

# An Introduction to R, part 2

Chris Hanretty

## Recoding and missing data

- Recoding

- Missing data

## Working smarter, not harder

- Mashups!

- Doing something to all somethings

- Reshaping data

## Workflow

- The big example

- Further reading

# 1. Recoding and missing data

# Recoding: preliminaries

- R was written by statisticians
- Their data is clean
- R suits their needs

# Recoding using in-built functions

- Suppose we have a Strong Agree/Agree/Disagree/Strong Disagree response
- We want to collapse it
- We can use R's internal functions

# Recoding using built-in functions

```
data$response[data$response==  
  "Strong Agree"]<-"Agree"  
data$response[data$response==  
  "Strong Disagree"]<-"Disagree"
```

# Recoding like this is fragile

- Suppose we have the same response, coded numerically.
- We want to collapse 4 and 3 to 1 (Agree)
- We want to collapse 1 and 2 to 0 (Disagree)

# Recoding like this is fragile

```
data$response[ data$response  
  == 4 | data$response==3]<-1  
data$response[ data$response  
  == 1 | data$response==2]<-0
```

What's wrong with this?

# car to the rescue

- Let's instead use the `recode` command from the `car` package
- syntax is almost SPSS-like!

```
library(car)
data$response<-recode(data$response,
  "c(3,4)=1;c(1,2)=0;else=NA")
```

# NA to you too

- NA is R-talk for 'missing data'
- (There's only one kind in R)
- `NA != "NA"`
- `NA > 0`?

# NA to you too

- This occasionally creates problems
- Let's return to our `ref` data
- Set the value in the third row of `ref$Protestant` to missing
- Suppose that, as before, we want to select cases where `ref$Protestant > 0.5`
- `NA > 0.5?`

# Subsetting with NAs create phantom cases

```
which(ref$Protestant>0.5)
ref2<-ref [ref$Protestant>0.5,]
nrow(ref2)
ref2
```

!

# Subsetting with NAs create phantom cases

- *If* you want to subset, and
- *if* you have missing data
- use `which` instead

# More on missing data

- You can test for missing data (`is.na(...)`)
- You can omit row with any missing data at all (`complete.cases`)
- Usually, missing data are dropped
- But there are some exceptions
- `mean(ref$Protestant)?`
- `mean(ref$Protestant, na.rm=TRUE)?`

# 2. Working smarter, not harder

# Merging information from $n > 1$ datasets

- People spend *far too much time* copying and pasting
- Avoid this with R!

# Example merge

- We have one data-set, the World Bank's Database on Political Institutions
- It doesn't contain GDP data
- We have another data-set, the Penn World Tables
- It does contain GDP data (and much more!)

# Getting the data

- Penn World Tables have been included in a package
- `install.packages("pwt")`
- `library(pwt)`
- `data("pwt6.3")`
- `names(pwt6.3)`

## Getting the data (2)

- The DBPI is on the web.
- `library(foreign)`
- `dbpi<-read.dta("http://siteresources.worldbank.org/INTRES/Resources/469232-1107449512766/DPI2006_rev42008.dta")`
- `names(dbpi)`

# Finding commonalities

- The PWT has a variable, `isocode`
- The DBPI has a variable, `ifs`
- Let's take these countrycodes and merge the two
- ```
union<-merge(dbpi, pwt6.3,  
by.x=c("ifs", "year"),  
by.y=c("isocode", "year"))
```
- Too much GNP? `all.y=FALSE`

# General mash-up tips

- The key is finding the right variables to match on
- For macro-work, that's generally country + year
- ISO 3166-3 letter codes are amazing.
- Check out the `countrycode` package for automatic conversion!

# Do $x$ to all $y$

- $x$ =calculate the mean;  $y$ =all numeric variables
- $x$ =rescale to have mean zero and  $SD=1$ ;  $y$ =all numeric variables
- in these cases, we use `apply` and `friends`

# Getting the mean

- `summary(ref)` gives us the mean for each variable
- How to get just the mean?
- How not to do it:

```
my_means<-NULL
for (i in 1:no_vars) {
  my_means<-c(my_means,mean(data[,i]))
}
my_means
```

# Getting the mean

- Use `apply(ref, 2, mean)`
- '2' = apply function to columns  
(remember, *Roman Catholic!*)

# Centering variables

- Here we can use the `scale` function
- `apply(ref, 2, scale)`
- `apply(ref, 2, mean)`

# Automatic centering

- Don't do this:  
`ref<-apply(ref,2,scale)`
- `ref$Canton`
- How can we do this just to numeric variables?
- Hint: how do we apply `is.numeric` to all variables?
- Hint 2: how do we use this to subset?

# Automatic centering

```
ref[,apply(ref,2,is.numeric)]<-  
  apply(ref[,apply(ref,  
                    2,is.numeric)],  
        2,scale)
```

more intelligible:

```
numeric_vars<-apply(ref,2,is.numeric)  
ref[,numeric_vars]<-  
  apply(ref[,numeric_vars],2,scale)
```

# apply and friends

- lapply: apply a function to every member of a list
- sapply: lazy apply
- tapply: apply a function according to group

# Toy tapply example

- Suppose we want to know whether parliamentary, presidential, or semi-presidential systems are richer
- `summary(union$system)`
- `union$system<-recode(union$system,  
"0='Presidential';  
1='Assembly-elected President';  
2='Parliamentary'; else=NA")`

## Toy `tapply` example (2)

- Now, we just need to use `tapply` on real GDP (`rgdpch`)
- `tapply(union$rgdpch, union$system, mean, na.rm=T)`
- Shame about endogeneity, non-independence of observations, etc.,

# Wide to long

Suppose someone gives you a spreadsheet on public debt. It looks like this:

| Country   | 1991  | 1992  | 1993  | 1994  |
|-----------|-------|-------|-------|-------|
| Australia | 11.2  | 15.7  | 21.3  | 25.7  |
| Austria   | 28.7  | 29.6  | 33.4  | 35.2  |
| Belgium   | 108.0 | 113.1 | 115.0 | 114.4 |
| Canada    | 50.5  | 59.1  | 64.2  | 67.9  |

## Wide to long (2)

You want the data to look like this:

Australia 1991 11.2

Australia 1992 15.7

Australia 1993 21.3

Australia 1994 25.7

...

# Wide to long (3)

- Simple! Use the reshape library
- `ref<-read.csv("http://www.chrishanretty.co.uk/debt.csv",  
header=T)`
- `debt.m<-melt(debt)`
- To go the other way? Use `cast`
- Much more could be said. But ?`cast`,  
?`melt`

# 3. Workflow

# Workflow

- `ls, rm`
- flush everything: `rm(list=ls())`
- more general remarks, backups

# 4. The big example

# The big example

1. Italian public debt is projected to reach 117% next year
2. Why such irresponsible deficit spending?

# Franzese's answer

- Robert Franzese argues that debts are determined by
- the partisan composition of government
- the rate at which governments which differ in partisan composition replace one another (replacement risk)
- (plus some other things)

# Replacement risk

- Intuitive idea, complicated operationalization
- rolling seven-year standard deviation of govt left-right position  $\times$  hazard rate
- Ideas?

# R to the rescue!

We can do all this in R with the data-sets we've just seen!

- Look at the file `franzese.r`
- This is an example session, you don't need to understand it all

# Data-sets

dpi World Bank database of political institutions

debt Table on public debt, taken from OECD

pwt Penn World Tables

```
dpi<-read.dta("DPI2006_rev42008.dta")
```

```
## CREATE HAZARD RATE
```

```
## Here we demonstrate how R
```

```
## handles missing data
```

```
dpi$yrsoffc[dpi$yrsoffc==999]<-NA
```

```
dpi$yrcurnt[dpi$yrcurnt==999]<-NA
```

```
dpi$durat<-dpi$yrsoffc+dpi$yrcurnt
```

```
dpi$hazard<-1/dpi$durat
```

```
## CREATE PARTISAN CENTRE OF GOVERNMENT
```

```
## Here we demonstrate how to recode
```

```
## easily in R
```

```
## 0 is no information
```

```
## 1 is right-wing
```

```
## 2 is centrist
```

```
## 3 is left wing
```

```
library(car)
```

```
dpi$gov1rlc<-recode(dpi$gov1rlc, "0=0;1=1;2=0;3=-1;else=NA")
```

```
dpi$gov2rlc<-recode(dpi$gov2rlc, "0=0;1=1;2=0;3=-1;else=NA")
```

```
dpi$gov3rlc<-recode(dpi$gov3rlc, "0=0;1=1;2=0;3=-1;else=NA")
```

```
## here's a basic calculation  
## the variable names should be self-explanatory
```

```
dpi$cog<-dpi$gov1rlc+(dpi$gov1seat/dpi$numgov)+  
  dpi$gov2rlc+(dpi$gov2seat/dpi$numgov)+  
  dpi$gov3rlc+(dpi$gov3seat/dpi$numgov)
```

```
## NOW CREATE RUNNING STANDARD DEVIATION OF CENTRE OF  
GOVERNMENT  
## following http://www.slideshare.net/hadley/02-ddply  
  
## our strategy? divide and conquer  
library(caTools)  
window_length<-7  
  
## split the data-set up according to country-codes  
pieces<-split(dpi, dpi$ifs)
```

```
## for each segment, calculate the running standard
  deviation
## the runsd function comes from caTools

results<-vector("list",length(pieces))
for (i in seq_along(pieces)) {
  piece<-pieces[[i]]
  piece$cog_sd7<-runsd(piece$cog,window_length,
    endrule="sd")
  results[[i]]<-piece
}

## combine
dpi<-do.call("rbind",results)
```

```
## NOW CREATE REPLACEMENT RISK
## replacement risk is cog_sd7 * hazard rate
dpi$rr<-dpi$hazard*dpi$cog_sd7

## LOAD INFORMATION ON DEBT
debt<-read.table("debt.csv",header=T)
library(reshape)
debt.m<-melt(debt)
names(debt.m)<-c("countryname","year","debt")
debt.m$year<-as.numeric(gsub("X","",debt.m$year))
```

```
## merge datasets
union<-merge(debt.m, dpi, all.x=T, all.y=F, by.x=c("
  countryname", "year"), by.y=c("countryname", "year"))
library(pwt)
data("pwt6.3")
union<-merge(union, pwt6.3, by.x=c("ifs", "year"), by.y=c("
  isocode", "year"), all.x=T, all.y=F)

## create presidential dummy
union$pres<-union$system==0
union$rrsq<-union$rr^2
## run regression, ignore autocorrelation
```

```
my.mod<-lm (debt ~ cog+rr+cog*rr+rrsq+rgdpch+openc+pres ,
            data=union)

## GENERATE PREDICTIONS
rr_seq<-seq (min (union$rr , na.rm=T) , max (union$rr , na.rm=T)
            , length.out=100)
my.newdata<-data.frame ( rr=rr_seq ,
                        rrsq=rr_seq^2 ,
                        cog=mean (union$cog , na.rm=T) ,
                        rgdpch=mean (union$rgdpch , na.rm=T) ,
                        openc=mean (union$openc , na.rm=T) ,
                        pres=FALSE)
pred.results<-predict (my.mod,
                       newdata=my.newdata ,
                       interval="confidence")
```

```
## plot the results
```

```
plot(rr_seq, pred.results[,1], ## fitted value  
     main="Predicted indebtedness as % of GDP",  
     xlab="Replacement risk", ylab="Indebtedness",  
     type="l", ylim=c(0,100))
```

```
## now overplot with a dashed line
```

```
## see ?par for line types
```

```
lines(rr_seq, pred.results[,2], lty=3) ## upper bound?  
lines(rr_seq, pred.results[,3], lty=3) ## lower bound?
```

# 5. Further reading

# Books

- The library's R collection has grown significantly
- Check out the listing under 'R (Computer program language)'

# Useful links

- <http://cran.r-project.org/>
- <http://www.rseek.org/>
- Quick-R for Stata/SPSS users:  
<http://www.statmethods.net/>
- Google's 'R style guide': <http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html>