IS 605 - Assignment 7

Dan Fanelli

March 16, 2016

Mean and SD functions:

Please write a function to compute the expected value and standard deviation of an array of values. Compare your results with that of R's mean and std functions

Basic Mean and SD Function

```r
mean_and_sd <- function(values, use_n_minus_1){
  # simple mean calc: sum em up, and divide by length
  my_mean <- sum(values)/length(values)
  # get the squared differences between the values and their mean.
  # Taking the square root later.
  delta_squared <- (values - my_mean)*(values - my_mean);
  # sum up the average of those delta_squareds:
  # get the average by summing deltas and dividing by n+1
  # its n+1 because when lots of numbers, they assume that one of
  # them HIT the mean, so don't need to account for it...)
  # take the sqrt of that sum (since we had squared before...)
  denominator <- length(values)
  if(use_n_minus_1){
    denominator <- length(values)-1
  }
  my_sd <- sqrt(sum(delta_squared)/denominator)

  cat(paste("my_mean = ", my_mean, "my_sd = ", my_sd, "\n"))
  cat(paste("R__mean = ", mean(values), "R__sd = ", sd(values), "\n"))
}

mean_and_sd(c(1,2,3,4,5), TRUE)
```

```
## my_mean =  3 my_sd =  1.58113883008419
## R__mean =  3 R__sd =  1.58113883008419
```

```r
mean_and_sd(c(-42,59,-10,-35,22), TRUE)
```

```
## my_mean =  -1.2 my_sd =  41.9726101165987
## R__mean =  -1.2 R__sd =  41.9726101165987
```

```r
mean_and_sd(c(1,2,3,4,5), FALSE)
```

```
## my_mean =  3 my_sd =  1.4142135623731
## R__mean =  3 R__sd =  1.58113883008419
```

```r
mean_and_sd(c(-42,59,-10,-35,22), FALSE)
```

```
## my_mean =  -1.2 my_sd =  37.541443765524
## R__mean =  -1.2 R__sd =  41.9726101165987
```

Now, consider that instead of being able to neatly fit the values in memory in an array, you have an infinite stream of numbers coming by. How would you estimate the mean and standard deviation of such a stream? Your function should be able to return the current estimate of the mean and standard deviation at any time it is asked. Your program should maintain these current estimates and return them back at any invocation of these functions

Rolling Mean and SD Function

```r
CURRENT_SUM <- 0
CURRENT_SUM_X_SQUARED <- 0
CURRENT_N <- 0

# Keep track of the current rolling sum, rolling sum of the x squareds, and n
rolling_mean_and_sd <- function(next_value, print_now){
  CURRENT_SUM <<- CURRENT_SUM + next_value
  CURRENT_N <<- CURRENT_N + 1
  CURRENT_SUM_X_SQUARED <<- CURRENT_SUM_X_SQUARED + (next_value * next_value)

  my_rolling_mean <- CURRENT_SUM/CURRENT_N
  my_rolling_sd <- sqrt(CURRENT_SUM_X_SQUARED / (CURRENT_N) - (my_rolling_mean * my_rolling_mean))

  if(print_now){
    cat(paste("my_rolling_mean = ", my_rolling_mean, "my_rolling_sd = ", my_rolling_sd, "\n"))
  }
}
```

Test It Out:

```r
cat(paste("R_mean = ", mean(c(1,2,3,4,5)), "R_sd = ", sd(c(1,2,3,4,5)), "\n"))
```

```
## R_mean =  3 R_sd =  1.58113883008419
```

```r
mean_and_sd(c(1,2,3,4,5), TRUE)
```

```
## my_mean =  3 my_sd =  1.58113883008419
## R__mean =  3 R__sd =  1.58113883008419
```

```r
mean_and_sd(c(1,2,3,4,5), FALSE)
```

```
## my_mean =  3 my_sd =  1.4142135623731
## R__mean =  3 R__sd =  1.58113883008419
```

```r
rolling_mean_and_sd(1, FALSE)
rolling_mean_and_sd(2, FALSE)
rolling_mean_and_sd(3, FALSE)
rolling_mean_and_sd(4, FALSE)
rolling_mean_and_sd(5, TRUE)
```

```
## my_rolling_mean =  3 my_rolling_sd =  1.4142135623731
```

```r
cat(paste("R_mean = ", mean(c(-42,59,-10,-35,22)), "R_sd = ", sd(c(-42,59,-10,-35,22)), "\n"))
```

```
## R_mean =  -1.2 R_sd =  41.9726101165987
```

```r
mean_and_sd(c(-42,59,-10,-35,22), TRUE)
```

```
## my_mean =  -1.2 my_sd =  41.9726101165987
## R__mean =  -1.2 R__sd =  41.9726101165987
```

```r
mean_and_sd(c(-42,59,-10,-35,22), FALSE)
```

```
## my_mean =  -1.2 my_sd =  37.541443765524
## R__mean =  -1.2 R__sd =  41.9726101165987
```

```r
rolling_mean_and_sd(-42, FALSE)
rolling_mean_and_sd(59, FALSE)
rolling_mean_and_sd(-10, FALSE)
rolling_mean_and_sd(-35, FALSE)
rolling_mean_and_sd(22, TRUE)
```

```
## my_rolling_mean =  0.9 my_rolling_sd =  26.6475139553392
```

Conclusion:

Basically, you can do the Standard Deviation calculation dividing by both n or n-1. My mean is the same as R's mean when I say "use the n-1". However my rolling mean calculation only seems to work for my "use the n" calculation.