Applied R for Social Scientists

J. Alexander Branham

Spring 2016

This class

This class

- Assumption of *some* previous exposure to R
    - We're not explaining assignment, packages, calling functions, etc
- Will *not* be covering statistics
- Alex will start with some common data tasks: loading, variable creation, merging, etc
- Daniel will take the second part to talk about tables and visualization

Getting the data

Download the data

- `source()` runs a file through R
- This one checks if you have the data already and tries to download it if not
- The dataset we're using is the General Social Survey spanning 1972-2014

```r
source("check-gss-and-maybe-download.R")
```

```
[1] "GSS file exists!"
```

That file

Reading data

package	function	file formats
foreign	**`read.*`**	dta, spss, etc
haven	**`read_*`**	dta, spss, etc
utils	**`read.csv`**	csv files
readr	**`read_csv`**	csv files

Read data

- We have stata data (`*.dta`)
- Two packages in R: `foreign` and `haven`
- `foreign` is older but more stable
- `haven` is newer and can read Stata 13+ files
- `convert.factors=FALSE` ensures that R doesn't convert the values to the labels that stata uses

```
GSS <- foreign::read.dta("./data/GSS7214_R4.DTA",
                         convert.factors = FALSE)
```

Data management

The data: a table

- Let’s get a sense of the data we’re working with
- Do you want more, less, or about the same spending? (education and social security)

```
table(GSS$nateduc, GSS$natsoc, exclude = NULL)
```

```
         1     2     3  <NA>
1     8540  4335   699  7939
2     2062  2288   296  5005
3      353   457   210  1317
<NA> 11841  7408  1282  5567
```

CORRELATION

- `cor` gives us (by default) the pearson's *r* between two variables
- Without setting `use`, R tries to use the whole data, some of which are missing and thus results in `NA`

```
cor(GSS$nateduc,
    GSS$natsoc,
    use="complete.obs")
```

```
[1] 0.1965368
```

Variable creation

- Let's make an indicator variable for whether a respondent is black or not
- Here's the `race` variable
- I also like to make sure that I'm not going to overwrite an existing variable

```
table(GSS$race)
```

```

    1     2     3 
48240  8312  3047 
```

```
table(GSS$black)
```

```
< table of extent 0 >
```

Variable creation

- Using `ifelse` to create a variable conditional on other var’s values

```
GSS$black <- with(GSS, ifelse(race == 2, TRUE, FALSE))
table(GSS$black)
```

```
FALSE  TRUE
51287  8312
```

Subsetting observations

- Now let’s check to see if that correlation is different for black people
- Note how ugly this looks!

```r
cor(GSS$nateduc[GSS$black == TRUE],
    GSS$natsoc[GSS$black == TRUE],
    use="complete.obs")
```

```
[1] 0.1772446
```

DPLYR

- dplyr is an R package that makes data management *much* easier
- Different functions for data munging:
    - filter(), select(), mutate()
- It introduces the pipe operator **%>%** to the language
- Functions for merging data
    - *_join: full, inner, left, right
- group_by, which lets us perform operations on groups of the data

```r
suppressPackageStartupMessages(library(dplyr))
```

Subsetting the dplyr way

- The pipe (`%>%`) “pipes” the output of the last thing into the first argument of the next thing
- `summarize` (or `summarise`) from `dplyr` returns a `data.frame`

```
with(filter(GSS, black == TRUE),
     cor(nateduc, natsoc,
         use = "complete.obs"))
```

```
[1] 0.1772446
```

```
GSS %>%
  filter(black == TRUE) %>%
  summarize(mycor =
      cor(nateduc, natsoc,
      use = "complete.obs"))
```

```
      mycor
1 0.1772446
```

Dropping observations

- 1972 doesn't have any observations we're interested in (our spending variables weren't asked), so let's drop it
- Again, we can use `filter`, but this time we assign the result back to `GSS`:

```
GSS <- GSS %>%
  filter(year != 1972)
```

Factors

- Variables with categories can be represented as factors in R
- If you want R to think they're ordered, you can use `ordered = TRUE` as an argument

```
table(GSS$sex)
GSS <- GSS %>%
  mutate(sex = factor(sex,
                      levels = c(1,2),
                      labels = c("M","F")))
```

```
    1     2
25479 32507
```

```
table(GSS$sex)
```

```
    M     F
25479 32507
```

Operations by subcategories

- `dplyr` provides `group_by`
- Lets us perform operations to grouped data

```
thecors <- GSS %>%
  group_by(sex, black) %>%
  summarize(thecor = cor(nateduc, natsoc,
                       use = "complete.obs"),
            n = n())
```

Operations by subcategories

```
print(thecors)

Source: local data frame [4 x 4]
Groups: sex [?]

     sex black    thecor     n
  (fctr) (lgl)     (dbl) (int)
1      M FALSE 0.1918454 22446
2      M  TRUE 0.1674413  3033
3      F FALSE 0.1786193 27489
4      F  TRUE 0.1820090  5018
```

Aggregation

- Maybe we're interested in preferences by year?

```
gss_yearly <- GSS %>%
  group_by(year) %>%
  summarize(educ = mean(nateduc,
                        na.rm=TRUE),
            soc = mean(natsoc,
                       na.rm=TRUE))
```

Aggregation

```
head(gss_yearly)
```

```
Source: local data frame [6 x 3]

   year     educ   soc
  (int)    (dbl) (dbl)
1  1973 1.582287   NaN
2  1974 1.562059   NaN
3  1975 1.604930   NaN
4  1976 1.579020   NaN
5  1977 1.605854   NaN
6  1978 1.576766   NaN
```

Functions

- Means are nice, but there are other ways to summarize data
- What if we want to look at the proportion of people who support more spending minus the proportion who support less?

```r
netsupport <- function(thedata){
  prop_more <- mean(thedata == 1, na.rm = TRUE)
  prop_less <- mean(thedata == 3, na.rm = TRUE)
  prop_more - prop_less
}
```

Functions

```
GSS %>%
  group_by(year) %>%
  summarize(support_educ=netsupport(nateduc),
            support_soc=netsupport(natsoc))
```

```
 Source: local data frame [29 x 3]

    year support_educ support_soc
   (int)        (dbl)       (dbl)
1   1973    0.4177127          NA
2   1974    0.4379408          NA
3   1975    0.3950704          NA
4   1976    0.4209800          NA
5   1977    0.3941457          NA
```

Merging

- The `ggplot2` package provides the `economics` data.frame that has US economic data starting in July 1967
- `?economics` gives more info

```
library(ggplot2)
head(economics)
```

```
Source: local data frame [6 x 6]

        date   pce    pop psavert uempmed unemploy
      (date) (dbl)  (int)   (dbl)   (dbl)    (int)
1 1967-07-01 507.4 198712    12.5     4.5     2944
2 1967-08-01 510.5 198911    12.5     4.7     2945
3 1967-09-01 516.3 199113    11.7     4.6     2958
4 1967-10-01 512.9 199311    12.5     4.9     3143
5 1967-11-01 518.1 199498    12.5     4.7     3066
```

Merging

- Let's make an unemployment rate by unemploy/pop

```
economics <- economics %>%
  mutate(unemp_rate = unemploy / pop)
```

Merging

- The `economics` data is monthly and our GSS data is yearly, so we need to aggregate

```
economics_yearly <- economics %>%
  mutate(year = format(date, "%Y")) %>%
  group_by(year) %>%
  summarize(unemp = mean(unemp_rate))
```

Merging

- Let's see what our data looks like now!

```
head(economics_yearly)
```

```
Source: local data frame [6 x 2]

   year      unemp
  (chr)      (dbl)
1  1967 0.01512179
2  1968 0.01394202
3  1969 0.01396464
4  1970 0.02012547
5  1971 0.02418970
6  1972 0.02323808
```

Merging

- Now we have two data.frame objects — `gss_yearly` and `economics_yearly` — that we want to join together
- `dplyr` provides a really easy way of doing this
- The jargon comes from SQL, a programming language used to store data
- What you probably call a “merge” dplyr calls a “join”
- `*_join` where `*` is either `full`, `inner`, `left`, or `right`
- We’ll use `left_join` since the economics data contains years that aren’t in the GSS

Merging

```r
gss_yearly <- left_join(gss_yearly,
                        economics_yearly,
                        by = "year")
```

```
Error: cannot join on columns 'year' x 'year': Can't join on 'yea
```

Errors

- Error: cannot join on columns ’year’ x ’year’: Can’t join on ’year’ x ’year’ because of incompatible types (character / integer)

Errors

- The error on the last slide indicates that the `year` variable in the two datasets is different
- Let's verify that:

```
class(gss_yearly$year)
```

```
[1] "integer"
```

```
class(economics_yearly$year)
```

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

Merging

- Solution: change `economics_yearly$year` to an integer

```
economics_yearly$year <- as.integer(economics_yearly$year)

gss_yearly <- left_join(gss_yearly,
                        economics_yearly,
                        by="year")
```

Merging

```r
head(gss_yearly)
```

```
Source: local data frame [6 x 4]

   year     educ   soc      unemp
  (int)    (dbl) (dbl)      (dbl)
1  1973 1.582287   NaN 0.02057710
2  1974 1.562059   NaN 0.02418823
3  1975 1.604930   NaN 0.03677624
4  1976 1.579020   NaN 0.03393594
5  1977 1.605854   NaN 0.03164388
6  1978 1.576766   NaN 0.02780555
```

Writing data

- Maybe you want to save this new data so you don't have to re-run the merging whenever you want to

package	function	result
readr	`write_csv`	csv file
utils	`write.csv`	csv file
base	`save`	Rdata file
xlsx	`write.xlsx`	excel file

- R can also write to stata/SPSS/SAS files through `foreign` or `haven`

Writing data

- Let's save a csv file
- If the `data/` subfolder doesn't exist, this will produce an error
- The script that we ran at the beginning created this if it didn't already exist

```
readr::write_csv(gss_yearly, "data/gss-yearly-data.csv")
```