STATS 401. Applied Statistical Methods II

Welcome!

Objectives: Linear statistical models are the foundation for most of applied statistics. We will develop statistical computation skills (R programming) and mathematical skills (working with matrices) while studying data analysis using linear models.

Pre-requisites: We will assume familiarity with material in STATS 250. All course notes and labs are at

`open.umich.edu/find/open-educational-resources/statistics`

If you have a different background (AP Statistics, STATS 280, or some other introductory statistics class) you should check the STATS 250 notes and if necessary come for help in office hours.

Let's get started

We will work through a data analysis using a linear model, and then study the math and stats so that (i) we can command the computer to generate what we want; (ii) we can interpret what the computer tells us.

- Obtain the data from the internet
- Install R (`www.r-project.org`) and Rstudio (`www.r-project.org`)
- Read the data into R
- Plot the data
- Develop a model
- Estimate parameters and test hypotheses of interest
- Interpret the results

The two rising stars in statistical computing are R and Python (`http://r4stats.com/articles/popularity/`). Generally, R is preferred for data analysis, and Python for larger programming projects.

Let's get started

We will work through a data analysis using a linear model, and then study the math and stats so that (i) we can command the computer to generate what we want; (ii) we can interpret what the computer tells us.

- Obtain the data from the internet
- Install R (`www.r-project.org`) and Rstudio (`www.r-project.org`)
- Read the data into R
- Plot the data
- Develop a model
- Estimate parameters and test hypotheses of interest
- Interpret the results

The two rising stars in statistical computing are R and Python (`http://r4stats.com/articles/popularity/`). Generally, R is preferred for data analysis, and Python for larger programming projects.

We live in an era of abundant data. Learn R!

Case study: Are people healthier in booms or busts?

- Is population health **pro-cyclical** (improving in business cycle booms) or **counter-cyclical** (improving in recessions), or neither?
- **Life expectancy at birth** combines instantaneous death rates at all ages and is a basic measure of current population health.
- USA data for 1933–2015 are in the file `life_expectancy.txt` on the course GitHub repository `github.com/ionides/401w18/01` or the website `ionides.github.io/401w18/01`. The first lines of this file are:

```
# The United States of America, Life expectancy at birth.
# Downloaded from Human Mortality Database on 30 Oct 2017.
# HMD request that you register at http://www.mortality.org
# if you use these data for research purposes.
  Year       Female    Male      Total
  1933        62.78    59.17     60.88
  1934        62.34    58.34     60.23
```

- Note: # denotes a comment in R, so the first four text lines will be ignored when we read in the data.

Read the data into R and then inspect it

```
L <- read.table(file="life_expectancy.txt",header=TRUE)
```

Question: Why should we prefer to use the command line form of R rather than a menu option, say in R Commander?

Now, let's check on the data. To see the first three rows,

```
L[1:3,]
```

```
##   Year Female  Male Total
## 1 1933  62.78 59.17 60.88
## 2 1934  62.34 58.34 60.23
## 3 1935  63.04 58.96 60.89
```

Here, we're using **matrix indexing**. `L[i,j]` is the row `i` column `j` entry of `L`. Also, `1:3` is the sequence `1,2,3` and the blank space after the comma in `L[1:3,]` requests all the rows for the specified columns.

Matrices and their dimensions

Mathematically, we write $L = \begin{bmatrix} \ell_{11} & \ell_{12} & \dots & \ell_{1n} \\ \ell_{21} & \ell_{22} & \dots & \ell_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{m1} & \ell_{m2} & \dots & \ell_{mn} \end{bmatrix}$.

We say L is a matrix with **dimension** $m \times n$. To get the dimension in R,

```
dim(L)
```

```
## [1] 83  4
```

We can also get the number of rows and columns separately,

```
cat("number of rows = ", nrow(L),
  "; number of columns = ", ncol(L))
```

```
## number of rows =  83 ; number of columns =  4
```

Vectors

A single row or column of a matrix is called a **vector**. For example, we can set `y` to be total life expectancy, combining men and women,

```
y <- L[,4]
y[1:3]


## [1] 60.88 60.23 60.89
```

Question: We read the assignment operator `<-` as "`y` gets `L[,4]`". We could have written `y=L[,4]`. However, `<-` is slightly better coding practice than `=`. Why?

Mathematically, we write $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} L_{14} \\ L_{24} \\ \vdots \\ L_{m4} \end{bmatrix}$

We see that y is a $m \times 1$ matrix. We call y a **column vector**. A row of L is a $1 \times n$ matrix called a **row vector**.

Vectors in R

For R, vectors are not matrices. The dimension of length 1 is dropped, and the vector has a `length` but not a `dim`.

```
dim(y)


## NULL
```

```
length(y)


## [1] 83
```

We can extract the components of a vector. For example, to obtain the increase in life expectancy each year over the previous year,

```
g <- y[2:length(y)] - y[1:(length(y)-1)]
```

Since the increase is not defined for the first year life expectancy is measured, let's set

```
g <- c(NA,g)
g[1:8]


## [1]    NA -0.65  0.66 -0.54  0.70  1.34  0.68  0.16
```

Note: here we've seen two of R's special non-numeric values. `NULL` means "doesn't exist". `NA` means "not available" or "missing". Data matrices can have `NA` entries but not `NULL`. R tries to treat missing data appropriately.

Numeric, logical and character data in R

Numeric data are matrices and vectors whose entries are numbers.
Qualitative data are **character strings**. **Logical data** are `TRUE` or `FALSE`.

```
g[1:4]

## [1]    NA -0.65  0.66 -0.54
```

```
L_up_logical <- g>0
L_up_logical[1:4]

## [1]    NA FALSE  TRUE FALSE
```

```
L_up_qualitative <- ifelse(g>0,"increased","decreased")
L_up_qualitative[1:4]


## [1] NA           "decreased" "increased" "decreased"
```

The `class` function tells us what data type R is working with

```
class(g)

## [1] "numeric"
```

```
class(L_up_logical)

## [1] "logical"
```

```
class(L_up_qualitative)

## [1] "character"
```

Getting help with R

Learning a computing language is sometimes frustrating. Please proceed in the following order

1. The R help, e.g., type `?ifelse`) for information on the syntax of `ifelse`.
2. The internet, e.g., google "R ifelse".
3. Classmates.
4. Office hours, start-and-end of class, lab
5. Email to instructor and/or GSI.

For detailed email help, please construct and email a simple example demonstrating the issue. Sometimes, the issue gets resolved by writing it out!

R data structures: dataframes and matrices

- A matrix in R must have all entries of the same type. The mathematics of fitting a linear statistical model will require type to be numeric.
- For example, to convert data to a numeric representation for statistical analysis, `L_up_logical` or `L_up_qualitative` could be coded using `0` for `FALSE` (or `"decreased"`) and `1` for `TRUE` (or `"increased"`).
- A dataframe in R may have different types in each column. Data are usually stored in dataframes, e.g., `read.table()` generates a dataframe.

```
class(L)

## [1] "data.frame"
```

```
L_matrix <- as.matrix(L)
class(L_matrix)

## [1] "matrix"
```

- For many purposes, dataframes and matrices behave the same.

Innuit have many words for snow (`wikipedia:Eskimo_words_for_snow`) and R has many ways of working with data. To do effective data analysis, these are worth learning!

Subsetting matrices and vectors in R

- Vectors and matrices can be subsetted using logical vectors. Each entry of a vector (or row/column of a matrix) is included if the logical vector is `TRUE` and excluded if `FALSE`.
- Rows and columns can be selected using row and column names:

```
colnames(L)
```

```
## [1] "Year"   "Female"
## [3] "Male"   "Total"
```

```
rownames(L)[1:8]
```

```
## [1] "1" "2" "3" "4"
## [5] "5" "6" "7" "8"
```

Example: What is computed below. Can you find any interpretation?

```
L[g<0,"Year"]
```

```
##  [1]   NA 1934 1936 1943 1957 1960 1962 1963 1966 1968 1980
## [12] 1985 1988 1993 2015
```

Building matrices and vectors in R

The `c()` function concatenates numbers into vectors, and also concatenates vectors into longer vectors.

```
u <- c(1,2)
u

## [1] 1 2
```

```
v <- c(3,4)
v

## [1] 3 4
```

```
w <- c(u,v)
w

## [1] 1 2 3 4
```

We can build a matrix using `matrix()`. Also, we can get a matrix by binding together vectors either as rows or columns.

```
A <- matrix(1:6,nrow=2)
A

##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
B <- rbind(u,v)
B

##   [,1] [,2]
## u    1    2
## v    3    4
```

```
C <- cbind(u,v)
C

##      u v
## [1,] 1 3
## [2,] 2 4
```

Exercises. What would `cbind(A,B)` produce? Play with these functions. Check out `?matrix` to get the syntax of this command.

Continuing our health economics case study

We looked at data on mortality. We'll use Bureau of Labor Statistics data on unemployment as a measure of the business cycle.

```
# Data extracted on: February 4, 2016
# from http://data.bls.gov/timeseries/LNU04000000
# Percent unemployment, age 16+, not seasonally adjusted
Year,Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
1948,4.0,4.7,4.5,4.0,3.4,3.9,3.9,3.6,3.4,2.9,3.3,3.6
1949,5.0,5.8,5.6,5.4,5.7,6.4,7.0,6.3,5.9,6.1,5.7,6.0
```

```
U <- read.table(file="unemployment.csv",sep=",",header=TRUE)
U[1:2,]
```

```
##   Year Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1 1948   4 4.7 4.5 4.0 3.4 3.9 3.9 3.6 3.4 2.9 3.3 3.6
## 2 1949   5 5.8 5.6 5.4 5.7 6.4 7.0 6.3 5.9 6.1 5.7 6.0
```

Note: the data are in a comma separated variable (csv) format, so we use `read.table(...,sep=",",...)`.

Averaging columns in R

We want annual average unemployment. For each row, we must average columns 2:13.

```
u <- apply(U[,2:13],1,mean)
u[1:6]
```

```
## [1] 3.766667 5.908333 5.325000 3.333333 3.033333 2.925000
```

- `apply()` is a useful function for manipulating data matrices. Learn to use it!
- The middle argument `1` to `apply()` asks for the function `mean()` to be applied to each row.
- Setting `2` would give the average over rows for each column.
- Remember: `apply(U,1,...)` gives a vector of length `dim(U)[1]`, and `apply(U,2,...)` gives a vector of length `dim(U)[2]`.

```
dim(U)
```

```
## [1] 68 13
```

```
length(apply(U,1,mean))
```

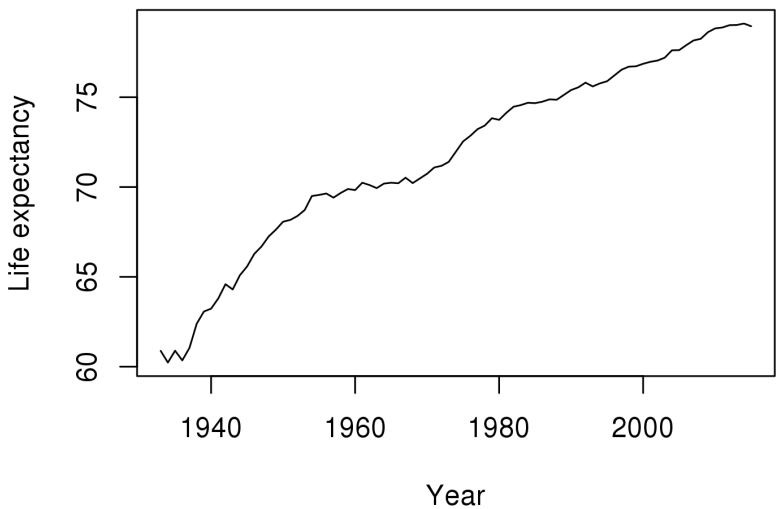
```
## [1] 68
```

```
length(apply(U,2,mean))
```

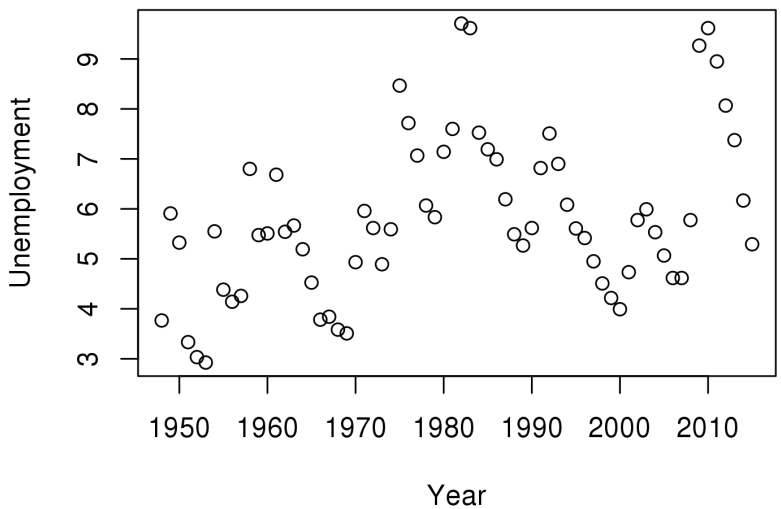
```
## [1] 13
```

Plotting the data

```
plot(L$Year,y,type="line",
  xlab="Year",
  ylab="Life expectancy")
```




```
plot(U$Year,u,
  xlab="Year",
  ylab="Unemployment")
```



- A basic rule of applied statistics is to plot the data.
- Carefully designed plots can reveal secrets in the data: (i) label axes; (ii) lines or points or both; (iii) any other creative ideas?
- This course will use the basic `plot()` function. A powerful modern approach to graphics is the "grammar of graphics" in the `ggplot2` package, taught in STATS 306.

Detrending life expectancy

Life expectancy shows an **increasing trend**. We're interested in whether it is above or below trend.