# Apply Data Wrangling To My Project

*Carl Larson*

*2/1/2018*

### Intro Capstone Project Data Wrangling

The data, in this case the gameflow table for basketball, needs some cleaning.

The time variable populates the x-axis, as is intuitive, and is central to the algorithm.

There are 48 minutes in a standard (non-overtime) NBA game, and the clock resets at the end of the first three 12-minute quarters.

Because of this clock resetting, we will need to format the eventual time variable so it can be graphed as a linearly timed game.

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(tidyr)
```

```
## Loading required package: tidyr
```

```
## Warning: package 'tidyr' was built under R version 3.4.2
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(reshape2)
```

```
## Loading required package: reshape2
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
df <- read.csv("/Users/EagleFace/Downloads/Gameflow of Bulls76ersApr282012.csv", header = FALSE)
head(df)
```

```
##   V1 V2    V3 V4    V5             V6     V7             V8
## 1  0  0 11:38  1 76ers    Elton Brand DefReb April 28, 2012
```

```
## 2  3  0 11:26  1 76ers   Jrue Holiday  Make3
## 3  3  0 11:26  1 76ers Andre Iguodala Assist
## 4  3  2 11:07  1 Bulls      Luol Deng  Make2
## 5  3  2 11:07  1 Bulls  Carlos Boozer Assist
## 6  3  2 10:44  1 76ers    Lavoy Allen  miss2
```

```
tail(df)
```

```
##     V1  V2   V3 V4 V5 V6 V7 V8
## 436 89 101 0:37  4
## 437 89 101 0:36  4
## 438 89 103 0:21  4
## 439 89 103 0:21  4
## 440 91 103 0:09  4
## 441 91 103 0:09  4
```

As shown above, the clock is V3 and the quarter is V4.

So ultimately the time variable we need is a function of these two.

```
df <- df[,1:4]
gf <- separate(df, V3, c("MIN", "SEC"), sep = ":", remove = TRUE)
head(gf)
```

```
##   V1 V2 MIN SEC V4
## 1  0  0  11  38  1
## 2  3  0  11  26  1
## 3  3  0  11  26  1
## 4  3  2  11  07  1
## 5  3  2  11  07  1
## 6  3  2  10  44  1
```

Now we have cleaned-up minutes and seconds columns.

```
#Next we need to operate arithmetically on the time columns,
#so we need them as numeric
gf$MIN <- as.numeric(gf$MIN)
gf$SEC <- as.numeric(gf$SEC)

#Now for the task of combining the columns for minutes and seconds into
#one single column representing total minutes left, in decimal form.
ft <- numeric(length = nrow(df))

for(i in 1:(nrow(gf))){
  while(gf$V4[i] == 1){
    ft[i] <- (gf$MIN[i] + (gf$SEC[i]/60) + 36)
      (i=i+1)}

  while(gf$V4[i] == 2){
    ft[i] <- (gf$MIN[i] + (gf$SEC[i]/60) + 24)
      (i=i+1)}

  while(gf$V4[i] == 3){
    ft[i] <- (gf$MIN[i] + (gf$SEC[i]/60) + 12)
      (i=i+1)}

  while(gf$V4[i] == 4){
```

```
    ft[i] <- (gf$MIN[i] + (gf$SEC[i]/60))
    if(i==nrow(gf)){break}
    else{
      (i=i+1)}}}
```

Just to take a look, we can add this vector "ft" back into the dataframe "df" and then see a glimpse of its final few observations.

```
#Adding this column to the dataframe and taking a look at it.
df$ft <- ft
tail(df)
```

```
##     V1  V2   V3 V4        ft
## 436 89 101 0:37  4 0.6166667
## 437 89 101 0:36  4 0.6000000
## 438 89 103 0:21  4 0.3500000
## 439 89 103 0:21  4 0.3500000
## 440 91 103 0:09  4 0.1500000
## 441 91 103 0:09  4 0.1500000
```

Now our dataframe has exactly what we want, for the time variable, minutes left, in its column "ft." We can see that as the clock counts down seconds, ft is counting down decimalized minutes, which is the formatting we need for the eventual algorithm.