UCLA Department of Statistics
Statistical Consulting Center

Migrating to R for SAS/SPSS/Stata Users

Vivian Lew
vlew@stat.ucla.edu

October 4, 2009

Outline

What are SAS (1966), Stata (1985), and SPSS (1968)?

- Statistical packages allow you to read raw data and transform it into a proprietary file (sas7bdat, dta, sav files)
- Provide statistical and graphical procedures to help you analyze data.
- Provide a management system to store output from statistical procedures for processing in other procedures, or to let you customize printed output.
- Provide a macro (internal programming) language which use commands
- Provide a matrix language to add new functions and procedures (SAS/IML and SPSS Matrix).

What is R (1997)?

R is a statistical "environment" – a fully planned and coherent system, rather than collection of specific and inflexible tools which have been added in increments

- It runs on a variety of platforms including Windows, Unix and MacOS.
- It provides an unparalleled platform for programming new statistical methods in a straightforward manner.
- It contains advanced statistical routines not yet available in other packages.
- It has state-of-the-art graphics capabilities.
- It's efficient with its resources and it is free!

Assumptions About Your Knowledge

- Assume that you have used a commercial stat package like SAS/STATA/SPSS
- Assume you have had the opportunity to read data into a commercial stat package
- Assume you have written data from commercial stat package and generated output
- Assume you have managed data (e.g., merging datasets) with a stat package
- Assume you have attempted to generate descriptive statistics or estimated models too

Some Basic Differences

- SAS/STATA & SPSS generate considerable output, R gives minimal output
- SAS & SPSS commands are not case sensitive, R is case sensitive so foo, FOO, fOO etc. are represent different things in R
- R allows the continuous modification of data, SAS will not (SPSS & Stata will)
- R Basic functions are available by default. Other functions are made available as needed. SAS and SPSS include all functions at the time of installation.

Basic R Components

Data Object (a "container"): Primary data object is the vector

- an ordered collection of elements
- R functions perform operations on vector contents
- Vectors can be specialized (e.g., character, logical)
- Objects can be collections of vectors

R functions : operate on an Object's contents. Functions are processing instructions

- Functions work differently on different objects
- Functions are built-in or user written

R command hierarchy

R script: a collection of R commands similar to a Stata Do file or a SAS program file. Typically calls one or more existing R functions in a script.

R function: a self-contained programming object similar to commands in Stata/SAS/SPSS

R program: A collection of functions for solutions you intend to reuse and/or share with others. Similar to SAS and Stata Macro programming.

R package: A expert level R program. Tested and documented to CRAN standards. Downloadable as a self-installing and configuring module. Contains information on dependencies (on other R modules). SAS does not allow this, Stata allows the use of Ado files written by users.

Let's examine a little R script

Key Characteristics of Data Objects in R

Every Data Object has Attributes and they define behavior

Class: Many R objects have a class attribute, a character vector giving the names of the classes from which the object inherits. If the object does not have a class attribute, it has an implicit class, e.g., "matrix" or "array"

Mode: a character string giving the desired mode or storage mode (type) of the object. Commonly: numeric, character, and logical (e.g., TRUE/FALSE), but also list

Length: Number of elements in the object

Having problems with a data object? It's probably being used incorrectly

Common Data Objects

Vector: a data structure consisting of elements that are accessed by indexing

List: a generic vector (can be mixed mode)

Array: A collection of items that are randomly accessible by integers, the index. Array objects cannot be mixed mode.

Matrix: A two-dimensional array. By convention, the first index is the row, and the second index is the column. ALL columns in a matrix must have the same mode(e.g., numeric) and the same length.

Dataframe: A generalized matrix, different columns can have different modes (e.g., numeric, character, factor). This is similar to SAS, Stata, and SPSS datasets.

Datasets in SAS, Stata, & SPSS

- Many data analyses involve the dataset – a collection of values that function as a single unit. Example, we survey the students enrolled in a basic statistics course, for each student we have gender, age, year in school, major, etc. For each student we may have values for each of these variables.
- If we were to store the data in a matrix in R (rows = observations, columns=variables) a matrix would want all of the data to have the same mode, so it would force the numeric variables to be stored as character (this creates problems)
- The solution is to store the data as a data frame.

Convert to CSV then to R

- The quickest way between commercial Stat Package datasets and R is to use the Stat Package software to convert an existing dataset to a "comma separated values" file (.csv) and transform it into an R object by using function read.csv

```
mydata1<-read.csv("http://www.stat.ucla.edu/~vlew/stat291/labdata1.csv")
```

- read.csv reads a .csv file and creates an R data frame from it, with observations corresponding to rows and variables to columns in the file. It just fills in arguments of the read.table function.

Check Our Results

Quick Descriptive Stats	summary(mydata1)
What is it	is(mydata1)
View its structure	str(mydata1)
simpler structural view	dim(mydata1)
what is a factor?	levels(mydata1$GENDER)
fivenum for numeric	fivenum(mydata1$HEIGHT)
view the first few	head(mydata1, n=10)
or the last few	tail(mydata1, n=5)

Beyond CSV

Suppose instead you do not have access to software other than R but need to convert the following: a Stata dataset and a SAS dataset. R will need some additional packages to make this happen. It seems like a hassle to go this route, but consider the following:

A minimum SAS install requires around 2gb of disk space now, a minimum R install might only require 30mb or so and you add on packages as needed. These two packages are particularly handy if you are doing a lot of work between R and other software.

```
1 library(foreign) \#Load the needed packages.
2 library(Hmisc)   \#Load the needed packages.
3 mydata2<-stata.get("http://www.stat.ucla.edu
    /~vlew/stat291/labdata1.dta")
4 mydata3<-sasxport.get("http://www.stat.ucla.
    edu/~vlew/stat291/labdata1.xpt")
```

Test Your Skills

We now have mydata1, mydata2, mydata3

1. Quickly Summarize the datasets
2. Check the structure of the dataset
3. Try creating a table or a summary for a specific variable

Nearly Anything Can Become a Data Frame

Suppose you see data on-line that you want to convert to R, fixed format data makes things easy

```
site<-"http://www.data.scec.org/ftp/catalogs/SCEC_DC/1993.catalog"
eq<-read.table(site,skip=12)
summary(eq)
```

We can make the data frame "prettier" by adding labels. Note that vl is a vector of characters that is used as a label.

```
vl<-c("DATE","TIME","ET","MAG","M","LAT","LON","DEPTH","Q","EVID",
"NPH","NGRM")
eq2<-read.table(site,col.names=vl,skip=12)
summary(eq2)
```

Extracting Data from Data Frames

In the data frame, the columns represent "variables" and the rows are "observations". One way to analyze the data is to work with the "variables" (columns).

1. Use it's name, e.g., summary(eq2$MAG) eq2 is the data frame, and MAG is the header of one of its columns. Dollar sign ($) is an operator that accesses that column.
2. Use subscripts, e.g., summary(eq2[4]) will work for data frames or summary(eq2[,4]) works generally to extract the 4th column MAG
3. Use attach() it essentially splits the data frame into variables and allows use the names directly (no need for $), for example:

```
attach(eq2)
summary(MAG)
```

Extracting Rows from Data Frames

In the data frame, the rows are "observations". Another way to analyze the data is to work with a combination of specific observations and variables.

1. Use subscripts, e.g., summary(eq2[1:10,]), notice the comma is telling us we want all the columns, but only rows 1:10
2. Use subscripts, e.g., summary(eq2[1:10,4]), now it's observations 1 through 10 and variable 4 (which is MAG)
3. Even with attach, understanding subscripts is VERY helpful, e.g., summary(MAG[1:10])

Subsetting (Conditioning or Keeping)

A common task is to create a smaller data set (data frame) based upon the values of a variable. For example, we might only want large earthquakes, function which() is quick:

```
1  eq2[MAG > 4,]
2  BIGONES <- eq2[MAG > 4,]
3  bigones <- eq2[MAG > 4,1:4]
4  str(BIGONES)
5  str(bigones)
```

Subsetting (Conditioning or Keeping)

For more complicated data extraction from data frames, use the subset() function because it is designed to handle more complex conditioning. Note that subset will always want to create a new data frame.

```
1  small1 <- subset(eq2, MAG < 1, select=c(MAG,
       LAT, LON, DATE))
2  str(small1)
3  small2 <- subset(eq2, MAG < 1, select=-c(MAG,
       LAT, LON, DATE))
4  str(small2)
5  small3 <- subset(eq2, MAG < 3 \& DEPTH < 5.9,
       select=c(MAG, LAT, LON, DEPTH))
6  str(small3)
7  summary(small3)
```

Sampling from a Data Frame

Drawing random samples from a data frame is simple but can be frustrating, this is a good example of why it matters that we're working with a data frame and the fact that function sample was written with vectors in mind:

```
notwhatiwanted <- sample(eq2,100)
Error in '[.data.frame'(x, .Internal(sample(
    length(x), size, replace, :
  cannot take a sample larger than the
     population when 'replace = FALSE'
```

But eq2 has over 20,000 observations so how can this be? Try it with a sample of size 10 instead and use the replace option:

```
sample(eq2,10, replace=TRUE)
```

Sampling from a Data Frame continued

So we learn that applying a function for a vector and incorrectly applying it to eq2 we get a sample of the variables. The solution:

```
rs <- eq2[sample(1:nrow(eq2), 100),]
rs2 <- eq2[sample(1:nrow(eq2), 100, replace=
    TRUE),]
```

We are now using a vector the same size as the number of rows in eq2 and sample size is 100. Note the comma outside of the parenthesis, this allows us to keep all of the variables (columns) in the data frame eq2.

Saving Your New Dataset

1. Issuing quit() and responding "yes" to the question of saving your Workspace image is a way to save all of your objects so that you can start again the next time.
2. But if you don't want everything, you can selectively save using the save function and load it later

```
save(bigones, jan, rs2, file='sas2r.rda')
load('sas2r.rda')
```

3. And you could export your data frames, here, we go back to a .csv file

```
write.table(eq2,file='eq2.csv',row.names=
    TRUE,sep=',')
```

Completely New Variable

Suppose you wish to number each observation, a simple one is to just use its row number. We can then combine the new column vector with the existing data frame.

```
ID <-c(1:nrow(eq2))
str(ID)
neweq2 <- cbind(ID, eq2)    # store it in a new
    object
tail(neweq2,n=10)           # check your work
```

Converting a Continuous Variable

This is an easy but handy conversion, a continuous variable will be converted to either TRUE or FALSE (or 1 or 0) based on simple decision. Here, earthquakes with magnitudes above 3 are consider "big":

```
bigones <- MAG >= 3.0
table ( bigones )
str( bigones )
names ( neweq2 )
neweq2 <- cbind (neweq2 , bigones )
names ( neweq2 )
summary ( neweq2 )
```

Converting A Factor

Here, we have Factor Q which has 4 levels "A", "B", "C", "D" which represent the quality of the measurement. Suppose we don't think C is very different from D and wish to collapse it to minimize the number of categories. Also, we don't want to deal with letters, we would rather have numbers:

```
newq = ifelse (Q %in% c("A"), 4, ifelse (Q %in
    % c("B") ,3 ,2))
is( newq )
table (newq ,Q)
mean ( newq )
```

Now it could be interpreted more like a GPA.

Converting a Continuous Variable into a Factor

The function cut is used to describe how ranges of a continuous variable are to be converted to factor values. Here we take the magnitude of an earthquake and divide it into 5 levels:

```
MAGCUT1 = cut (MAG , breaks =5)
table ( MAGCUT1 )
MAGCUT2 = cut (MAG , pretty (MAG ,5))
table ( MAGCUT2 )
MAGCUT3 = cut (MAG , quantile (MAG ,(0:5) /5,
    na.rm= TRUE ))
table ( MAGCUT3 )
```

Q. If you want to know what pretty and quantile do, how can you find out?

Factors in Detail

- Factors in R are variables which take on a limited number of values. They are also known as categorical variables. Categorical variables are different from continuous variables and need to be treated differently particularly when you employ models.
- Factors represent a very efficient way to store character values, because each unique character value is stored only once, and the data itself is stored as a vector of integers.
- The function cut() is used to convert a numeric variable into a factor. The breaks= argument to cut is used to describe how ranges of numbers will be converted to factor values. If a number is provided through the breaks= argument, the resulting factor will be created by dividing the range of the variable into that number of equal length intervals if a vector of values is provided, the values in the vector are used to determine the breakpoint.

NOTE: if a vector of values is provided, the levels of the resultant factor will be one less than the number of values in the vector.

Factors in Detail cont'd

Another way

```
1  MAG2 <- 0 # initialize MAG2
2  MAG2[MAG == 0] <- 0
3  MAG2[MAG > 0 & MAG < 1.0] <- 1
4  MAG2[MAG >= 1.0 & MAG < 2.0] <- 2
5  MAG2[MAG >= 2.0 & MAG < 3.0] <- 3
6  MAG2[MAG >= 3.0] <- 4
7  table(MAG2) #check
8  is(MAG2)     # check, not a factor
9  MAG2F <- factor(MAG2, levels=0:4, labels=c("0"
       , "0-1", "1-2", "2-3", "GT 3"))
10 table(MAG2F) # check
11 detach()      # add it to eq2
12 attach(eq2)
13 is.factor(MAG2F)  # check
```

Reshaping

Reshape converts what is called "long" (typically multiple measurements on the same subject with one measurement per line) data to "wide" (typically one subject per line with multiple measurements as variables) or "wide" to "long" to suit the needs of the particular method used to analyze the data.

```
1 library(reshape)
2 gdp <-  read.csv("http://www.stat.ucla.edu
     /~vlew/stat291/gdpgrowth.csv")
3 head(gdp)
4 gdplong <- reshape(gdp,varying=list(names(gdp)
     [-1: -3]), times=seq(1960, 2006, by=1),
     direction='long')
5 head(gdplong, n=232)
```

Appending or Stacking

Appending (or Stacking) is the same as adding rows to a matrix. An example using 3 data frames. R demands that the columns have the same mode. There may also be problems if the datasets have different numbers of columns.

```
ww1 <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/ww1.csv")
ww2 <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/ww2.csv")
ww3 <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/ww3.csv")
str(ww1)
str(ww2)
str(ww3)
ww <- rbind(ww1, ww2, ww3)
dim(ww)
str(ww)
```

Merging

Merging is concatenating columns with the addition of matching on an identifier variable.

```
team <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/team.csv")
player1 <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/player1.csv")
player2 <- read.csv("http://www.stat.ucla.edu/~vlew/stat291/player2.csv")
head(player1)
head(player2)
easy1 <- merge(player1, player2, by=c("Rank", "Player", "Team"), all=T)
head(easy1)
head(team)
easy2 <- merge(team, easy1, by="Team", all=T)
head(easy2)
```

Aggregating

R provides many methods for aggregating data. The simplest version is to use the table function to reduce any given dataset down to an array of as many dimensions as tabulated. In this example, the number of bedrooms in a home are crossed with the type of home and stored in a new data object.

```
bedbytype <- table(BEDS,HOME.TYPE)
bedbytype
is(bedbytype)
bedbytype2 <- as.data.frame(bedbytype)
bedbytype2
is(bedbytype2)
```

More Aggregating

A slightly more complex aggregation involving data frames utilizes two additional packages and the function aggregate. Here the mean number of bedrooms is computed for combinations of the type of home and the type of sales transaction.

```
library(stats)
library(gdata)
meanbeds <- aggregate.table(BEDS,HOME.TYPE, SALE.TYPE, mean )
meanbeds
is(meanbeds)
```

When attempting to aggregate data, consider first the desired end result and then understand the structure (variables, unit of analysis) of the dataset.

NA

- NA is a logical constant of length 1 which contains a missing value indicator.
- Numeric missing take NA (character missing are blanks)
- Data may have missing values or existing values can be set to missing
- Some functions will not execute if NAs are present unless an option to exclude NAs is included.

```
x <- c(8, 6, 7, 5, 3, 0, 9, -1, -2, -3) # the
    c() operator, for concatenation
summary(x)
x[8:10] <- NA
summary(x)
x <- c(8, 6, 7, 5, 3, 0, 9, -1, -2, -3)
x[x < 0] <- NA
summary(x)
eq2[which(ET =="qb"), 4] <- NA
summary(eq2)
```

Dates

- Dates are represented as the number of days since 1970-01-01 (Unix Epoch), with negative values for dates prior to January 1, 1970.
- They are always printed following the rules of the current Gregorian calendar.
- A variable might be named "date" and may look like a date, but it may not be recognized as a date by R.
- The solution is to convert date variables to an R date. Once that is done, mathematical operations can be performed.

```
attach(eq2)
summary(DATE)
is(DATE)
newdate <-as.Date(DATE, format='%Y/%m/%d')
is(newdate)
summary(newdate)
```

More about Dates

Additionally, converting non-R dates into R recognized dates allows a user access to some useful functions

```
x <- paste(DATE, TIME) # concatenates character strings
head(x)
is(x)
y <- strptime(x, "%Y/%m/%d %H:%M:%S")
head(y)
is(y)
DOW <- weekdays(y)
MONTH <- months(y)
table(DOW)
```

Extra Data For You To Try

```
UCLA Statistics
http://www.stat.ucla.edu/data/
Southern California Earthquake Center
http://scedc.caltech.edu/ftp/catalogs/SCEC_DC/
Chance Datasets
http://www.dartmouth.edu/~chance/teaching_aids/data.html
Federal Reserve Board of Governors
http://www.federalreserve.gov/econresdata/researchdata.htm
American Deaths in Iraq
http://icasualties.org/Iraq/BY_DOD.aspx
```

Basic Stats & Stuff

- We offer basic courses in R which would go over things like other basic statistics, correlations, simple graphs which I have chosen to treat lightly today.
- Someone has a clever R page where they take Stata Corporation's demonstration of their software and completely rewrite the demonstration in R. I have a copy of their R script saved to the 291 directory, but you can find their original work at:

http://wiki.r-project.org/rwiki/doku.php?id=guides:demos:stata_demo_with_r

Upcoming Mini-Courses

Next Week:

- Integrating R and LaTeX: Sweave (October 12, Monday)
- Introductory Statistics with R (October 14, Wednesday)

For a schedule of all mini-courses offered please visit *http://scc.stat.ucla.edu/mini-courses* .

Our Survey

PLEASE Follow this link and take our brief survey

http://scc.stat.ucla.edu/survey

It will help us improve this course, thank you.