Random Walks

October 16, 2014

Convergence of Binomial Probabilities[1]

We all intuitively understand that when a fair coin (50% heads, 50% tails) is flipped many times. the probability of heads versus tails should start to converge. For example, let's say we get 1 point for every head and 0 points for every tail. What will the *average* number of points we get at any point in time? Let's compute and plot this in R with `n=5000` coin flips!

[1] These examples are based on: `http://nvenkataraman1.github.io/random-walks/`

```
n <- 5000
flips <- rbinom(n, 1, 0.5)  # a fair coin follows a simple binomial distribution
means <- c(0)
for (i in 1:n) {
    means[i] <- mean(flips[1:i])
}

plot(means, type = "l", ylim = c(0, 1))
abline(h = 0.5, col = "red")
```

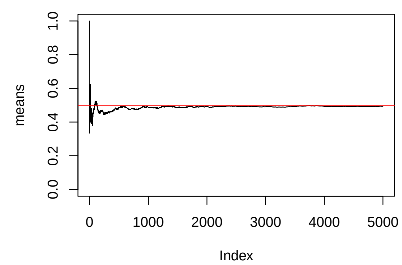

Figure 1: Coin flipping probabilities

We see that the average number of points quickly converges to 0.5 (a *half*), just as we suspected. This is our common intuition about how probabilities average out over time.

Now, instead of average probabilities, let's make a game where we gain 1 point for heads and lose one point for tails. How many *total points* would we make over time? Will it converge?

```
n <- 5000
flip <- 2 * rbinom(n, 1, 0.5) - 1
total <- c(0)
for (i in 2:n) {
    total[i] <- total[i - 1] + flip[i]
}

plot(total, type = "l", ylim = c(-300, 300))
abline(h = 0)
```

Figure 2: A random (binomial) walk

When we plot out our total number of points, rather than converging, it seems to just randomly walk away! *Random walks* are one way of modeling many everyday phenomena that are affected by so many variables that they appear random in their progress: stock markets, weather fluctuations, ant foraging, and more[2].

[2] `http://en.wikipedia.org/wiki/Random_walk`

But how far from their starting points do random walks go *on average*? It isn't easy to tell, so let's simulate random walks and find out!

```
# Random binomial walk function
random_binomial_walk <- function(steps) {
    flip <- 2 * rbinom(n, 1, 0.5) - 1
    total <- c(0)
    for (i in 2:steps) {
        total[i] <- total[i - 1] + flip[i]
    }
    return(total)
}

# Simulate six random walks of 5000 steps each
walks <- data.frame(matrix(0, nrow = 5000, ncol = 6))
for (i in 1:6) {
    walks[i] <- random_binomial_walk(n)
}

# Plot out our six random walks
par(mfrow = c(2, 3))  # setup plotting to 2rows x 3 columns
for (i in 1:6) {
    plot(walks[[i]], type = "l", ylim = c(-200,
        200))
    abline(h = 0, col = "red")
    abline(h = 0, col = "red")
}
```

Even with a small number of random walks, we see they can go anywhere. It will be hard to plot a large number of walks. So let's simulate a large number of walks and see how their final distance from zero is distributed.

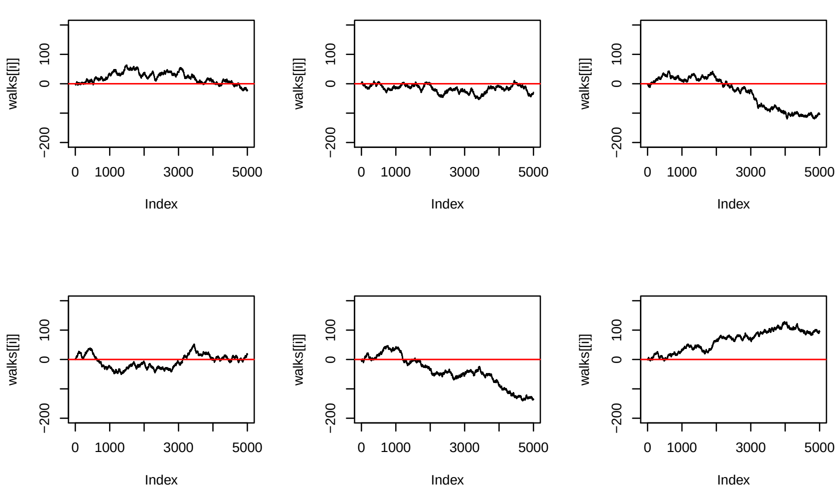

Figure 3: Six Binomial Walks

```
par(mfrow = c(1, 1))  # return graphics settings to 1 x 1
```

Even with a small number of random walks, we see they can go anywhere. It will be hard to plot a large number of walks. So let's simulate a large number of walks and see how their final distance from zero is distributed.

```
m <- 3000
n <- 2000
walks <- data.frame(matrix(0, nrow = n, ncol = m))
for (i in 1:m) {
    walks[i] = random_binomial_walk(n)
}
distance <- c(0)
for (i in 1:m) {
    distance[i] <- walks[n, i]
}

hist(distance, breaks = 20, prob = TRUE)
lines(density(distance), lwd = 3, col = "blue")
```

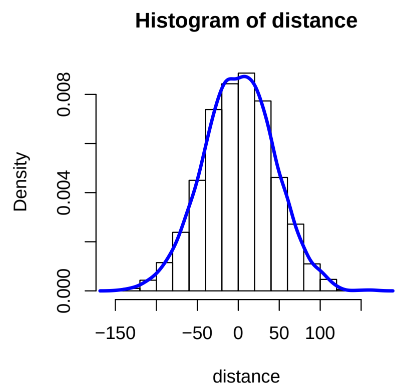

Figure 4: Distribution of binomial walks

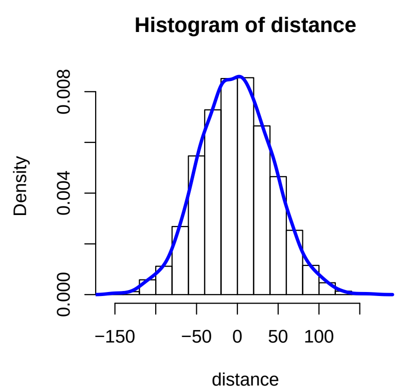

Figure 5: Distribution of gaussian walks