Learning R: A Guide for Clinician Scientists

Data Wrangling (Basic-to-Intermediate)

Alvin D. Jeffery, PhD, RN

Last Updated 2018-05-02

Contents

An Overview & Assumptions

What is Data Wrangling?

- A newer data science term that essentially means data cleaning
- We commonly spend at least 80% of our 'analysis' preparing the data for the analysis

Where is the make_my_data_pretty() function?

- Although there are several packages and functions available, getting your data into the format specific for your analytical plan is individualized.
- There are several packages created specifically to help with data wrangling.
- You'll want to have the following 5 packages installed on your system: Hmisc, dplyr, tidyr, data.table, & lubridate.

Assumptions

- You have at least a generic analytical plan in mind.
- You are comfortable with being exposed to multiple ways of achieving the same outcome.
- You have enough working knowledge of R to load data and manipulate objects.

Opening & Exploring

We'll get started by opening & getting a 'feel' for what data we have. I have created a 'messy' data set that is a smaller version of a real data set I am currently cleaning/analyzing. The data set is rapid response team (RRT) data collected from ICU nurse practitioners that responded to RRT calls on the floor at an academic medical center. Only a few rows and columns are provided for teaching purposes.

```r
library(Hmisc); library(data.table); library(dplyr); library(tidyr); library(lubridate)
```

```r
# fread() is the "fast and friendly file finagler"
# works more reliably that read.table()/read.csv() & although csv.get() states it calls
# fread(), I haven't found this to be the case.

w <- fread('./messy_rrt.csv',
           data.table = FALSE) # I'm not a huge fan of data.table's, personally

head(w) # review some of the data...could also use View()
str(w) # what is the object type, and what are our column classes?
```

```r
# first, grab the longer name & keep it for labels later
w <- cleanup.import(w, labels=colnames(w))
w <- upData(w, # from the Hmisc package
            lowernames = TRUE, # make variable names lowercase if keeping them
            charfactor = TRUE) # convert character if < n/2 unique values

str(w) # always look at your data after making changes!

# because these are longer names, let's just replace them easily
colnames(w) <- c('id', 'dob', 'f_name', 'l_name', 'sex', 'location', 'ward_loc1',
                 'ward_loc2', 'ward_loc3', 'date', 'service', 'rrt_caller', 'code',
                 'minutes', 'md_resident', 'md_fellow', 'md_attending')
str(w)
```

Date Manipulation (Demo 1)

```r
# make a copy as we begin manipulation
w2 <- cleanup.import(w, datevars = c('dob', 'date'), dateformat='%m/%d/%y')
str(w2)
# well, that date transformation didn't work!

w2 <- w # going back to make a copy of the 'good' dataframe before next step...
w2$dob <- as.Date(w2$dob, origin='1970-01-01') # produces an error!

# ugh! still not working!

# go back to the cleanup.import step and set a 'rule' for unrealistic data
# (notice that I'm starting with the 'w' dataframe again, not w2)
w2 <- cleanup.import(w, datevars = c('dob', 'date'), dateformat='%m/%d/%y')
str(w2)

w2$dob <- ifelse(w2$dob >= Sys.Date(), # conditional statement
                 w2$dob - 100*365.25, # if TRUE...subtract 100 years
                 w2$dob) # otherwise, if FALSE...keep it the same
head(w2) # getting closer!

w2$dob <- as.Date(w2$dob, origin='1970-01-01')
head(w2)
# YAY!!!

w2$age <- Sys.Date() - w2$dob # create age as of TODAY
w2$age <- round(w2$age/365.25, 2) # convert to years & round
```

Test Your Knowledge: Rather than setting the patient’s age as of today, calculate the patient’s age at the date the RRT was called.

Re-Cap and Apply (Demo 2)

```r
w <- fread('./messy_rrt.csv',
           data.table = FALSE, na.strings = '')
w <- cleanup.import(w, labels=colnames(w),
                    datevars = c('Date of Birth', 'Date of RRT'),
                    dateformat='%m/%d/%y')
w <- upData(w, lowernames = TRUE, charfactor = TRUE)

colnames(w) <- c('id', 'dob', 'f_name', 'l_name', 'sex', 'location', 'ward_loc1',
                 'ward_loc2', 'ward_loc3', 'date', 'service', 'rrt_caller', 'code',
                 'minutes', 'md_resident', 'md_fellow', 'md_attending')

# calculating age at the time of the RRT
w$dob <- ifelse(w$dob >= Sys.Date(), # conditional statement
                 w$dob - 100*365.25, # if TRUE...subtract 100 years
                 w$dob) # otherwise, if FALSE...keep it the same
w$dob <- as.Date(w$dob, origin='1970-01-01')

w$age <- w$date - w$dob
w$age <- as.numeric(w$age)/365.25 # should provide age in years
w$age <- round(w$age, 2)

# check to see if truly NA before setting up rules
is.na(w$ward_loc1)
# let's change the 'na.strings' argument of the fread() function: na.strings=""

# some had converted to factors during upData(), so we can convert back to make it easier to read
w$ward_loc1 <- as.character(w$ward_loc1)
w$ward_loc2 <- as.character(w$ward_loc2)
w$ward_loc3 <- as.character(w$ward_loc3)

# merge ward_loc1, ward_loc2, and ward_loc3 into 1 column
w$ward_loc <- ifelse(is.na(w$ward_loc1), # if loc1 empty, look in loc2...
                       ifelse(is.na(w$ward_loc2), # if loc2 empty, use loc3...
                              w$ward_loc3,
                              w$ward_loc2), # use loc2 if it's not empty
                     w$ward_loc1) # use loc1 if it's not empty
label(w$ward_loc) <- "Ward Location?"
temp <- select(w, ward_loc1:ward_loc3, ward_loc) # check to ensure working
rm(temp)
# drop original ward_loc values
w <- select(w, -ward_loc1, -ward_loc2, -ward_loc3)

colnames(w) # find column numbers for variables of interest
for (i in 12:14) { # turn 'checked' and 'uncheck' into 1/0 for md_resident (12)
                   # to md_attending (14)
  w[,i] <- ifelse(w[,i]=='Checked', 1, 0)
}

# look for duplicates
duplicated(w) # entire row
sum(duplicated(select(w, id:l_name))) # for select variables
```

```r
dup <- duplicated(select(w, id:l_name)) # store
w[which(dup==TRUE), ]
w[which(w$id==1905), ] # notice that the RRT dates are different!
w <- arrange(w, id, date) # use desc(date) to arrange in opposite direction
w <- distinct(w, id, .keep_all = TRUE) # let's keep the incident RRT date

# we also notice that Amelia has been classified as a 'male' - let's change this
w$sex <- ifelse(w$f_name=="Amelia", 1, w$sex)
w$sex <- ifelse(w$f_name=="Alvin" | w$f_name=="Thomas", 2, w$sex)
w$sex <- factor(w$sex, labels = c("Female", "Male"))
```

dplyr and Chaining

```r
w <- fread('./messy_rrt.csv',
           data.table = FALSE, na.strings = '')
w <- cleanup.import(w, labels=colnames(w),
                    datevars = c('Date of Birth', 'Date of RRT'),
                    dateformat='%m/%d/%y')
w <- upData(w, lowernames = TRUE, charfactor = TRUE)

colnames(w) <- c('id', 'dob', 'f_name', 'l_name', 'sex', 'location', 'ward_loc1',
                 'ward_loc2', 'ward_loc3', 'date', 'service', 'rrt_caller', 'code',
                 'minutes', 'md_resident', 'md_fellow', 'md_attending')

w2 <- w %>% # begin chain
  mutate(dob = ifelse(dob >= Sys.Date(), dob - 100*365.25, dob)) %>%
  mutate(dob = as.Date(dob, origin='1970-01-01')) %>%
  mutate(age = as.numeric(date - dob)/365.25) %>% # age in years
  mutate(age = round(age, 2)) %>%
  mutate(ward_loc1 = as.character(ward_loc1)) %>%
  mutate(ward_loc2 = as.character(ward_loc2)) %>%
  mutate(ward_loc3 = as.character(ward_loc3)) %>%
  mutate(ward_loc = ifelse(is.na(ward_loc1), # if loc1 empty, look in loc2...
                    ifelse(is.na(ward_loc2), # if loc2 empty, use loc3...
                           ward_loc3,
                           ward_loc2), # use loc2 if it's not empty
                    ward_loc1)) %>% # use loc1 if it's not empty
  select(-ward_loc1, -ward_loc2, -ward_loc3) %>%
  mutate(md_resident = ifelse(md_resident == 'Checked', 1, 0)) %>%
  mutate(md_fellow = ifelse(md_fellow == 'Checked', 1, 0)) %>%
  mutate(md_attending = ifelse(md_attending == 'Checked', 1, 0)) %>%
  arrange(id, date) %>%
  distinct(id, .keep_all=TRUE) %>% # remove the duplicate from Amelia's record
  mutate(sex = ifelse(f_name == "Amelia", 1, sex)) %>%
  mutate(sex = ifelse(f_name == "Alvin" | f_name == "Thomas", 2, sex)) %>%
  mutate(sex = factor(sex, labels = c("Female", "Male")))

# can do some tests, if desired
length(unique(w$id)) == length(unique((w2$id)))
min(w$date) == min(w2$date)
```