Automating Data Exploration with R

Dates

Brief Look at Date Formats

What is an R-friendly date format? The full year, month, day separated by either a forward slash or a dash:

```
as.Date('1950-06-16')
```

```
## [1] "1950-06-16"
```

```
as.Date('1950-6-16')
```

```
## [1] "1950-06-16"
```

```
as.Date('1950/02/17')
```

```
## [1] "1950-02-17"
```

```
as.Date('50/06/16')
```

```
## [1] "0050-06-16"
```

Getting system time:

```
print(Sys.Date())
```

```
## [1] "2016-03-31"
```

```
class(Sys.Date())
```

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

```
print(Sys.time())
```

```
## [1] "2016-03-31 17:12:37 PDT"
```

```
class(Sys.time())
```

```
## [1] "POSIXct" "POSIXt"
```

POSIX

`POSIX` functions manipulate objects of classes “POSIXlt” and “POSIXct” representing calendar dates and times. Though we’ll build our pipeline using only dates, it’s good to know about time-based functions (DateTimeClasses (https://stat.ethz.ch/R-manual/R-devel/library/base/html/DateTimeClasses.html)) as well. `POSIXct` is the total seconds since UNIX time, and `POSIXlt` converts time to various formats (these are from the help file, see ?POSIXlt for plenty more):

```
## These may not be correct names on your system
as.POSIXlt(Sys.time(), "America/New_York")  # in New York
```

```
## [1] "2016-03-31 20:12:37 EDT"
```

```
as.POSIXlt(Sys.time(), "EST5EDT")           # alternative.
```

```
## [1] "2016-03-31 20:12:37 EDT"
```

```
as.POSIXlt(Sys.time(), "EST" )   # somewhere in Eastern Canada
```

```
## [1] "2016-03-31 19:12:37 EST"
```

```
as.POSIXlt(Sys.time(), "HST")    # in Hawaii
```

```
## [1] "2016-03-31 14:12:37 HST"
```

```
as.POSIXlt(Sys.time(), "Australia/Darwin")
```

```
## [1] "2016-04-01 09:42:37 ACST"
```

Note: for list of time zones, see: Date and Time Gateway - Timezone Selector (http://twiki.org/cgibin/xtra/tzdatepick.html)

Custom Dates

If the you date isn't in the correct order then you can use the `format` parameter to shape it correctly (see ?format.Date for more details):

```
as.Date('1/12/2001',format='%d/%m/%Y')
```

```
## [1] "2001-12-01"
```

```
as.Date('April 26, 2001',format='%B %d, %Y')
```

```
## [1] "2001-04-26"
```

Check out plenty more examples from Berkeley's Concepts in Computing with Data class Dates and Times in R (http://www.stat.berkeley.edu/~s133/dates.html)

So, if you have dates that are in the base R format (year-month-day) then you can use either the `readr` package to correctly cast them, but what about other formats?

```
mix_dataset <- data.frame(
                id=c(10,20,30,40,50),
                gender=c('male','female','female','male','female'),
                some_date=c('01/11/2012','04/12/2012','28/02/2013','17/06/2014','0
8/03/2015'),
                value=c(12.34, 32.2, 24.3, 83.1, 8.32),
                outcome=c(1,1,0,0,0))

write.csv(mix_dataset, 'mix_dataset.csv', row.names=FALSE)

library(readr)
mix_dataset <- read_csv('mix_dataset.csv')
mix_dataset$some_date <-  as.Date(mix_dataset$some_date, format="%d/%m/%Y")
str(mix_dataset$some_date)
```

```
##  Date[1:5], format: "2012-11-01" "2012-12-04" "2013-02-28" "2014-06-17" ...
```

Let's wrap the above code into a handy function for our pipeline:

```
Fix_Date_Features <- function(data_set) {
     text_features <- c(names(data_set[sapply(data_set, is.character)]), names(dat
a_set[sapply(data_set, is.factor)]))
     for (feature_name in text_features) {
          feature_vector <- as.character(data_set[,feature_name])
          # assuming date pattern: '01/11/2012'
          date_pattern <- '[0-9][0-9]/[0-9][0-9]/[0-9][0-9][0-9][0-9]'
          if (max(nchar(feature_vector)) == 10) {
               if (sum(grepl(date_pattern, feature_vector)) > 0) {
                    print(paste('Casting feature to date:',feature_name))
                    data_set[,feature_name] <-  as.Date(feature_vector, format="%
d/%m/%Y")
               }
          }
     }
     return (data_set)
}
```

Pipeline Check

In case you have to deal with dates in a non-standard format, here is code to run through every feature and use `grepl` to identify and transform them:

```
path_and_file_name <- 'mix_dataset.csv'
# quick peek at top lines
print(readLines(path_and_file_name, n=5))
```

```
## [1] "\"id\",\"gender\",\"some_date\",\"value\",\"outcome\""
## [2] "10,\"male\",\"01/11/2012\",12.34,1"
## [3] "20,\"female\",\"04/12/2012\",32.2,1"
## [4] "30,\"female\",\"28/02/2013\",24.3,0"
## [5] "40,\"male\",\"17/06/2014\",83.1,0"
```

```
# format dates
mix_dataset <- read.csv(path_and_file_name, stringsAsFactor=FALSE)
print(head(Fix_Date_Features(mix_dataset)))
```

```
## [1] "Casting feature to date: some_date"
##   id gender  some_date value outcome
## 1 10   male 2012-11-01 12.34       1
## 2 20 female 2012-12-04 32.20       1
## 3 30 female 2013-02-28 24.30       0
## 4 40   male 2014-06-17 83.10       0
## 5 50 female 2015-03-08  8.32       0
```