I move on to an example with the BBS data.

where am I?

When I open R studio I am in a directory. I can interact directly with files that are in my working directory. To interact with a file in a different directory, I need to either supply a `path` to the directory that it occupies, or I set my working directory using the `Session` tab in Rstudio.

I can determine which directory I am in with the function `getwd()`. I can move to a different director using `setwd(pathToThere)`. Below is a screenshot of my directories:


dataBBS

dataBBS
FACEtrees.txt
fiaSpBayes.Rdata
UScovarGrid.rdata

50-StopData
BBSStrata.txt
BCR.txt
FiftySt.txt
MetaData-..._2015_1.xml
MetaDataSt..._2015_1.xsl
NCaroli.csv
NCaroli.zip
pkzipc.exe
README.txt
RegionCodes.txt
RouteInf.txt
routes.csv
Routes.zip
RunProtocolID.txt
runtype.txt
SpeciesList
Summary.txt
weather.csv
Weather.zip
weathercodes.txt
WeatherInf.txt
whatsnew.txt

Figure 2: Current directory.

Here I am:

```
getwd()
```

```
## [1] "/Users/jimclark/Box/Home Folder jimclark/Private/classes/bayesClass2020spring/2d
```

Syntax is standard unix/linux. I move down a directory by giving a path with the directory name followed by `/`. I can move up directory by starting the path with `".."`. For example, the path to a file in the directory `dataFiles` (see screenshot above) is up, then down:

```
setwd("../dataFiles/")           # go there
setwd("../2data/")               # back here
```

where's the file?

As discussed above, files are identified by their location, or `path`, and the file name. The `path` can be relative to the current. To determine which files are in my current directory, I use `list.files()`. This returns a `character vector` of file names. To find the names of files in a different directory I need to include the `path`. This code works for me, because the paths match the locations of directories relative to my current directory:

```
list.files()                          # files in current directory
```

```
##  [1] "2data.docx"                     "2data.html"
##  [3] "2data.pdf"                      "2data.Rmd"
##  [5] "jagsData.Rdata"                 "jagsWoodThrush.txt"
##  [7] "mastifOutput.Rmd"               "poisBayesGraph.jpg"
##  [9] "poisBayesGraph.pdf"             "poisBayesGraph.pptx"
## [11] "ScreenShotBBS.png"              "ScreenShotDirectories.png"
## [13] "ScreenShotDirectoriesOld.png" "tmp.html"
## [15] "tmp.md"                         "tmp.r"
## [17] "wood_thrush.jpg"
```

```
list.files("../dataFiles/")           # in dataFiles
```

```
## [1] "dataBBS"          "FACEtrees.txt"    "fiaSpBayes.Rdata"
```

```
list.files("../dataFiles/dataBBS")  # in dataFiles/dataBBS
```

```
##  [1] "50-StopData"
##  [2] "BBSexampleTime.rdata"
##  [3] "BBSStrata.txt"
##  [4] "BCR.txt"
##  [5] "FiftySt.txt"
##  [6] "MetaData-NorthAmericanBreedingBirdSurvey_Dataset_1966-2015_version_2015_1.xml"
##  [7] "MetaDataStyleSheet-NorthAmericanBreedingBirdSurvey_Dataset_1966-2015_version_20
##  [8] "NCaroli.csv"
##  [9] "NCaroli.zip"
## [10] "pkzipc.exe"
## [11] "README.txt"
## [12] "RegionCodes.txt"
## [13] "RouteInf.txt"
## [14] "routes.csv"
## [15] "Routes.zip"
## [16] "RunProtocolID.txt"
## [17] "runtype.txt"
## [18] "Summary.txt"
## [19] "weather.csv"
## [20] "Weather.zip"
## [21] "weathercodes.txt"
## [22] "WeatherInf.txt"
## [23] "whatsnew.txt"
```

What's in the file

With so many file formats, ingesting data files has become almost an art. The BBS data confront us with `.txt` and `.csv` files, both of which can be straightforward, but not always. There are now dozens of file formats that can be loaded into R, but each can present challenges. An internet search will often locate an R package that reads a specific file type. Some files may require modification, either to make them readable or to extract information they contain that is not extracted by the package I find to read them. The R function `readLines` allows the desperate measure of reading files line-by-line. This is useful when, say, a `.txt` file contains non-ASCII characters. However, this is no help at all for many file types. One of the most flexible editors appears to be atom, freely available on the internet. Some of the challenges are presented by the BBS data.

Here I load observations in two objects, `xdata` (predictors) and `ydata` (bird counts):

```
load("../dataFiles/dataBBS/BBSexampleTime.rdata") #load xdata, ydata

xdata[1:5,]
```

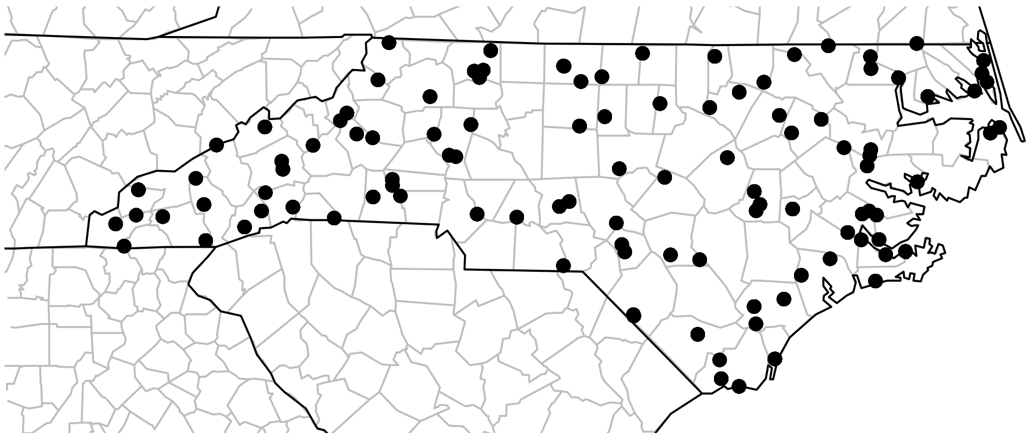
```
##     insert groups times       lon      lat Route      soil   nlcd year temp
## 1-0      1      1     0 -78.37957 33.98146     1 Spodosols forest 1964   NA
## 1-1      0      1     1 -78.37957 33.98146     1 Spodosols forest 1965   NA
## 1-2      0      1     2 -78.37957 33.98146     1 Spodosols forest 1966 24.4
## 1-3      0      1     3 -78.37957 33.98146     1 Spodosols forest 1967 26.9
## 1-4      0      1     4 -78.37957 33.98146     1 Spodosols forest 1968 23.1
##     StartWind StartSky precSite precAnom defSite defAnom winterTmin juneTemp
## 1-0        NA       NA     1250    314.0      71  -103.0       9.11     25.6
## 1-1        NA       NA     1250    -58.2      71   -79.2      10.20     24.5
## 1-2         2        2     1250     74.3      71   -16.3       8.02     23.8
## 1-3         2        2     1250   -115.0      71   -75.0       9.20     24.1
## 1-4         2        2     1250   -314.0      71    88.7       7.03     25.4
```

```
ydata[1:5,]
```

```
##     MourningDove Red-belliedWoodpecker ChimneySwift EasternWood-Pewee BlueJay
## 1-0            0                     0            0                 0       0
## 1-1            4                     1            1                 0       4
## 1-2          12                      5            4                 2      16
## 1-3          51                      5            4                 8      19
## 1-4          15                      5            6                 4      27
##     AmericanCrow EuropeanStarling CommonGrackle ChippingSparrow EasternTowhee
## 1-0            0                0             1               0             0
## 1-1            2                3             8               0             4
## 1-2            8                0            70               0            27
## 1-3           16                6            41               0            34
## 1-4           16                8            46               1            45
##     NorthernCardinal IndigoBunting Red-eyedVireo BrownThrasher CarolinaWren
## 1-0                0             0             0             0            0
## 1-1                5             3             1             0            5
## 1-2               11            11            12             2           21
## 1-3               25             5            10             2           16
## 1-4               40            11             3             2           18
##     TuftedTitmouse CarolinaChickadee WoodThrush AmericanRobin
## 1-0              0                 0          0             0
## 1-1              1                 2          1             0
## 1-2             12                 1          0             1
## 1-3              4                 0          0             0
## 1-4              2                 3          2             0
```

Here is a map of the routes:

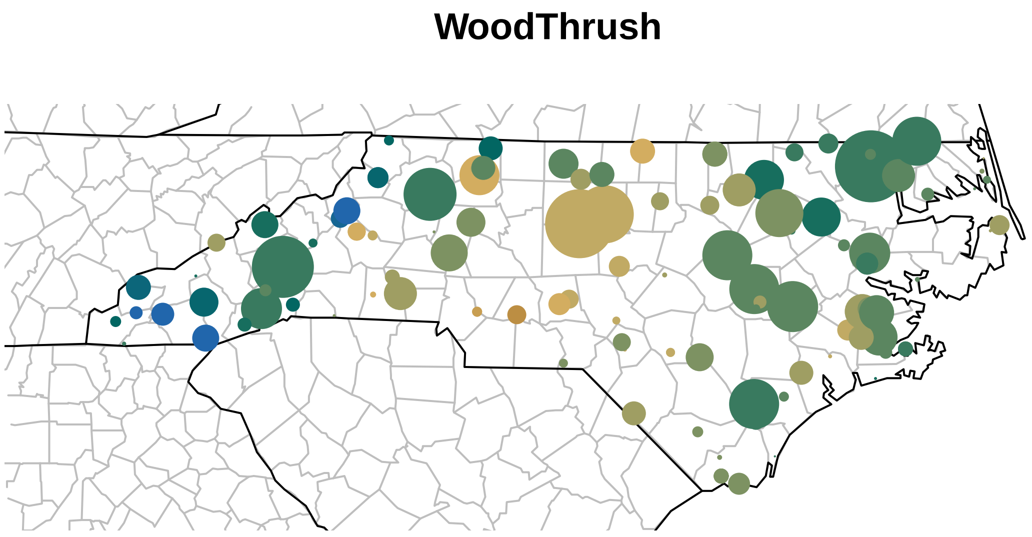
```
maps::map('county', xlim = c(-85, -75), ylim = c(33.6, 36.8), col='grey')
maps::map('state', xlim = c(-85, -75), ylim = c(33.6, 36.8), add=T)
points(xdata$lon, xdata$lat, pch = 16, cex = 1)
```



I now examine counts for wood thrush:Z

```
def <- xdata$defSite
def <- (def - mean(def, na.rm=T))/sd(def, na.rm=T)

df <- seq(-3, 3, length=20)
```

```r
colT <- colorRampPalette( c('#8c510a','#d8b365','#01665e','#2166ac') )

cols <- rev( colT(20) )

di <- findInterval(def, df, all.inside = T)


spec <- 'WoodThrush'

maps::map('county', xlim = c(-85, -75), ylim = c(33.6, 36.8), col='grey')
maps::map('state', xlim = c(-85, -75), ylim = c(33.6, 36.8), add=T)
cex <- 5*ydata[,spec]/max(ydata[,spec], na.rm=T)
points(xdata$lon, xdata$lat, pch = 16, cex = cex, col = cols[di] )
title(spec)
```



Files downloaded from the BBS site.

As mentioned previously, files we will use are located on Sakai. Familiarize yourself with the BBS sampling protocols here. The data can be downloaded from here. I download most of files, but limit the geographic data files to the state of North Carolina.

OxV format and the `data.frame`

Data are often stored in formats that are not amendable to analysis; reformatting is required. For analysis I typically want a `matrix` or `data.frame` with observations as rows and variables as columns, **OxV** format. Here are terms I use:

terms	definition
observation	what I record at a site/location/plot, numeric or not

terms	definition
sample	observations-by-variables `matrix` or `data.frame`
design matrix	observations-by-variables `matrix`, any factors converted to indicators
covariate	a predictor that takes continuous values
factor	a discrete predictor that can have 2, 3, ... *levels*
level	a *factor* has at least 2
main effect	an individual predictor
interaction	a combination of more than one predictor

The `list` that I called `data` looks like a `matrix`. I find the dimensions of `data` with a call to `dim(data)` = []. These two numbers are rows and columns, respectively. Although `data` has rows and columns, it is not stored as a R `matrix`, because it may include factors or even characters. A `data.frame` is rendered as rows and columns for user, but it is not stored as a matrix and, thus, does not admit matrix operations. The columns of a `data.frame` must have the same lengths, but they do not need to be of the same `mode`. Although I cannot do numeric operations across columns of a `list`, a `data.frame` is useful for structuring data. If all elements of a `data.frame` are `numeric`, then the `data.frame` is converted to a `numeric matrix` using the function `as.matrix`.

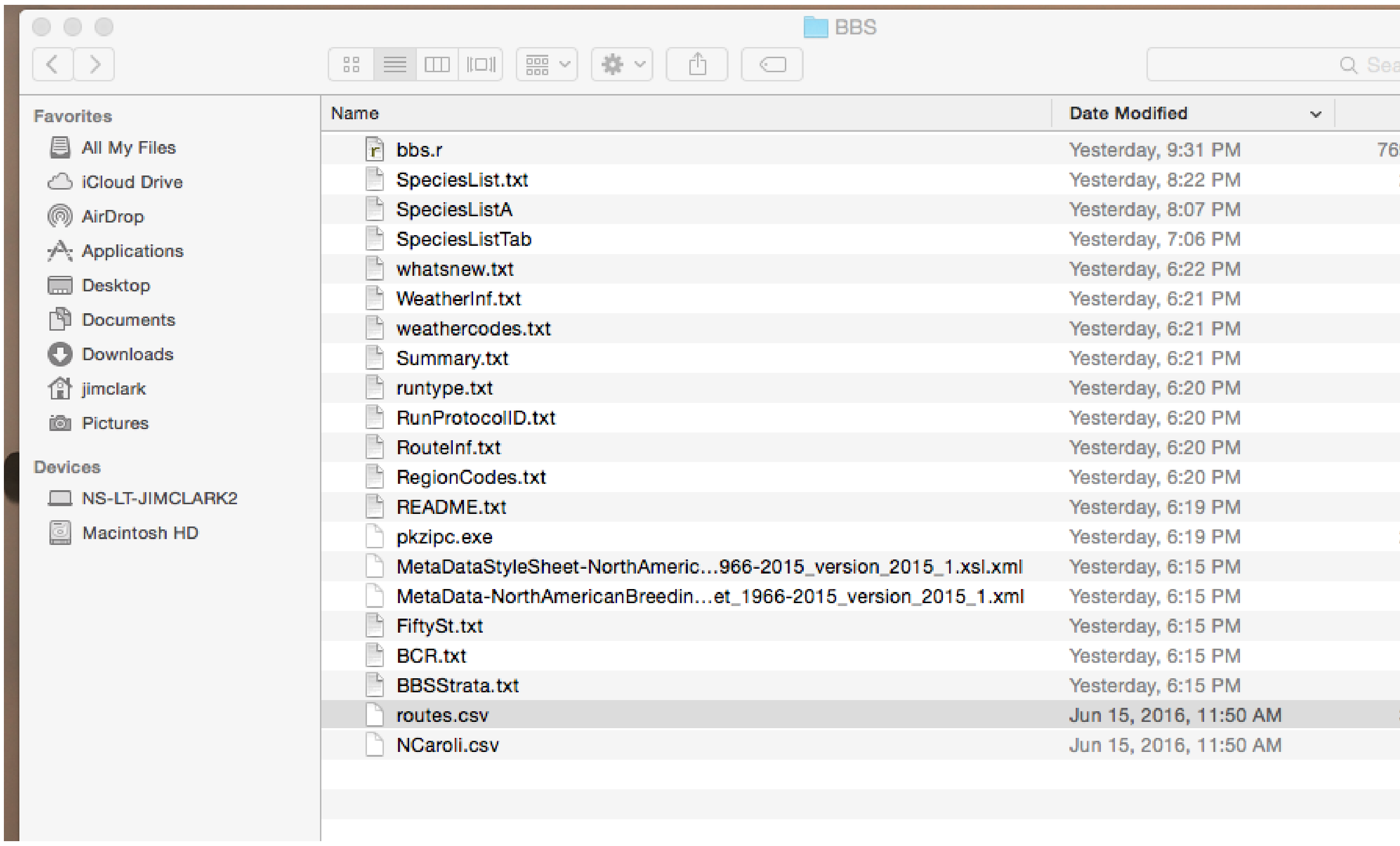
This is a good time to mention a new object in R, the `tribble`, see a description here `??tribble`. Some `data.frame` behaviors can be troublesome if you are not familiar with them. These will come up as we proceed. Because there is so much code available that uses the `data.frame`, I will continue to use it here.

Be careful when changing the dimension of a `matrix`. For example, I might construct a matrix like this:

```r
nd <- 3
z <- matrix(1:nd, 2*nd, 2)
colnames(z) <- paste('c',1:2,sep='')
rownames(z) <- paste('r',1:(2*nd),sep='')
z
```

```
##    c1 c2
## r1  1  1
## r2  2  2
## r3  3  3
## r4  1  1
## r5  2  2
## r6  3  3
```

I provided `1:nd` = 3 elements to fill a matrix that holds 6 rows and 2 columns. This did not generate an `error` or `warning`, because the number of elements in the matrix is a multiple of 3. This behavior can simplify code, but it can also create bugs when this is not what I want and I am not given a `warning`.



Name	Date Modified
bbs.r	Yesterday, 9:31 PM
SpeciesList.txt	Yesterday, 8:22 PM
SpeciesListA	Yesterday, 8:07 PM
SpeciesListTab	Yesterday, 7:06 PM
whatsnew.txt	Yesterday, 6:22 PM
WeatherInf.txt	Yesterday, 6:21 PM
weathercodes.txt	Yesterday, 6:21 PM
Summary.txt	Yesterday, 6:21 PM
runtype.txt	Yesterday, 6:20 PM
RunProtocolID.txt	Yesterday, 6:20 PM
RouteInf.txt	Yesterday, 6:20 PM
RegionCodes.txt	Yesterday, 6:20 PM
README.txt	Yesterday, 6:19 PM
pkzipc.exe	Yesterday, 6:19 PM
MetaDataStyleSheet-NorthAmeric…966-2015_version_2015_1.xsl.xml	Yesterday, 6:15 PM
MetaData-NorthAmericanBreedin…et_1966-2015_version_2015_1.xml	Yesterday, 6:15 PM
FiftySt.txt	Yesterday, 6:15 PM
BCR.txt	Yesterday, 6:15 PM
BBSStrata.txt	Yesterday, 6:15 PM
routes.csv	Jun 15, 2016, 11:50 AM
NCaroli.csv	Jun 15, 2016, 11:50 AM

Figure 3: Downloaded files are shown in a directory.

is the wood thrush declining?

To explore this question I need to structure the BBS data. An observation includes information (`Route`, `Year`) and the `count` for each species that is seen or heard. It includes a description of the weather, which affects behavior. From the table above, a **sample** is an observations-by-variables `matrix` or `data.frame`. It is the fundamental unit for analysis; I draw inference from a **sample**. This format is **OxV** ('obs by variables') format. The **design matrix** is a matrix. In R it is stored as a `matrix`, which is numeric. It contains predictors, which can be main effects and interactions. It can include covariates and factors. It can be analyzed, because it is `numeric`.

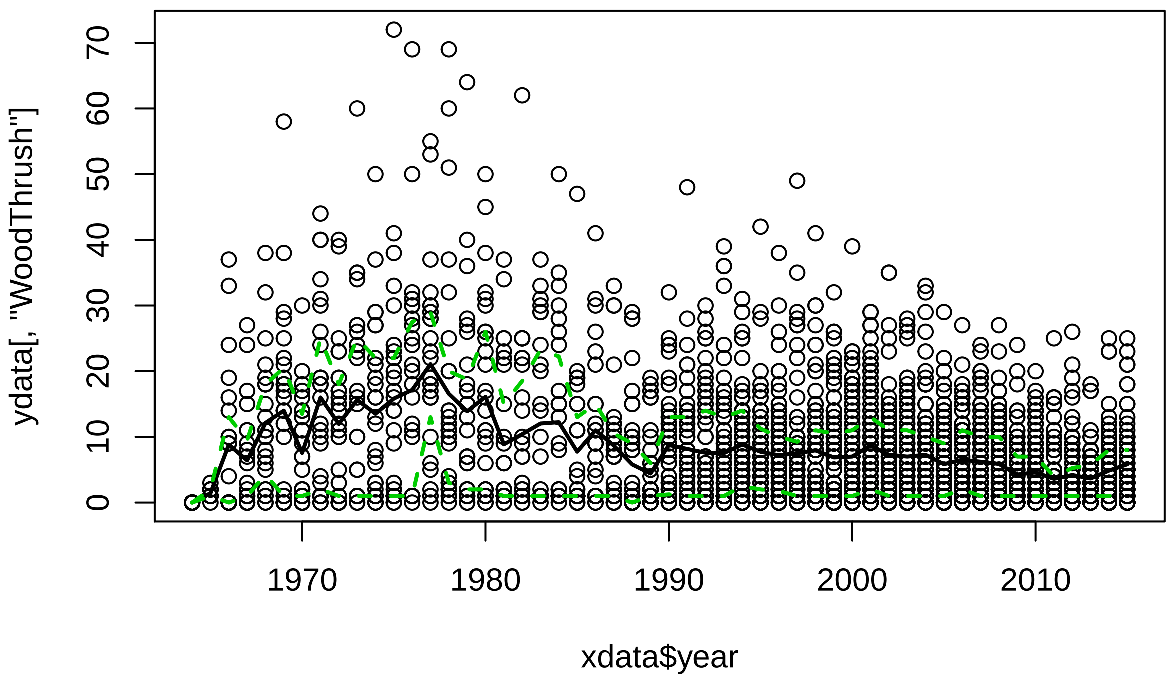
If **factors** are included in the `data.frame`, they should be declared using the function `factor`. If the factors are identified as words in a `data.frame`, then you will be interpreted that way. Confusion can arise when a factor is represented by integer values, e.g., $1, 2, \ldots$; it will be analyzed as a factor only if I declare it with `factor(variable)`. When the `data.frame` is converted to a design `matrix`, the `factor` levels each occupy a column of zeros and ones (more on this later).

Many data sets are not organized as OxV, often for good reasons, but they must be put in OxV format for analysis. The data may not be stored as OxV, because it can be inefficient. In the BBS data, an *observation* consists of the counts of all species at a stop–it includes everything the observer recorded at that stop. Species not seen are zero. Rather than enter many zeros in columns for all of the species not seen, the BBS data offers one line per species–a single observation occupies many rows in `data`.

Second, OxV format may be hard to see. For viewing/editing data I prefer to see entire rows. When I can see the full width of an object I can find columns, check formatting, correct errors, and so forth. If BBS data were stored as OxV, I would have to scroll across hundreds of columns. If the number of variables is so large that I cannot see it, I might want to format it differently. I do so at the risk of creating more work when I analyze it. The object `data` is easy to read, because I see entire rows, e.g., with `head(data)` and `tail(data)`.

If some variables are repeated across multiple observations they might be stored separately. This practice not only reduces size and, thus, storage, but also reduces errors that can enter simply due to redundancy. In the object `data`, the column `Route` could be treated as a variable (a 'factor' or group to be treated as a 'random effect'), but it could also be an indicator for other variables. Any attribute of a `Route` that does not change through time can be recorded once and then retrieved using the index represented by `Route`. It can be an index that allows us to link it with information in other files. To put it in OxV I need to assemble this outside information into the proper rows.

```r
plot(xdata$year, ydata[,'WoodThrush'])



# route by year
thrush <- tapply(ydata[,'WoodThrush'], list( route = xdata$Route, year = xdata$year), me
# average by year

thrushYear <- colMeans( thrush, na.rm=T)

year <- as.numeric(names(thrushYear))

lines(year, thrushYear, lwd=2)

thrushQuant <- apply( thrush, 2, quantile, na.rm=T)
lines(year, thrushQuant[2,], lty=2, lwd=2, col = 3)
lines(year, thrushQuant[4,], lty=2, lwd=2, col = 3)
```


ydata[, "WoodThrush"]
xdata$year