Manipulating Data in R

Andrew Jaffe

July 13, 2016

Reshaping Data

In this module, we will show you how to:

1. Reshaping data from long (tall) to wide (fat)
2. Reshaping data from wide (fat) to long (tall)
3. Merging Data

Setup

We will show you how to do each operation in base R then show you how to use the `dplyr` or `tidyr` package to do the same operation (if applicable).
See the "Data Wrangling Cheat Sheet using `dplyr` and `tidyr`":

- `https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf`

Data used: Charm City Circulator

http://www.aejaffe.com/summerR_2016/data/Charm_City_Circulator_Ridership.csv

```
circ = read.csv("../data/Charm_City_Circulator_Ridership.cs
                as.is = TRUE)
head(circ, 2)
```

```
      day       date orangeBoardings orangeAlightings orang
1  Monday 01/11/2010             877             1027
2 Tuesday 01/12/2010             777              815
  purpleBoardings purpleAlightings purpleAverage greenBoar
1              NA               NA            NA
2              NA               NA            NA
  greenAlightings greenAverage bannerBoardings bannerAlight
1              NA           NA              NA
2              NA           NA              NA
  bannerAverage daily
1            NA   952
2            NA   796
```

Creating a Date class from a character date

```
library(lubridate) # great for dates!
library(dplyr) # mutate/summarise functions
circ = mutate(circ, date = mdy(date))
sum( is.na(circ$date) ) # all converted correctly
```

```
[1] 0
```

```
head(circ$date)
```

```
[1] "2010-01-11" "2010-01-12" "2010-01-13" "2010-01-14" "20
[6] "2010-01-16"
```

```
class(circ$date)
```

```
[1] "Date"
```

Making column names a little more separated

We will use `str_replace` from `stringr` to put periods in the column names.

```r
library(stringr)
cn = colnames(circ)
cn = cn %>%
  str_replace("Board", ".Board") %>%
  str_replace("Alight", ".Alight") %>%
  str_replace("Average", ".Average")
colnames(circ) = cn
cn
```

```
 [1] "day"                "date"               "orange.Boardi
 [4] "orange.Alightings" "orange.Average"    "purple.Boardi
 [7] "purple.Alightings" "purple.Average"    "green.Boardin
[10] "green.Alightings"  "green.Average"     "banner.Boardi
[13] "banner.Alightings" "banner.Average"    "daily"
```

Reshaping data from wide (fat) to long (tall)

See `http://www.cookbook-r.com/Manipulating_data/Converting_data_between_wide_and_long_format/`

- Wide - multiple columns per observation
  - e.g. visit1, visit2, visit3

```
  id visit1 visit2 visit3
1  1     10      4      3
2  2      5      6     NA
```

- Long - multiple rows per observation

```
  id visit value
1  1     1    10
2  1     2     4
3  1     3     3
4  2     1     5
5  2     2     6
```

Reshaping data from wide (fat) to long (tall): base R

The `reshape` command exists. It is a **confusing** function. Don't use it.

Reshaping data from wide (fat) to long (tall): tidyr

`tidyr::gather` - puts column data into rows.
We want the column names into “`var`” variable in the output dataset and the value in “`number`” variable. We then describe which columns we want to “gather:”

```r
library(tidyr)
long = gather(circ, key = "var", value = "number",
              starts_with("orange"),
              starts_with("purple"),
              starts_with("green"),
              starts_with("banner"))
head(long, 2)
```

```
      day       date daily              var number
1  Monday 2010-01-11   952 orange.Boardings    877
2 Tuesday 2010-01-12   796 orange.Boardings    777
```

```r
table(long$var)
```

Reshaping data from wide (fat) to long (tall): tidyr

Now each var is boardings, averages, or alightings. We want to separate these so we can have these by line.

```
long = separate_(long, "var",
                 into = c("line", "type"),
                 sep = "[.]")
head(long, 3)
```

```
        day       date  daily   line      type number
1    Monday 2010-01-11  952.0 orange Boardings    877
2   Tuesday 2010-01-12  796.0 orange Boardings    777
3 Wednesday 2010-01-13 1211.5 orange Boardings   1203
```

```
unique(long$line)
```

```
[1] "orange" "purple" "green"  "banner"
```

```
unique(long$type)
```

```
[1] "Boardings"  "Alightings" "Average"
```

Finding the First (or Last) record

```
long = long %>% filter(!is.na(number) & number > 0)
first_and_last = long %>% arrange(date) %>% # arrange by da
  filter(type %in% "Boardings") %>% # keep boardings only
  group_by(line) %>% # group by line
  slice( c(1, n())) # select ("slice") first and last (n()
first_and_last %>%  head(4)
```

```
Source: local data frame [4 x 6]
Groups: line [2]

      day       date   daily   line       type number
    <chr>     <date>   <dbl>  <chr>      <chr>  <dbl>
1  Monday 2012-06-04 13342.5 banner Boardings    520
2  Friday 2013-03-01      NA banner Boardings    817
3 Tuesday 2011-11-01  8873.0  green Boardings    887
4  Friday 2013-03-01      NA  green Boardings   2592
```

Reshaping data from long (tall) to wide (fat): tidyr

In tidyr, the spread function spreads rows into columns. Now we have a long data set, but we want to separate the Average, Alightings and Boardings into different columns:

```
# have to remove missing days
wide = filter(long, !is.na(date))
wide = spread(wide, type, number)
head(wide)
```

```
     day       date  daily   line Alightings Average Boardi
1 Friday 2010-01-15 1644.0 orange       1643  1644.0      1
2 Friday 2010-01-22 1394.5 orange       1388  1394.5      1
3 Friday 2010-01-29 1332.0 orange       1322  1332.0      1
4 Friday 2010-02-05 1217.5 orange       1204  1217.5      1
5 Friday 2010-02-12  671.0 orange        678   671.0
6 Friday 2010-02-19 1642.0 orange       1647  1642.0      1
```

Reshaping data from long (tall) to wide (fat): tidyr

We can use rowSums to see if any values in the row is NA and keep if the row, which is a combination of date and line type has any non-missing data.

```
# wide = wide %>%
#     select(Alightings, Average, Boardings) %>%
#     mutate(good = rowSums(is.na(.)) > 0)
namat = !is.na(select(wide, Alightings, Average, Boardings)
head(namat)
```

```
  Alightings Average Boardings
1       TRUE    TRUE      TRUE
2       TRUE    TRUE      TRUE
3       TRUE    TRUE      TRUE
4       TRUE    TRUE      TRUE
5       TRUE    TRUE      TRUE
6       TRUE    TRUE      TRUE
```

```
wide$good = rowSums(namat) > 0
```

Reshaping data from long (tall) to wide (fat): tidyr

Now we can filter only the good rows and delete the good column.

```r
wide = filter(wide, good) %>% select(-good)
head(wide)
```

```
     day       date  daily   line Alightings Average Boardi
1 Friday 2010-01-15 1644.0 orange       1643  1644.0      1
2 Friday 2010-01-22 1394.5 orange       1388  1394.5      1
3 Friday 2010-01-29 1332.0 orange       1322  1332.0      1
4 Friday 2010-02-05 1217.5 orange       1204  1217.5      1
5 Friday 2010-02-12  671.0 orange        678   671.0
6 Friday 2010-02-19 1642.0 orange       1647  1642.0      1
```