# Introduction to R for Epidemiologists

Jenna Krall, PhD

Thursday, January 22, 2015

# Review: class policies

- **Please read the course syllabus, located on the course website**
- There is no blackboard page for the class
- Questions:
  1. Visit office hours
  2. E-mail instructor
  3. **Do not e-mail TAs**

# Review: lab policies

- Lab counts for 20% of your grade
- To receive full credit you **must be present in lab**
- You do not need to submit any lab materials
- If you will miss lab for any reason you must see me **prior to your absence**
- Please review posted lab solutions after each week's lab

# Review: homework policies

- Homework 1 is posted
- You must submit a .R file to me by email
- Due by midnight on Wednesday 2/4
- You may work with other students, but your submitted assignment **must be your own work**

# Outline

1. Subsetting data
2. Missing data
3. Matrices and data frames
4. Naming objects
5. Summary statistics
6. Dates
7. Lists
8. Sorting data
9. Creating new variables

# Subsetting vectors in R

Last class, we dealt with vectors in R, which we created as:

```r
x <- c("item 1", "item 2", "item 3", "item 4")
```

We can subset vectors (select elements) using brackets:

```r
x[2]
```

```
## [1] "item 2"
```

We can also use the combine function to select multiple elements:

```r
x[c(1, 3, 4)]
```

```
## [1] "item 1" "item 3" "item 4"
```

# Subsetting vectors in R

Or a sequence using a colon

```
1 : 3
```

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

```
x[1 : 3]
```

```
## [1] "item 1" "item 2" "item 3"
```

We can also use the same techniques to remove items

```
x[-c(1, 4)]
```

```
## [1] "item 2" "item 3"
```

```
x[-(1 : 3)]
```

```
## [1] "item 4"
```

# Subsetting vectors in R

The `which` function is also useful for selecting items:

```r
which(x == "item 2")
```

```
## [1] 2
```

```r
x[which(x == "item 2")]
```

```
## [1] "item 2"
```

```r
newvector <- c(2, 5, 2, 6, 7, 6, 4, 10)
which(newvector < 5)
```

```
## [1] 1 3 7
```

```r
newvector[which(newvector < 5)]
```

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

# Subsetting vectors in R

We have introduced relational operators

```r
x <- 5
y <- 5
x < y
```

```
## [1] FALSE
```

```r
x <= y
```

```
## [1] TRUE
```

```r
x != y
```

```
## [1] FALSE
```

# Subsetting vectors in R

```
newvector
```

```
## [1]  2  5  2  6  7  6  4 10
```

```
newvector >= 5
```

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

```
which(newvector >= 5)
```

```
## [1] 2 4 5 6 8
```

```
newvector[which(newvector >= 5)]
```

```
## [1]  5  6  7  6 10
```

```
newvector[newvector >= 5]
```

```
## [1]  5  6  7  6 10
```

# Missing data

- NA is an R symbol used to denote missing values
- When we take the mean of vectors including missing values, we need to remove those missing values

```
vector1 <- c(2, 5, NA, 10, NA, 1, 1, 2.5, 9, 2)
mean(vector1)
```

```
## [1] NA
```

```
mean(vector1, na.rm = TRUE)
```

```
## [1] 4.0625
```

# Missing data

We can use `is.na` to determine which elements are missing

```
is.na(vector1)
```

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

```
!is.na(vector1)
```

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

```
which(is.na(vector1))
```

```
## [1] 3 5
```

```
vector1[is.na(vector1)]
```

```
## [1] NA NA
```

# Matrices and data frames

- Rows = observations, columns = variables
- Matrices: all numeric variables
- Data frames: mix of numeric, string/character, factor variables

Fisher's Iris dataset

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
class(iris)
```

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

# Matrices and data frames

```r
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1
```

# Matrices and data frames

We can subset data frames like vectors using brackets

We use commas to indicate which rows and columns we want (e.g. [rows, columns]):

```
iris[c(1, 3), 1 : 2]
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 3          4.7         3.2
```

We can also use column names to subset data frames

```
head(iris$Sepal.Length)
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

```
iris[1 : 6, "Sepal.Length"]
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

# Matrices and data frames

From the R datasets package:

- State statistics for the US from the 1970s

```r
head(state.x77)
```

```
##            Population Income Illiteracy Life Exp Murder HS Grad Frost
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20
## Alaska            365   6315        1.5    69.31   11.3    66.7   152
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65
## California      21198   5114        1.1    71.71   10.3    62.6    20
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166
##              Area
## Alabama     50708
## Alaska     566432
## Arizona    113417
## Arkansas    51945
## California 156361
## Colorado   103766
```

# Matrices and data frames

From the R `datasets` package:

- State statistics for the US from the 1970s

```
str(state.x77)
```

```
##  num [1:50, 1:8] 3615 365 2212 2110 21198 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##   ..$ : chr [1:8] "Population" "Income" "Illiteracy" "Life Exp" ...
```

# Matrices and data frames

US state statistics from 1970s (R datasets)

```
# Subset the dataset to look at a few variables
state.x77 <- state.x77[1 : 10, c(1, 2, 3)]
state.x77
```

```
##             Population Income Illiteracy
## Alabama           3615   3624        2.1
## Alaska             365   6315        1.5
## Arizona           2212   4530        1.8
## Arkansas          2110   3378        1.9
## California       21198   5114        1.1
## Colorado          2541   4884        0.7
## Connecticut       3100   5348        1.1
## Delaware           579   4809        0.9
## Florida           8277   4815        1.3
## Georgia           4931   4091        2.0
```

```
class(state.x77)
```

```
## [1] "matrix"
```

# Matrices and data frames

We can generally subset matrices like data frames:

```
state.x77[c(1, 3), 1 : 2]
```

```
##          Population Income
## Alabama        3615   3624
## Arizona        2212   4530
```

Except, we have to use brackets when referring to a column by its name

```
state.x77[ 1: 2, "Income"]
```

```
## Alabama  Alaska
##    3624    6315
```

```
state.x77$Income
```

```
## [1] "Error in state.x77$Income : $ operator is invalid for atomic vectors\n"
```

# Naming objects

- Women in the US Senate 2009-2015

```
nwomen <- c( 17, 17, 20)
nwomen
```

```
## [1] 17 17 20
```

```
names(nwomen)
```

```
## NULL
```

```
newnames <- c("111th Congress", "112th Congress", "113th Congress")
names(nwomen) <- newnames
nwomen
```

```
## 111th Congress 112th Congress 113th Congress
##             17             17             20
```

# Naming objects

```
state.x77
```

```
##             Population Income Illiteracy
## Alabama           3615   3624        2.1
## Alaska             365   6315        1.5
## Arizona           2212   4530        1.8
## Arkansas          2110   3378        1.9
## California       21198   5114        1.1
## Colorado          2541   4884        0.7
## Connecticut       3100   5348        1.1
## Delaware           579   4809        0.9
## Florida           8277   4815        1.3
## Georgia           4931   4091        2.0
```

# Naming objects

```
colnames(state.x77)
```

```
## [1] "Population" "Income"     "Illiteracy"
```

```
rownames(state.x77)
```

```
##  [1] "Alabama"     "Alaska"      "Arizona"     "Arkansas"    "California"
##  [6] "Colorado"    "Connecticut" "Delaware"    "Florida"     "Georgia"
```

# Naming objects

If we select one column from a matrix, it behaves like a vector

```
state.x77[1 : 4, "Illiteracy"]
```

```
##  Alabama   Alaska  Arizona Arkansas
##      2.1      1.5      1.8      1.9
```

```
names(state.x77[1 : 4, "Illiteracy"])
```

```
## [1] "Alabama"  "Alaska"   "Arizona"  "Arkansas"
```

# Naming objects

Suppose we have created a matrix ourselves:

```
ourmatrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   13   14    1    7
## [2,]    4   20   14   17
## [3,]   15    9    7   14
## [4,]    7   19    4    4
## [5,]    4   15   14   14
```

```
rownames(ourmatrix)
```

```
## NULL
```

```
colnames(ourmatrix)
```

```
## NULL
```

# Naming objects

We can then assign names:

```
rownames(ourmatrix) <- c("Day 1", "Day 2", "Day 3", "Day 4", "Day 5")
colnames(ourmatrix) <- c("Variable 1", "Variable 2", "Variable 3",
  "Variable 4")
ourmatrix
```

```
##       Variable 1 Variable 2 Variable 3 Variable 4
## Day 1         13         14          1          7
## Day 2          4         20         14         17
## Day 3         15          9          7         14
## Day 4          7         19          4          4
## Day 5          4         15         14         14
```

# Naming objects

Use paste and sequence as a shortcut:

```
1 : 5
```

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

```
seq(1, 5)
```

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

```
rownames1 <- paste("Day", seq(1, 5))
rownames1
```

```
## [1] "Day 1" "Day 2" "Day 3" "Day 4" "Day 5"
```

```
rownames(ourmatrix) <- rownames1
```

# Naming objects

```r
colnames(ourmatrix) <- paste("Variable", seq(1, 4))
ourmatrix
```

```
##       Variable 1 Variable 2 Variable 3 Variable 4
## Day 1         13         14          1          7
## Day 2          4         20         14         17
## Day 3         15          9          7         14
## Day 4          7         19          4          4
## Day 5          4         15         14         14
```

# Naming objects

**In general, best to have no spaces in variable names**

Naming is critical because

- R will not force you to name your objects
- You will forget which columns correspond to which variables
- You will be working with other people, who may not be able to infer information about the data

> *Your closest collaborator is you six months ago but you don't reply to email. – Erin Jonaitis (via Andrew Gelman)*

# Summary statistics

Last class we learned how to apply some functions in R including

- mean
- median
- standard deviation
- summary

Now, we learn how to apply these to matrices and data frames

# Summary statistics

## Average Sepal.Length

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
head(iris[, 1])
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

```
dim(iris)
```

```
## [1] 150   5
```

```
length(iris[, 1])
```

```
## [1] 150
```

# Summary statistics

Average Sepal.Length

```r
mean(iris[, 1])
```

```
## [1] 5.843333
```

```r
mean(iris[, "Sepal.Length"])
```

```
## [1] 5.843333
```

```r
mean(iris$Sepal.Length)
```

```
## [1] 5.843333
```

# Summary statistics

Average illiteracy rate

```
head(state.x77)
```

```
##            Population Income Illiteracy
## Alabama          3615   3624        2.1
## Alaska            365   6315        1.5
## Arizona          2212   4530        1.8
## Arkansas         2110   3378        1.9
## California      21198   5114        1.1
## Colorado         2541   4884        0.7
```

```
mean(state.x77[, 3])
```

```
## [1] 1.44
```

```
mean(state.x77[, "Illiteracy"])
```

```
## [1] 1.44
```

Cannot use “$” to subset matrices

# Dates

```
load("googleflu.RData")
dates <- flu$Date
head(dates)
```

```
## [1] "2003-09-28" "2003-10-05" "2003-10-12" "2003-10-19" "2003-10-26"
## [6] "2003-11-02"
```

```
class(dates)
```

```
## [1] "Date"
```

```
dates[2] - dates[1]
```

```
## Time difference of 7 days
```

# Dates

What if R doesn't know we have a date?

```
flu <- read.csv("googleflu.csv", stringsAsFactors = F)
dates <- flu$Date
head(dates)
```

```
## [1] "2003-09-28" "2003-10-05" "2003-10-12" "2003-10-19" "2003-10-26"
## [6] "2003-11-02"
```

```
class(dates)
```

```
## [1] "character"
```

```
dates[2] - dates[1]
```

```
## [1] "Error in dates[2] - dates[1] : non-numeric argument to binary operator\
```

# Dates

Use the function as.Date

```r
dates <- as.Date(dates, format = "%Y-%m-%d")
dates[2] - dates[1]
```

```
## Time difference of 7 days
```

Revise the flu dataset

```r
class(flu$Date)
```

```
## [1] "character"
```

```r
flu$Date <- dates
class(flu$Date)
```

```
## [1] "Date"
```

# Lists

- Collections of unlike R objects

```
grades1 <- c(90, 70, 50)
names(grades1) <- paste("Student", seq(1, 3))

instructor <- "Dr. Jenna Krall, PhD"
numberstudents <- 42
```

# Lists

```
introRepi <- list(instructor, numberstudents, grades1)
introRepi
```

```
## [[1]]
## [1] "Dr. Jenna Krall, PhD"
##
## [[2]]
## [1] 42
##
## [[3]]
## Student 1 Student 2 Student 3
##        90        70        50
```

```
length(introRepi)
```

```
## [1] 3
```

# Lists

```
names(introRepi)
```

```
## NULL
```

```
names(introRepi) <- c("instructor", "numberstudents", "grades")
introRepi
```

```
## $instructor
## [1] "Dr. Jenna Krall, PhD"
##
## $numberstudents
## [1] 42
##
## $grades
## Student 1 Student 2 Student 3
##        90        70        50
```

# Lists

```
statedata <- list("State data for 1970 from R datasets package", state.x77)
names(statedata) <- c("Info", "Data")
statedata
```

```
## $Info
## [1] "State data for 1970 from R datasets package"
##
## $Data
##             Population Income Illiteracy
## Alabama           3615   3624        2.1
## Alaska             365   6315        1.5
## Arizona           2212   4530        1.8
## Arkansas          2110   3378        1.9
## California       21198   5114        1.1
## Colorado          2541   4884        0.7
## Connecticut       3100   5348        1.1
## Delaware           579   4809        0.9
## Florida           8277   4815        1.3
## Georgia           4931   4091        2.0
```

# Lists

- Can subset lists by name or order

```
statedata[[1]]
```

```
## [1] "State data for 1970 from R datasets package"
```

```
statedata$Info
```

```
## [1] "State data for 1970 from R datasets package"
```

# Lists

```r
statedata[1]
```

```
## $Info
## [1] "State data for 1970 from R datasets package"
```

```r
statedata[[1]]
```

```
## [1] "State data for 1970 from R datasets package"
```

Different classes when subset using double brackets:

```r
class(statedata[1])
```

```
## [1] "list"
```

```r
class(statedata[[1]])
```

```
## [1] "character"
```

# Sorting data

We can use the sort command to sort a vector:

```r
s_date <- sort(flu$Date)
head(s_date)
```

```
## [1] "2003-09-28" "2003-10-05" "2003-10-12" "2003-10-19" "2003-10-26"
## [6] "2003-11-02"
```

```r
s_date <- sort(flu$Date, decreasing = TRUE)
head(s_date)
```

```
## [1] "2014-11-23" "2014-11-16" "2014-11-09" "2014-11-02" "2014-10-26"
## [6] "2014-10-19"
```

```r
summary(flu$Date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2003-09-28" "2006-07-12" "2009-04-26" "2009-04-26" "2012-02-08"
##         Max.
## "2014-11-23"
```

# Sorting data

We can also reorder our data based on one column using `order`

```
head(flu)
```

```
##          Date United.States Georgia Atlanta HHSRegion4
## 1 2003-09-28           902     514     519        631
## 2 2003-10-05           952     532     484        652
## 3 2003-10-12          1092     557     497        735
## 4 2003-10-19          1209     608     563        822
## 5 2003-10-26          1249     745     845        797
## 6 2003-11-02          1374     767     771        850
```

```
ord_date <- order(flu$Date)
head(ord_date)
```

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

# Sorting data

```r
ord_date <- order(flu$Date, decreasing = TRUE)
head(ord_date)
```

```
## [1] 583 582 581 580 579 578
```

```r
flu_ord_date <- flu[ord_date, ]
head(flu_ord_date)
```

```
##           Date United.States Georgia Atlanta HHSRegion4
## 583 2014-11-23          1673    3046    3152       1858
## 582 2014-11-16          1549    2569    2884       1522
## 581 2014-11-09          1379    1679    1427       1380
## 580 2014-11-02          1374    1884    2227       1384
## 579 2014-10-26          1224    1440    1536       1244
## 578 2014-10-19          1349    1437    1582        949
```

# Creating new variables

“Initializing” a new vector or matrix

```
vector1 <- vector(length = 5)
mat1 <- matrix(nrow = 2, ncol = 2)
```

# Creating new variables

“Initializing” a new vector or matrix

```
vector1 <- vector(length = 5)
vector1
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```
mat1 <- matrix(nrow = 2, ncol = 2)
mat1
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
```

# Creating new variables

## Filling in values

```
mat1
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
```

```
mat1[1, 1] <- 1
mat1[1, 2] <- 2
mat1[2, 1] <- 3
mat1[2, 2] <- 4
mat1
```

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

# Creating new variables

## Creating a dataframe

```
class(mat1)
```

```
## [1] "matrix"
```

```
mat1 <- data.frame(mat1)
class(mat1)
```

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

```
mat1
```

```
##   X1 X2
## 1  1  2
## 2  3  4
```

# Creating new variables

We want to create a new variable that is "High" when the flu activity in Georgia is over 3500 and "Low" when flu activity in Georgia is less than or equal to 3500

```
flu_high <- vector(length = length(flu$Georgia))
head(flu_high)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
wh_g3500 <- which(flu$Georgia > 3500)
head(wh_g3500)
```

```
## [1] 11 12 13 14 72 73
```

```
wh_lt3500 <- which(flu$Georgia <= 3500)
head(wh_lt3500)
```

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

# Creating new variables

```
flu_high[wh_g3500] <- "High"
flu_high[wh_lt3500] <- "Low"
head(flu_high)
```

```
## [1] "Low" "Low" "Low" "Low" "Low" "Low"
```

```
class(flu_high)
```

```
## [1] "character"
```

```
flu_high <- factor(flu_high)
head(flu_high)
```

```
## [1] Low Low Low Low Low Low
## Levels: High Low
```

# Creating new variables

We may want to change the labels of a factor variable

```
levels(flu_high)
```

```
## [1] "High" "Low"
```

```
flu_high_2 <- factor(flu_high, levels = c("High", "Low"),
  labels = c(">3500", "<=3500"))
head(flu_high_2)
```

```
## [1] <=3500 <=3500 <=3500 <=3500 <=3500 <=3500
## Levels: >3500 <=3500
```

# Creating new variables

Multiple conditions

```
head(flu$Date)
```

```
## [1] "2003-09-28" "2003-10-05" "2003-10-12" "2003-10-19" "2003-10-26"
## [6] "2003-11-02"
```

```
flu$Date[flu$Georgia > 6000]
```

```
##  [1] "2003-12-14" "2003-12-21" "2008-02-10" "2008-02-17" "2012-12-02"
##  [6] "2012-12-09" "2012-12-16" "2012-12-23" "2012-12-30" "2013-01-06"
## [11] "2013-01-13" "2013-01-20" "2013-01-27" "2013-12-22"
```

```
flu$Date[flu$Atlanta > 6000]
```

```
##  [1] "2003-12-07" "2003-12-14" "2003-12-21" "2008-02-17" "2012-11-25"
##  [6] "2012-12-02" "2012-12-09" "2012-12-16" "2012-12-23" "2012-12-30"
## [11] "2013-01-06" "2013-01-13" "2013-01-20" "2013-01-27"
```

# Creating new variables

## Multiple conditions

```
flu$Date[flu$Georgia > 6000 & flu$Atlanta > 6000]
```

```
##  [1] "2003-12-14" "2003-12-21" "2008-02-17" "2012-12-02" "2012-12-09"
##  [6] "2012-12-16" "2012-12-23" "2012-12-30" "2013-01-06" "2013-01-13"
## [11] "2013-01-20" "2013-01-27"
```

```
flu$Date[flu$Georgia > 6000 | flu$Atlanta > 6000]
```

```
##  [1] "2003-12-07" "2003-12-14" "2003-12-21" "2008-02-10" "2008-02-17"
##  [6] "2012-11-25" "2012-12-02" "2012-12-09" "2012-12-16" "2012-12-23"
## [11] "2012-12-30" "2013-01-06" "2013-01-13" "2013-01-20" "2013-01-27"
## [16] "2013-12-22"
```

```
flu$Date[(flu$Georgia > 6000 & flu$Atlanta <= 6000) |
  (flu$Georgia <= 6000 & flu$Atlanta > 6000) ]
```

```
## [1] "2003-12-07" "2008-02-10" "2012-11-25" "2013-12-22"
```

# Saving data

- Objects in your workspace (console) are not saved
- Don't save your workspace (the prompt when closing R)

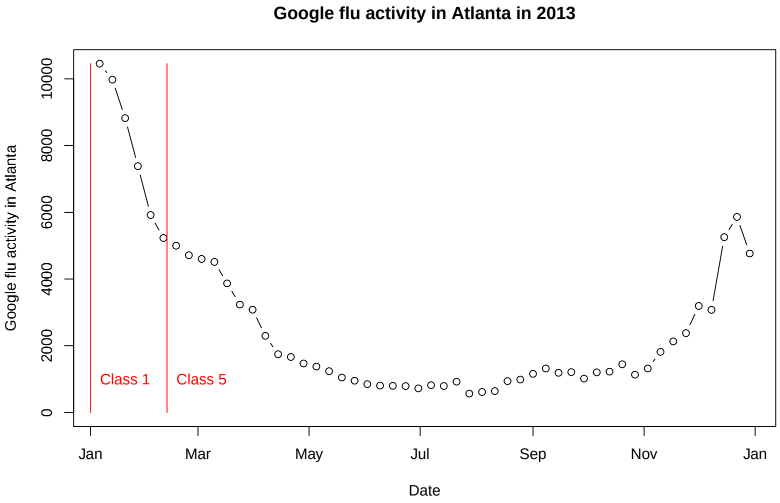
How to save?

- Save your code (in the editor using a .R file)
- Save only relevant output

```
save(flu, file = "revised_flu.RData")
```

- *Will not overwite unless your file name is same as the old file name*
- Remember to set your working directory
- Other functions to save output include `write.csv`, `write.table`

# Next week: plotting data


Google flu activity in Atlanta in 2013
Google flu activity in Atlanta
Jan
Mar
May
Jul
Sep
Nov
Jan
Date
Class 1
Class 5