an introduction to R for epidemiologists

manipulating data

Charles DiMaggio, PhD, MPH, PA-C

New York University Department of Surgery and Population Health
NYU-Bellevue Division of Trauma and Surgical Critical Care
550 First Avenue, New York, NY 10016

Spring 2015

- http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/
- Charles.DiMaggio@nyumc.org

Outline

R is not a DBMS

...clear presumption by the designers of R that you will be able to modify your input files using other tools... (Venebles)

Outline

spreadsheet interfaces
(not recommended)

- vectorized approach best, but if you must...
- data.entry() - automatically saves changes
  - looks like spreadsheet
  - automatically saves changes
  - better for vectors and matrices
- edit()
  - looks like original object
  - must explicitly assign object name (e.g. overwrite original name)
  - better for arrays and data frames
- fix() - like edit() but automatically overwrites and saves

rearrange parts of variables

substr()/paste()/strsplit()

- substr(x, start, end) - extract
- paste(x,sep) - stitch together
- strsplit(x, split) - split string by substring

```
date<-c("29Jan2007", "13Jul1963", "10Mar1999")
m<-substr(date,3,5)
d<-substr(date,1,2)
md<-paste(m,"/",d, sep="")
a<-strsplit(md,"/")
str(a)
```

conditional transformations
ifelse()

ifelse(test, if test = TRUE do this, else do that).

```
x <- sample(c("M", "F"), 10, replace = T)
x
y <- ifelse(x=="M", "Male", "Female")
```

Outline

merge()

```
set.seed(1972)
a<-data.frame(id=sample(1:100, 25),var1=round(rnorm(25,50,2))
b<-data.frame(id=sample(1:100, 25),var2=round(rnorm(25,10,1))
ab<-merge(a,b,by="id")
ab

match(a$id, b$id)
a$id %in% b$id
intersect(a$id, b$id)
```

about merge()

- default for two dataframes merges rows based on columns (natural join) returning only those rows which had observations for variables common to both,
    - all=TRUE returns a full outer join
    - all.x=TRUE left join (if x is named first)
    - all.y=TRUE right join(if y is named second)
- by= argument for fuller control of join (like in DBMS)
    - by more than one id variable: by=c("id1", "id2")
    - if id has different names in each dataset: by.x="ID", by.y="ident"
- as in any merging, caution multiple occurrences of values of a merging variable
- factors seem to mess with merge, best to merge on character variable

when merge doesn't work

- in general, merge() works intuitively and as expected...
- ... but, sometimes merge() behaves badly
  - e.g. all.x=TRUE left join returns multiple matches
- plyr::join() a good alternative
  - works more like sql
  - setting match to "first" takes care of multiple matches
  - the default "all", is set to emulate merge()

```
join(x, y, by = matchingVar,
     type = "left", match = "first")
```

Outline

subsetting data

indexing

1. create logical vector (index)
2. apply index

the oswego data set

```
library(epitools)
data(oswego)
ill<-oswego$ill=="Y" # create index
cases<-oswego[ill,] # apply index
```

multiple criteria: ill women who ate ice cream

```
ill.fem.ice<-oswego$ill=="Y" & oswego$sex=="F"
& oswego$vanilla.ice.cream=="Y"
cases2<-oswego[ill.fem.ice,]
```

subsetting data

subset()

- alternative to indexing (data frames only)
- data frame object name
- "subset=" creates logical vector (index)
- "select=" variables to keep

```
oswego.fcv <- subset(oswego, subset = (ill=="Y" & sex=="F"
  & vanilla.ice.cream=="Y"),
  select = c(id:onset.date, vanilla.ice.cream))
```

Outline

wide to long
stack()

- e.g. anova expect data in single column, 2nd column identifying group
- *select*= argument to choose just those variables you want stacked
- unstack() goes in other direction, needs formula to explain roles of variables

```
m <- matrix(data=round(cbind(rnorm(10, 0), rnorm(10, 2),
rnorm(10, 5))), nrow=10, ncol=3)
colnames(m)<-c("a", "b", "c")
m<-as.data.frame(m)
ms<-stack(m)
ms
```

reshape package

another approach

flexible aggregation, cross-tabulation; can apply functions

- melt() - identify grouping ("id") and analysis variables
  - default factor and integer vars as "id", others "measure"
  - override with id.var= or measure.var= (need only specify one)
- cast() - aggregate or cross-tab, apply function

```
cast(melted data, row ~ column, function)
e.g. cast(mstates, region~ses, mean)
returns mean SES for each region in a state
```

Outline

"fix" missing values

assignment

individually replace missing with NA

```
x$age[wd$age=="."] <- NA
x$sex[wd$sex=="."]<-NA x$syndrome[wd$syndrome=="Unknown"]<-NA
x$death[wd$death=="."] <- NA
```

or, replace globally

```
x[x=="." | x=="Unknown"] <- NA
```

or, correct errors

```
x$County[wd$County=="Qweens"] <- "Queens"
```

"fix" missing values

matrix and dataframe

matrix

```
m <- m2 <- matrix (c(1, -99, 3, 4, -88, 5), 2, 3)
m[m[,1]==-99, 1] <- NA # one column at a time
m[m[,3]==-88, 3] <- NA
m2[m2==-99 | m2==-88] <- NA # globally
```

data frame

```
fname <- c("Tom", "Unknown", "Jerry")
age <- c(56, 34, -999)
z1 <- z2 <- data.frame(fname, age)
z1$fname[z1$fname=="Unknown"] <- NA # one column at a time
z1$age[z1$age==-999] <- NA
z2[z2=="Unknown" | z2==-999] <- NA # globally
```

missing vs. really missing

na.rm()

- either value is truly missing
- or the result of operation on object with missing values
- can be addressed with na.rm = TRUE

```
x <- c(2, 4, NA, 5)
sum(x)
sum(x, na.rm = TRUE)
```

"na.action=" - set NA behavior in statistical models

logical vector (index) NA positions

is.na()

```
x <- c(10, NA, 33, NA, 57)
is.na(x) #generate logical vector
which(is.na(x)) #which positions are NA
x[is.na(x)] <- 999 #replacement
# assigning NA's
x <- c(1, -99, 3, -88, 5)
x[x==-99 | x==-88] <- NA
x [1] 1 NA 3 NA 5
```

NA values in data frames

na.fail()

tests for *any* NA values

```
name <- c("Tom", "Dick", "Harry", "James", "John")
gender <- c("M", "F", "M", NA, "F")
age <- c(34, NA, 22, 18, 34)
df <- data.frame(name, gender, age)
df
na.fail(df) # all observations
na.fail(df[c(1, 3, 5),]) # complete obs
```

NA values in data frames

na.omit(), na.exclude(), complete.cases()

- na.omit() / na.exclude() - remove observations contain NA
- complete.cases() - return logical vector observations do not contain NAs
  `x[complete.cases(x),]  equivalent to na.omit`
- is.na() - to remove NA observations in indexing operations (differs from above functions that remove all missing values from data frame)

```
df$age
df[df$age<25, ] # index ages < 25
df[df$age<25 & !is.na(df$age), ] # remove uninformative row
```

na.strings= read.table option

```
what characters are to be converted to NA
(default na.strings="NA")
mydat <- read.table("dataset.txt",
na.strings = c(999, 888, "."))
```

Outline

write/read R binary file

save()/load()

```
save(objects, file="~/file_name.Rdata")

x <- 1:5; y <- x^3
save(x, y, file="xy.RData")
rm(x, y)
ls()
load(file="xy.RData")
ls()
save(list=c("x", "y"), file="xy.RData") #using list
```

write to generic ascii

write.table(), write.csv(), dump(), dput()

- write.table() /write.csv() - data frame
  ```
  write.table(infert, file="infert.dat")
  write.csv(infert, file="infert.csv")
  ```
- read.table() - to read back in
- write() - matrix
- dump() - takes list of R objects, converts to ascii text file
  - use to export or source the objects to another R session

  ```
  dump(c("tab1", "array2"),"infert_tab.R")
  ```

  - open the infertTab.R file and run or source() to read back in)
- dput() - like dump, writes R object R code to the console, or (if give a name) to an ascii text file
  ```
  dput(tab1) # to console
  dput(tab1, "tab1.R") # to file
  dget("tab1.R") # read  back in
  ```

write or export to non-R statistics packages

package "foreign"

text files

- write.foreign() - write to SPSS, Stata, SAS
- write.foreign(infert, datafile="infert.dat", codefile="infert.txt", package = "SPSS")

binary files (foreign package)

- write.dbf()
- write.dbf(infert, "infert.dbf")
- write.dta (Stata)
- write.dta(infert, "infert.dta")

Outline

input from external file

source()

- commands in external file
- e.g. complex user-written function found online...
- "echo = TRUE" print commands and results to console (otherwise no printed output)

```
# save this code as "~/testSource.R"
i <- 1:5
x <- outer(i, i, "*")
show(x) # to get results on console

source("~/testSource.R", echo=TRUE)
```

send output to an external file

sink(), capture.output()

```
# save this code as "~/testSource2.R"
i <- 1:5
x <- outer(i, i, "*")
sink("~/testSource.log") # creates output file
cat("Here are the results of the outer function",
   fill=TRUE)
show(x)
sink()
```

Outline

SQL

package sqldf

- sqldf package allows sql queries on R data frames
- NB - in addition to installing the package, need to install tcltk-8.5.5-x11.dmg
  - http://socserv.mcmaster.ca/jfox/Courses/soc3h6/RInstallation.html

```
library(sqldf)
write.table(iris, "iris.csv", sep = ",", quote = FALSE,
row.names = FALSE)
iris.csv <- read.csv.sql("iris.csv",
    sql = "select * from file where Sepal_Length > 5")
```