Econ B2000
Michael Stewart

Homework 2

Group: Michael Stewart & Nasrin Khanam

Lab 1 Results

We decided to roll our die 60 times. With six sides, this means any given number should be rolled ten times on a "fair" die.

With this in mind, we chose 20 as the number of sixes for our criteria for "unfair". The likelihood of rolling 20+ sixes on a fair die (i.e., the likelihood that a fair die would be judged "unfair" under this criteria) was remote, a less than 1% chance (0.001232336).

Additionally, since the number one is opposite six on a standard die, we decided five or fewer ones would also be considered "unfair". This created a larger likelihood of a fair die being judged "unfair", just over 5% (0.05120829).

Further, we looked at the likelihood of a fair die rolled 60x would result in 20+ sixes AND 5 or fewer ones and found this to be a very remote possibility, well under 1% (0.0002120251).

Lastly, we decided to look at the cross section of these unfairness criteria, and calculated the likelihood a fair dice rolled 60x would result in EITHER 20+ sixes OR 5 or fewer ones. This was slightly higher than the likelihood with the fewer than five ones criteria alone, at 5% (0.0522286).

It was interesting that the results were so different between doubling or halving the expected result. The doubling meant the likelihood a fair dice would be judge unfair to under 1%, which the halving meant a 5% chance a fair dice would be judged unfair. Perhaps this is more a question of number of rolls than ratio of results, because the increase was of 10 rolls whereas the decrease was of only 5 rolls.

Playlist Experiment

We decided to create a playlist of 30 songs and pres 'next' 60 times. On average this should've meant each song would come up twice, and so we decided to consider our criteria for considering the playlist potentially weighted at 4+ instances of a particular song.

Hypothesis: songs will play an average of 2x

However, when running the experiment, each song appeared exactly 2x. This to me suggests the algorithm protects against repeats by including something along the lines of "replace = FALSE" somewhere in the code. Even though it meets our hypothesis, it suggests some manipulation since there were no instance of a song being played even one extra time or one time fewer. Results perfectly matching the probability are very unlikely.

To see the likelihood that this would occur if each draw was truly random, I performed the following:

> # number of songs

```
> n_songs <- 30
>
> # number of plays
> n_plays <- 60
>
> # probability that each song appears exactly 2 times
> # if each play is an independent random draw
> prob <- factorial(n_plays) / ((factorial(2)^n_songs) * (n_songs^n_plays))
>
> format(prob, scientific = TRUE)
[1] "1.828099e-16"
```

With this in mind, I decided to run a simulation of a playlist that does a true instance-buy-instance shuffle. This led to 10 songs being played exactly twice (out of 30). Meanwhile five were played 4x and three were never played.

```
# Load CSV
songs_df <- read.csv("HW 2 Playlist.csv", stringsAsFactors = FALSE)

# Load unique 30 songs
songs <- unique(songs_df$Song)

# Sanity check
length(songs)  # should be 30

# Number of plays
n_plays <- 60

# Random draw WITH replacement
set.seed(123)  # optional, for reproducibility
plays <- sample(songs, size = n_plays, replace = TRUE)

# Count including songs that got 0 plays
counts <- table(factor(plays, levels = songs))

# Show results in descending order
sort(counts, decreasing = TRUE)
```

```
> # Create a data frame with song names and counts
> counts_df <- data.frame(
+     Song = names(counts),
+     Count = as.integer(counts)
+ )
>
> # Write to CSV
> write.csv(counts_df, "simulated_playlist_counts.csv", row.names = FALSE)
```

Resulting CSV