# Streaming mean and standard deviation

*Giri Iyengar*

## Variance of a random variable

We can factor out the Variance formula thus.

$$V = E[(X-\mu)^2] = E[X^2 - 2X\mu + \mu^2] = E[X^2] - 2\mu^2 + \mu^2 = E[X^2] - \mu^2$$

## Streaming calculations

We need only the following quantities to be stored to calculate the mean and standard deviation exactly

- $\Sigma X$
- $\Sigma X^2$
- $n$: number of values seen so far

This works when $n$ is small. Eventually, we run out of the numerical limits of the underlying system. When that happens, we have to come up with alternative strategies. Here is a commonly used approach – work out the calculations in logarithm space. This approach is extremely suitable when you are dealing with really small numbers such as probabilities. If we know that all our numbers are going to be strictly positive, then we can safely work in log-space.

## Calculations in log-space

Let's assume that instead of working with X, we work with $\ln X$. Has natural advantages when dealing with large number of values.

- Let's assume that we have two numbers $A$ and $B$ that we want to add
- Further, $A$ is larger than $B$. E.g. $A$ could represent the sum and $B$ could be the current value
- We want to perform $e^{\ln A} + e^{\ln B}$ instead of $A + B$

## Streaming mean in log-space

- We want to store the sum in log space as well

$$\ln(e^{\ln A} + e^{\ln B}) = \ln e^{\ln A}(1 + e^{\ln B - \ln A}) = \ln A + \ln(1 + e^{\ln B - \ln A})$$

- We assume here that $A > B$
- Therefore $\ln B - \ln A$ is negative and $e^{\ln B - \ln A}$ is a small number
- We can do Taylor series expansion of $\ln(1+z)$ where $z = e^{\ln B - \ln A}$
- $\ln(1+z) \approx z - \frac{1}{2}z^2 + \frac{1}{3}z^3 - \frac{1}{4}z^4 + \ldots$
- Similarly, $\ln(1-z) \approx -z - \frac{1}{2}z^2 - \frac{1}{3}z^3 - \frac{1}{4}z^4 + \ldots$

As long as the number we are adding is relatively small compared with the accumulator, we can get a reasonably accurate Taylor series expansion. We can always detect when Taylor series approximation is not valid and switch to a different mode if required. Let's assume that we are dealing with the case where Taylor series approximation is reasonably valid. We can ensure this by doing normal arithmetic while $n$ is small and switch to log-space calculations when $n$ gets large.

### Putting it all together

```
# Logarithmic adding of a small value to a large value
log_add <- function(lnAccum,lnVal) {
  diff <- lnVal - lnAccum
    z    <- exp(diff)
    appx <- z - 0.5*z*z + z*z*z/3.0 # Retain only the first few Taylor series terms
    lnAccum + appx
}

# Logarithmic subtracting of a small value from a large value
log_min <- function(lnAccum,lnVal) {
    diff <- lnVal - lnAccum
    z    <- exp(diff)
    appx <- - z - 0.5*z*z - z*z*z/3.0 # Retain only the first few Taylor series terms
    lnAccum + appx
}

# Helper to convert statistics to log space if needed
convert_stats_if_needed <- function(stats) {
    max_linear_threshold <- 10000

    if (stats$N > max_linear_threshold) {
        # In this invocation, we need to convert to logarithmic statistics
        stats$lnSumX  <- log(stats$sumX)
        stats$lnSumX2 <- log(stats$sumX2)
        stats$sumX    <- NA
    stats$sumX2   <- NA
    }
    stats
}

# Streaming stats that performs log space calculations as needed.
streaming_stats <- function(stats,values) {
    max_linear_threshold <- 10000

    if (stats$N < max_linear_threshold) {
        stats$N     <- stats$N + length(values)
        stats$sumX  <- stats$sumX  + sum(values)
        stats$sumX2 <- stats$sumX2 + sum(values**2)

        stats$mu    <- stats$sumX/stats$N
        stats$std   <- sqrt((stats$sumX2 - (stats$sumX**2)/stats$N)/(stats$N - 1))

        stats <- convert_stats_if_needed(stats)
    } else {
        stats$N     <- stats$N + length(values)
        lv  <- log(sum(values))
        lv2 <- log(sum(values**2))

        # Now, accumulate them away
        stats$lnSumX  <- log_add(stats$lnSumX,lv)
        stats$lnSumX2 <- log_add(stats$lnSumX2,lv2)
```

```
    # Mean
    lnN  <- log(stats$N)
    lnN1 <- log(stats$N-1)
        stats$mu  <- exp(stats$lnSumX - lnN)

    # Compute the different parts of the standard deviation
        lnV1 <- stats$lnSumX2 - lnN1
        lnV2 <- 2.0*stats$lnSumX - lnN - lnN1

    # Convert to real space as lnV1 and lnV2 are comparable
        stats$std <- sqrt(exp(lnV1) - exp(lnV2))
  }

    stats
}
```

### Let's test this out

```
set.seed(42)
stats  <- data.frame(N=0,sumX=0,sumX2=0,lnSumX=0,lnSumX2=0,mu=0,std=0)
values <- rnorm(1000,mean=5,sd=2)
(stats  <- streaming_stats(stats,values))
```

```
##      N     sumX    sumX2 lnSumX lnSumX2       mu      std
## 1 1000 4948.351 28502.35      0       0 4.948351 2.005042
```

```
# Add another 8000 examples
values <- rnorm(8000,mean=5,sd=2)
(stats  <- streaming_stats(stats,values))
```

```
##      N     sumX    sumX2 lnSumX lnSumX2       mu      std
## 1 9000 44839.87 259850.3      0       0 4.982208 2.012536
```

```
# Add another 2000 examples. We trigger log space now.
values <- rnorm(2000,mean=5,sd=2)
(stats  <- streaming_stats(stats,values))
```

```
##       N sumX sumX2   lnSumX lnSumX2       mu      std
## 1 11000   NA    NA 10.91051 12.6669 4.977147 2.013346
```

```
# Let's keep adding
for (i in 1:5) {
  values <- rnorm(2000,mean=5,sd=2)
  stats  <- streaming_stats(stats,values)
  show(stats)
}
```

```
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
```

```
## 1 13000   NA    NA 11.07848 12.83544 4.981696 2.012732
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
## 1 15000   NA    NA 11.22138 12.97753 4.980706 2.007914
##       N sumX sumX2  lnSumX  lnSumX2       mu      std
## 1 17000   NA    NA 11.3477 13.10562 4.986461 2.014688
##       N sumX sumX2   lnSumX lnSumX2       mu      std
## 1 19000   NA    NA 11.45875 13.2167 4.985606 2.015715
##       N sumX sumX2   lnSumX lnSumX2       mu      std
## 1 21000   NA    NA 11.55981 13.3182 4.990503 2.013773
```

```
# Add a small shift to the mean and std.dev.
for (i in 1:5) {
  values <- rnorm(2000,mean=7,sd=3)
  stats  <- streaming_stats(stats,values)
  show(stats)
}
```

```
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
## 1 23000   NA    NA 11.68653 13.49564 5.172109 2.196658
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
## 1 25000   NA    NA 11.79852 13.64683 5.322201 2.337817
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
## 1 27000   NA    NA 11.89887 13.77652 5.448169 2.436866
##       N sumX sumX2   lnSumX  lnSumX2       mu      std
## 1 29000   NA    NA 11.99022 13.89231 5.557629 2.519574
##       N sumX sumX2   lnSumX lnSumX2       mu      std
## 1 31000   NA    NA 12.07374 13.9949 5.651917 2.579247
```

```
# Compare with normal calculations
set.seed(42)
values <- c(rnorm(21000,mean=5,sd=2),rnorm(10000,mean=7,sd=3))
mean(values)
```

```
## [1] 5.649884
```

```
sd(values)
```

```
## [1] 2.579
```

### Here is a another technique – From Knuth's art of computer programming

http://www.johndcook.com/blog/standard_deviation/