Psychology 350: Special Topics
An introduction to R for psychological research
data manipulation

William Revelle
Northwestern University
Evanston, Illinois USA

https://personality-project.org/courses/350

May, 2024

Outline

Basic data manipulation

1. Data input
   - find files

     ```
     file.choose(new = FALSE)
     ```

   - input from files

     ```
     read.table(file, header = FALSE, sep = "", quote = "\"'",
                dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
                row.names, col.names, as.is = !stringsAsFactors, tryLogical = TRUE,
                na.strings = "NA", colClasses = NA, nrows = -1,
                skip = 0, check.names = TRUE, fill = !blank.lines.skip,
                strip.white = FALSE, blank.lines.skip = TRUE,
                comment.char = "#",
                allowEscapes = FALSE, flush = FALSE,
                stringsAsFactors = FALSE,
                fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

     read.csv(file, header = TRUE, sep = ",", quote = "\"",
              dec = ".", fill = TRUE, comment.char = "", ...)

     read.csv2(file, header = TRUE, sep = ";", quote = "\"",
               dec = ",", fill = TRUE, comment.char = "", ...)

     read.delim(file, header = TRUE, sep = "\t", quote = "\"",
                dec = ".", fill = TRUE, comment.char = "", ...)

     read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
                 dec = ",", fill = TRUE, comment.char = "", ...)
     ```

cbind rbind Combining two data frames/matrices

R code

```
x <- matrix(1:30, ncol=5)
y <- matrix(1:40, ncol=8)
xy <- rbind(x,y) #does not work
xy <- cbind(x,y) #nor does this
xy <- rbind(x,y[,1:5])  #this works
XY <- cbind(x[1:5,],y)
```

```
x <- matrix(1:30, ncol=5)
> y <- matrix(1:40, ncol=8)
> xy <- rbind(x,y) #does not work
Error in rbind(x, y) :
  number of columns of matrices must match (see arg 2)
> xy <- cbind(x,y) #nor does this
Error in cbind(x, y) : number of rows of matrices must match (see arg 2)
> xy <- rbind(x,y[,1:5])  #this works
> XY <- cbind(x[1:5,],y)
> xy
      [,1] [,2] [,3] [,4] [,5]
 [1,]    1    7   13   19   25
 [2,]    2    8   14   20   26
 [3,]    3    9   15   21   27
...
[10,]    4    9   14   19   24
[11,]    5   10   15   20   25
> XY
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    1    7   13   19   25    1    6   11   16    21    26    31    36
[2,]    2    8   14   20   26    2    7   12   17    22    27    32    37
[3,]    3    9   15   21   27    3    8   13   18    23    28    33    38
[4,]    4   10   16   22   28    4    9   14   19    24    29    34    39
```

Merge

Combines two files according to a common id field. "Merge two data frames by common columns or row names, or do other versions of database join operations."

R code

```
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE,
      incomparables = NULL, ...)

   x <- data.frame(k1 = c(NA,NA,3,4,5), k2 = c(1,NA,NA,4,5), data = 1
y <- data.frame(k1 = c(NA,2,NA,4,5), k2 = c(NA,NA,3,4,5), data = 1:5)
merge(x, y, by = c("k1","k2")) # NA's match
merge(x, y, by = "k1") # NA's match, so 6 rows
merge(x, y, by = "k2", incomparables = NA) # 2 rows
```

```
 merge(x, y, by = c("k1","k2")) # NA's match
  k1 k2 data.x data.y
1  4  4      4      4
2  5  5      5      5
3 NA NA      2      1
> merge(x, y, by = "k1") # NA's match, so 6 rows
  k1 k2.x data.x k2.y data.y
1  4    4      4    4      4
2  5    5      5    5      5
3 NA    1      1   NA      1
4 NA    1      1    3      3
```

Data display

R code

```
head(attitude)   #defaults to 6 rows

tail(attitude,n=8) #specify the number to show
```

```
>  head(attitude)   #defaults to 6 rows
  rating complaints privileges learning raises critical advance
1     43         51         30       39     61       92      45
2     63         64         51       54     63       73      47
3     71         70         68       69     76       86      48
4     61         63         45       47     54       84      35
5     81         78         56       66     71       83      47
6     43         55         49       44     54       49      34

> tail(attitude,n=8) #specify the number to show
   rating complaints privileges learning raises critical advance
23     53         66         52       50     63       80      37
24     40         37         42       58     50       57      49
25     63         54         42       48     66       75      33
26     66         77         66       63     88       76      72
27     78         75         58       74     80       78      49
28     48         57         44       45     51       83      38
29     85         85         71       71     77       74      55
30     82         82         39       59     64       78      39
```

psych and psychTools

1. Developed for specific applications to psychology research
2. Attempts to make R easier to use but still using core-R
3. Many of the functions take advantage of core-R functions

Basic data manipulation

1. Input from a file

   Will search for file and then read it depending upon suffix

```
read.file(file=NULL,header=TRUE,use.value.labels=FALSE,to.data.frame=TRUE,sep=",",
quote="\"", widths=NULL,f=NULL, filetype=NULL,...)
 #for .txt, .text, TXT, .csv, .sav, .xpt, XPT,  R, r, Rds, .rds, or .rda,
# .Rda, .RData, .Rdata, .dat and .DAT  files

#includes the following given appropriate suffix
read.https(filename,header=TRUE)
read.file.csv(file=NULL,header=TRUE,f=NULL,...)
```

2. Input from `clipboard`

```
read.clipboard(header = TRUE, ...)   #assumes headers and tab or space delimited
read.clipboard.csv(header=TRUE,sep=',',...)   #assumes headers and comma delimited
read.clipboard.tab(header=TRUE,sep='\t',...)   #assumes headers and tab delimited
 #read in a matrix given the lower off diagonal
read.clipboard.lower(diag=TRUE,names=FALSE,...)
read.clipboard.upper(diag=TRUE,names=FALSE,...)
#read in data using a fixed format width (see read.fwf for instructions)
read.clipboard.fwf(header=FALSE,widths=rep(1,10),...)
```

3. For output:

   creates file if not specified and then writes according to suffix.

```
write.file(x,file=NULL,row.names=FALSE,f=NULL,...)
write.file.csv(x,file=NULL,row.names=FALSE,f=NULL,...)
```

Combine two matrices/data frames

- Core-R functions
  1. `rbind` adds rows if the columns match
  2. `cbind` combines columns if number of rows is the same
  3. `merge` combines rows and columns matching on a variable
- *psych* and *psychTools*
  1. `vJoin` combines rows and columns matching on row names

Combining two matrices or data frames using vJoin

R code

```
x <- matrix(1:30, ncol=5)
y <- matrix(1:40, ncol=8)
xy <- vJoin(x,y)
headTail(xy, top=2, bottom=2)
XY <- vJoin(x,y,  cnames=FALSE)
headTail(XY)
```

```
> x <- matrix(1:30, ncol=5)
> y <- matrix(1:40, ncol=8)
> xy <- vJoin(x,y)
> headTail(xy, top=2, bottom=2)
      x1   x2   x3   x4   x5   y1   y2   y3   y4   y5   y6   y7   y8
Sx1    1    7   13   19   25 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
Sx2    2    8   14   20   26 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
Sy4 <NA> <NA> <NA> <NA> <NA>    4    9   14   19   24   29   34   39
Sy5 <NA> <NA> <NA> <NA> <NA>    5   10   15   20   25   30   35   40

    x1 x2 x3 x4 x5 y1 y2 y3 y4 y5 y6 y7 y8
Sx1  1  7 13 19 25 NA NA NA NA NA NA NA NA
Sx2  2  8 14 20 26 NA NA NA NA NA NA NA NA
Sx3  3  9 15 21 27 NA NA NA NA NA NA NA NA
Sx4  4 10 16 22 28 NA NA NA NA NA NA NA NA
Sx5  5 11 17 23 29 NA NA NA NA NA NA NA NA
Sx6  6 12 18 24 30 NA NA NA NA NA NA NA NA
Sy1 NA NA NA NA NA  1  6 11 16 21 26 31 36
Sy2 NA NA NA NA NA  2  7 12 17 22 27 32 37
Sy3 NA NA NA NA NA  3  8 13 18 23 28 33 38
Sy4 NA NA NA NA NA  4  9 14 19 24 29 34 39
Sy5 NA NA NA NA NA  5 10 15 20 25 30 35 40
```

Or, can assume variable are in same order

R code

```
x <- matrix(1:30, ncol=5)
y <- matrix(1:40, ncol=8)
XY <- vJoin(x,y,  cnames=FALSE)
XY
```

```
XY
    x1 x2 x3 x4 x5 x6 x7 x8
Sx1  1  7 13 19 25 NA NA NA
Sx2  2  8 14 20 26 NA NA NA
Sx3  3  9 15 21 27 NA NA NA
Sx4  4 10 16 22 28 NA NA NA
Sx5  5 11 17 23 29 NA NA NA
Sx6  6 12 18 24 30 NA NA NA
Sy1  1  6 11 16 21 26 31 36
Sy2  2  7 12 17 22 27 32 37
Sy3  3  8 13 18 23 28 33 38
Sy4  4  9 14 19 24 29 34 39
Sy5  5 10 15 20 25 30 35 40
```

vJoin will also combine data.frames and matrices

R code

```
vJoin(bfi[1:5,1:6],x)
```

```
vJoin(bfi[1:5,1:6],x)
      A1 A2 A3 A4 A5 C1 y1 y2 y3 y4 y5
61617  2  4  3  4  4  2 NA NA NA NA NA
61618  2  4  5  2  5  5 NA NA NA NA NA
61620  5  4  5  4  4  4 NA NA NA NA NA
61621  4  4  6  5  5  4 NA NA NA NA NA
61622  2  3  3  4  5  4 NA NA NA NA NA
Sy1   NA NA NA NA NA NA  1  7 13 19 25
Sy2   NA NA NA NA NA NA  2  8 14 20 26
Sy3   NA NA NA NA NA NA  3  9 15 21 27
Sy4   NA NA NA NA NA NA  4 10 16 22 28
Sy5   NA NA NA NA NA NA  5 11 17 23 29
Sy6   NA NA NA NA NA NA  6 12 18 24 30
```

vJoin uses several core R functions

vJpoin just strings together a number of standard R functions

1. NCOL and NROW (ncol and nrow)
2. colnames, rownames
3. paste and paste0
4. unique
5. length
6. data.frame

How does vJoin work?

Examine the code

```
#Created 05/04/23  to combine two data frames byrows and columns
#slight modification 05/06/23 to create column names if missing
vJoin <- function(x,y, rnames=TRUE,cnames=TRUE) {
  n.x <- NCOL(x)
  n.y <- NCOL(y)
  n.obsx <- NROW(x)
  n.obsy <- NROW(y)
  if(is.null(colnames(x))) colnames(x)<- paste0("x",1:n.x)
  if(is.null(colnames(y))) if(cnames) {colnames(y) <- paste0("y",1:n.y)
          } else {colnames(y) <- paste0("x",1:n.y)}
  if(is.null(rownames(x))) rownames(x) <- paste0("Sx",1:n.obsx)
  if(is.null(rownames(y))) rownames(y) <- paste0("Sy",1:n.obsy)

  xy <- unique(c(colnames(x),colnames(y)))
  n.xy.col <- length(xy)
  if(!rnames) { new.names <- n.obsx + 1:n.obsy
  rownames(y) <- new.names}
  xy.rows  <- unique(c(rownames(x),rownames(y)))
  n.xy.rows <- length(xy.rows)

#now, put them together -- matching on row and column names
  new <- data.frame(matrix(NA, ncol=n.xy.col,nrow = n.xy.rows))
  colnames(new) <- xy
  rownames(new) <- xy.rows
  new[rownames(x),colnames(x)] <- x[rownames(x),colnames(x)]
  new[rownames(y),colnames(y)] <- y[rownames(y),colnames(y)]
  return(new)
  }
```

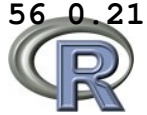
Sometime need to reorganize/recode variables

1. Consider education: High school, Some college, College, graduate work, graduate degree.
2. With numerical values of 1, 2, 3,4, 5 (in bfi)

```
bfi$education[50:60]
 [1] 3 3 4 4 3 4 5 3 5 2 5
 temp$education[50:60]
 [1] "College"         "College"         "graduate_work"   "graduate_work"   "College"
 [6] "graduate_work"   "graduate_degree" "College"         "graduate_degree" "Some_coll
[11] "graduate_degree"
```

3. So far, so good. But now describe them

```
 describe(bfi[26:28])
          vars    n  mean    sd median trimmed   mad min max range  skew kurtosis   se
gender       1 2800  1.67  0.47      2    1.71  0.00   1   2     1 -0.73    -1.47 0.01
education    2 2577  3.19  1.11      3    3.22  1.48   1   5     4 -0.05    -0.32 0.02
age          3 2800 28.78 11.13     26   27.43 10.38   3  86    83  1.02     0.56 0.21
> describe(temp[26:28])
           vars    n  mean    sd median trimmed   mad min max range  skew kurtosis   se
gender        1 2800  1.67  0.47      2    1.71  0.00   1   2     1 -0.73    -1.47 0.01
education*    2 2577  2.18  1.40      2    1.98  1.48   1   5     4  0.84    -0.66 0.03
age           3 2800 28.78 11.13     26   27.43 10.38   3  86    83  1.02     0.56 0.21
```



What happened?

1. The character values are sorted in terms of alphabetical order

```
table(bfi[27])
education
   1    2    3    4    5
 224  292 1249  394  418
> table(temp[27])
education
        College graduate_degree   graduate_work     High_school    Some_college
           1249              418              394              224              292
```

2. We need to recode the character strings into the appropriate numeric values.
3. Hence the `recode` function.

Change to numeric, and recode

R code

```
temp2 <- char2numeric(temp,flag=FALSE)
new.education <- c(3,5,4,1,2)
temp3 <- recode(temp2,27,isvalue=1:6, newvalue=new.education)
describe(temp3[26:28])
```

```
temp2 <- char2numeric(temp,flag=FALSE)
> new.education <- c(3,5,4,1,2)
> temp3 <- recode(temp2,27,isvalue=1:6, newvalue=new.education)
> describe(temp3[26:28])
          vars    n  mean    sd median trimmed   mad min max range  skew kurtosis   se
gender       1 2800  1.67  0.47      2    1.71  0.00   1   2     1 -0.73    -1.47 0.01
education    2 2577  3.19  1.11      3    3.22  1.48   1   5     4 -0.05    -0.32 0.02
age          3 2800 28.78 11.13     26   27.43 10.38   3  86    83  1.02     0.56 0.21

describe(bfi[26:28])
          vars    n  mean    sd median trimmed   mad min max range  skew kurtosis   se
gender       1 2800  1.67  0.47      2    1.71  0.00   1   2     1 -0.73    -1.47 0.01
education    2 2577  3.19  1.11      3    3.22  1.48   1   5     4 -0.05    -0.32 0.02
age          3 2800 28.78 11.13     26   27.43 10.38   3  86    83  1.02     0.56 0.21
```

Convert numbers to character strings

R code

```
n.names <- cs(one,two, three, four, five, six)
temp <-recode(bfi, where =1:6, isvalue=1:6, newvalue = n.names)
headTail(temp,from=1, to=8)
headTail(bfi,from=1, to=8)
```

```
n.names <- cs(one,two, three, four, five, six)
> temp <-recode(bfi, where =1:6, isvalue=1:6, newvalue = n.names)
> headTail(temp,from=1, to=8)
        A1    A2    A3    A4   A5   C1  C2  C3
61617  two  four three  four four  two   3   3
61618  two  four  five   two five five   4   4
61620 five  four  five  four four four   5   4
61621 four  four   six  five five four   4   3
...   <NA>  <NA>  <NA>  <NA> <NA> <NA> ... ...
67552  two  four  four three five  two   3   4
67556  two three  five   two five five   5   5
67559 five   two   two  four four five   5   5
67560  two three   one  four  two five   5   3
> headTail(bfi,from=1, to=8)
       A1  A2  A3  A4  A5  C1  C2  C3
61617   2   4   3   4   4   2   3   3
61618   2   4   5   2   5   5   4   4
61620   5   4   5   4   4   4   5   4
61621   4   4   6   5   5   4   4   3
...   ... ... ... ... ... ... ... ...
67552   2   4   4   3   5   2   3   4
67556   2   3   5   2   5   5   5   5
67559   5   2   2   4   4   5   5   5
67560   2   3   1   4   2   5   5   3
```

Unfortunately, converting back produces something wrong

R code

```
new.temp <- char2numeric(temp)
headTail(new.temp,from=1,to=8)
```

```
new.temp <- char2numeric(temp)
> headTail(new.temp,from=1,to=8)
      A1. A2. A3. A4. A5. C1.  C2  C3
61617   6   2   5   2   2   6   3   3
61618   6   2   1   6   1   1   4   4
61620   1   2   1   2   2   2   5   4
61621   2   2   4   1   1   2   4   3
...   ... ... ... ... ... ... ... ...
67552   6   2   2   5   1   6   3   4
67556   6   5   1   6   1   1   5   5
67559   1   6   6   2   2   1   5   5
67560   6   5   3   2   6   1   5   3
```

We need to recode the data

R code

```
new.map <- c(3,6,5,2,1,4)
correct.temp <- recode(new.temp,1:6, isvalue=new.map,newvalue=1:6)
headTail(correct.temp, from=1, to=8)
#or compare the two files
sum(correct.temp-bfi,na.rm=TRUE)
```

```
new.map <- c(3,6,5,2,1,4)
> correct.temp <- recode(new.temp,1:6, isvalue=new.map,newvalue=1:6)
> headTail(correct.temp, from=1, to=8)
      A1. A2. A3. A4. A5. C1.  C2  C3
61617   2   4   3   4   4   2   3   3
61618   2   4   5   2   5   5   4   4
61620   5   4   5   4   4   4   5   4
61621   4   4   6   5   5   4   4   3
...   ... ... ... ... ... ... ... ...
67552   2   4   4   3   5   2   3   4
67556   2   3   5   2   5   5   5   5
67559   5   2   2   4   4   5   5   5
67560   2   3   1   4   2   5   5   3
> #or compare the two files
> sum(correct.temp-bfi,na.rm=TRUE)
[1] 0
```

recode combines conventional R funtions

1. missing
2. unique (unique returns a vector, data frame or array like x but with duplicate elements/rows removed.)
3. unlist
4. any (Given a set of logical vectors, is at least one of the values true?)
5. max, min
6. for(var in seq) expr

How does recode work.

Examine the code

```
#added 5/6/23
 #code is truly ugly
 "recode" <- function(x,where,isvalue,newvalue) {
  if (missing(where))  where <-1:NCOL(x)
    all.values <- unique(unlist(x[,where]))
    if(!any(all.values %in% isvalue)) {stop ("data has more values than specified in isvalue")
    }
  maxv <- max(isvalue)
  minv <- min(isvalue)
  tempv <- isvalue + maxv + minv
  x[,where ]<- x[,where] + maxv + minv  #make these larger to allow swaps in place
  for (k in 1:length(where)) {where1 <- where[k]  #begin the where lop
          for(j in 1:NROW(x)) {   #replace across all cases
                for (i in (1: length(isvalue)) ) {#search one column for each possible value
                if(!is.na(x[j,where1])) {
                  if((x[j,where1] == tempv[i])) x[j,where1]   <- newvalue[i]}
                  } #end of values loop
          }  #end of subjects loop
  }  #end of where loop
  return(x)
}
```

scrub

A tedious part of data analysis is addressing the problem of miscoded data that need to be converted to NA or some other value. For a given data.frame or matrix, scrub will set all values of columns from=from to to=to that are less than a set (vector) of min values or more than a vector of max values to NA. Can also be used to do basic recoding of data for all values=isvalue to newvalue. Will also recode continuus variables into fewer categories. Will convert Nan, -Inf and Inf to NA

R code

```
x <- scrub(attitude,isvalue=55) #make all occurrences of 55 NA
x1 <- scrub(attitude, where=c(4,5,6), isvalue =c(30,40,50),
     newvalue = c(930,940,950)) #will do this for the 4th, 5th, and 6t
x2 <- scrub(attitude, where=c(4,4,4), isvalue =c(30,40,50),
            newvalue = c(930,940,950)) #will just do it for the 4th co
new <- scrub(attitude,1:3,cuts= c(10,40,50,60,100))  #change many valu
y <- scrub(attitude,where=2:4,min=c(20,30,40),max= c(120,110,100),isva
y1 <- scrub(attitude,where="learning",isvalue=55,newvalue=999) #change
y2 <- scrub(attitude,where="learning",min=45,newvalue=999) #change a c

y3 <- scrub(attitude,where="learning",isvalue=c(45,48),
    newvalue=999) #change a column by name look for multiple values in
```