Spicy Burger Roulette

Sean Soutar

10 December 2017

Background

Sliders are a great way to sample the tastes of many burgers. They typically come in 3's and are miniature versions of the full meal.

I was at a popular burger restaurant with two friends a while ago. On the menu, there was an option to play "Burger Roulette" when you ordered sliders. How it works is that if you order a plate of sliders (3 burgers on a plate), one of them will be loaded with very hot chilies. As a result, the three of us each ordered a different plate of sliders where one burger on each plate was loaded with very hot chilies.

The plan was that we would swap burgers between ourselves randomly as we wanted to taste from each others' selection. I thought to myself, what are the chances that I get all three spicy burgers on my plate?

Assumptions

- I presumed that you have absolutely no idea as to which burger contains the chilies. We would be naively swapping burgers.
- There is a strict order of swapping. Player 1 and 2 will exchange burgers, then 2 and 3. Finally, 3 and 1 will swap.

Script Logic

I used the `tidyverse` and `doParallel` packages in conducting the simulation. I used `ggalt` for added visualisation features on top of `ggplot2`. I start by setting up a parallel cluster and creating the burger swapping index function.

```
swapping_positions <- function(number,min,max){
  runif(n = number,min,max) %>% round(digits = 0)
}
```

A uniform random variable is created with a lower and upper bound of min and max. This number is then rounded to an integer. This will be the burger index that player A will choose from player B.

The script will use the `swapping_positions` function to exchange boolean elements between players for a given number of games at a round size (r). An element is `TRUE` when it is a spicy burger. The round size refers to how many swapping rounds there will be per game. If r = 3 , the process as described in the second assumption will be performed 3 times per game.

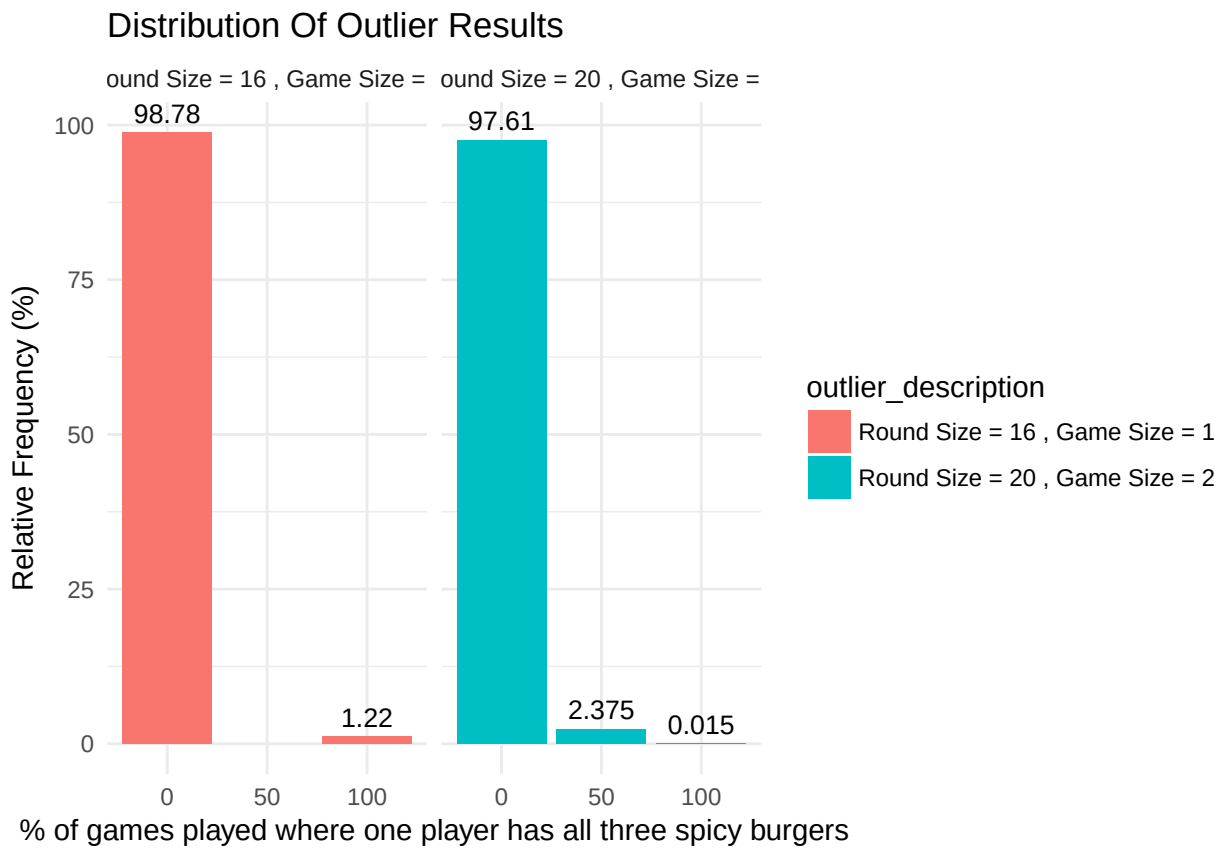
The script used is available as `spicy_roulette.R`.

Results

The simulation was conducted for round sizes up to 20 with each round size being played for a total of 1 to 1000 games. The results are plotted

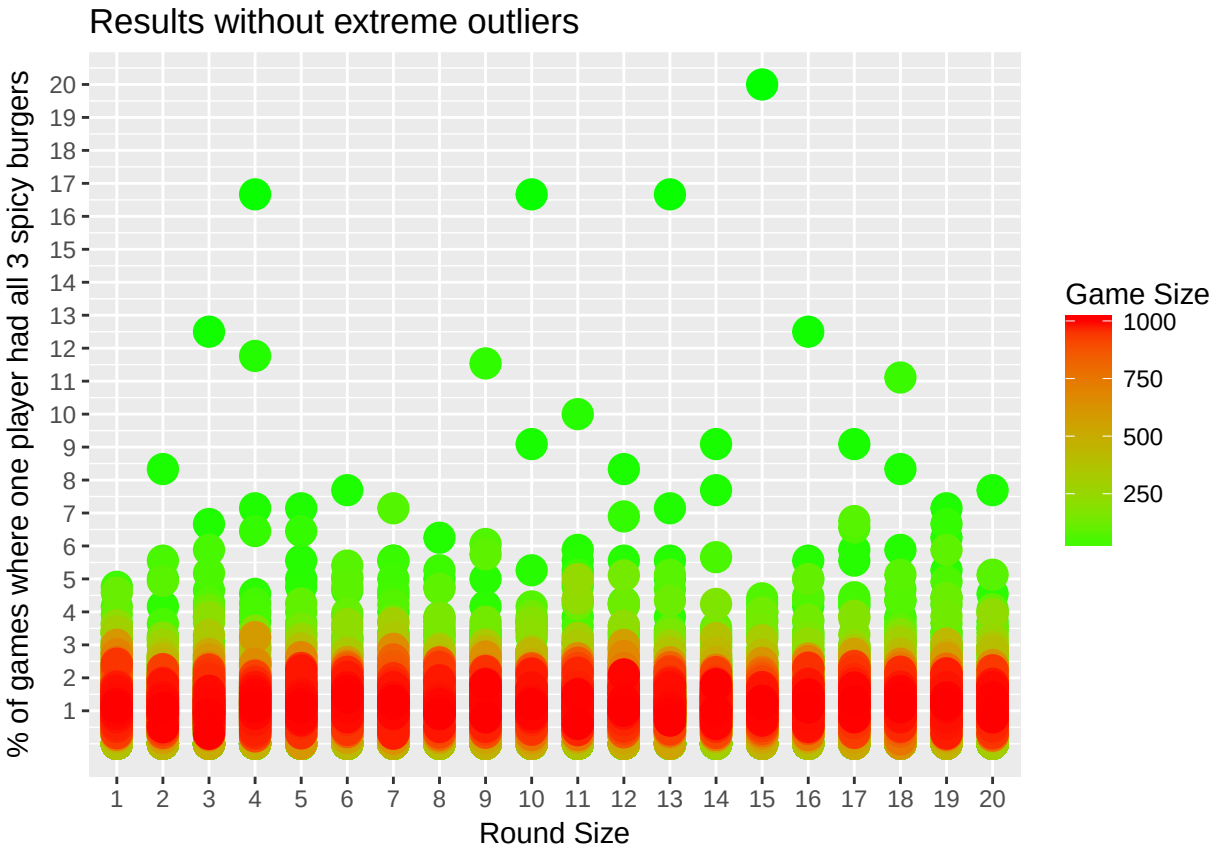
Figure 1:

What immediately stands out are the two outliers at round size 16 and 20. These round sizes were played for 1 and 2 games respectively. The script seemed to have produced pretty consistent results (see the plot below) except for these two points. I decided to simulate a round size of 16 and 20 for 1 and 2 games respectively in order to figure out how unusual those points really are. I simulated the aforementioned burger roulette descriptions 20000 times each. I generated the simulated distributions below.

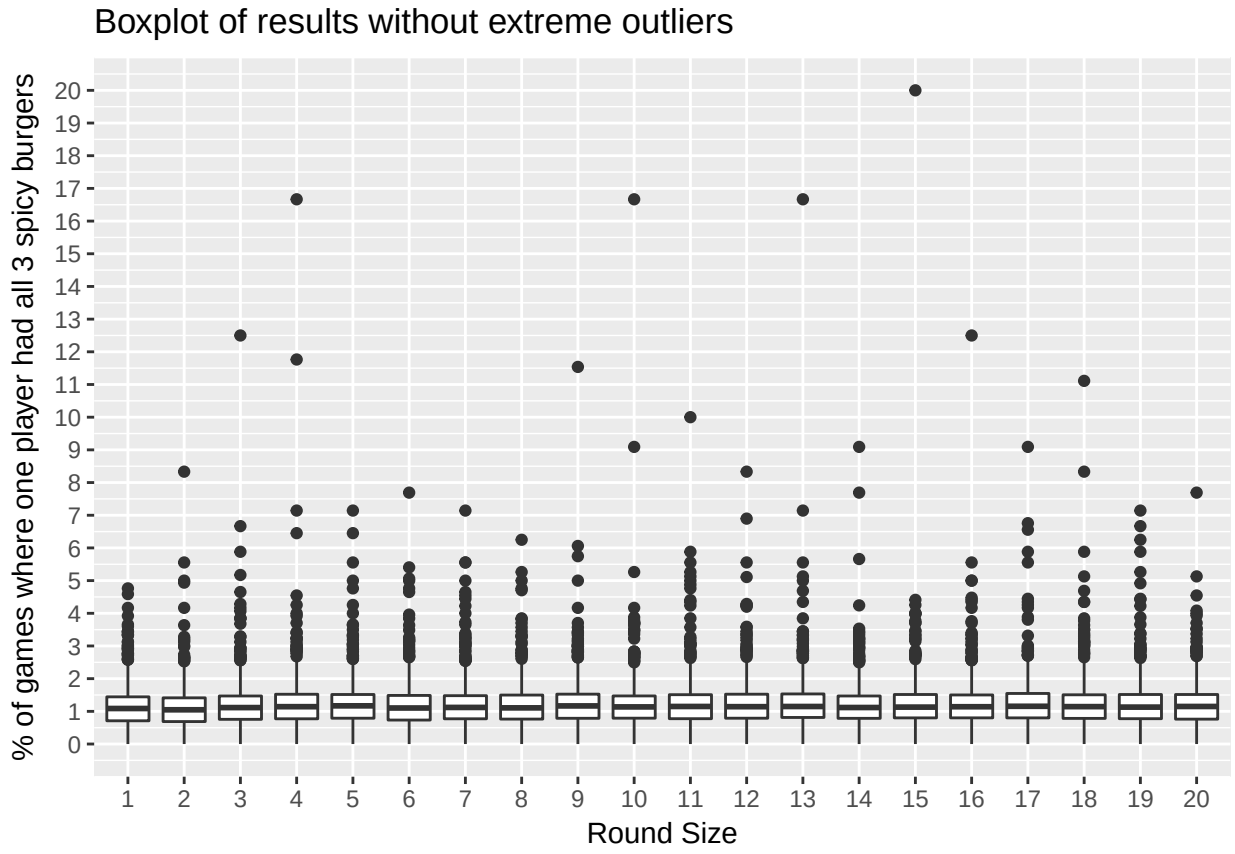


The chances of observing the outlying point at round size of 16 is approximately 1.22% and the other outlier chance is 2.375%. Although these are all small probabilities, observing these points through the main simulation is not unusual enough to bring the simulating procedure into question.

Regardless, those points are clearly extreme outliers. They were excluded from the plot to produce the second graph below.



The colour scale according to game size proves to be quite useful in this plot. You can clearly see that there is much more variation in the results under low game sizes in green (<250) compared to large game sizes in red (>750). Under large game sizes, the percentage chance of success hovers around the same band of between 0.8% and 1.3%. Whilst this plot illustrates how the variance reduces with increased game sizes, it doesn't show the density of the results well. This is why I decided to plot a box plot for each round size in the graph below.



It can be clearly seen that 50% of results for all game sizes for a given round number occur approximately in the small band of 0.8% and 1.3%.

I attribute the small success rate due to the fact that you are naively swapping burgers. As a result, it is quite likely that you undo your progress (if you made any) with each success round. Under my Assumptions, I stated a strict swapping order between players. I believe that if I were to make that random, the chances of success would be even lower as it would add another stochastic element that the players aren't mitigating by their naive swapping.

So what is the long run probability of you sitting down and winning the burger roulette? I ordered the results in ascending order and constructed a 95% confidence interval for the percentage chance of a player winning the burger roulette.

```
ordered_results <- sort(simulation_results$all_spicy_chance)
lower_bound <- ordered_results[length(ordered_results)*0.025]
upper_bound <- ordered_results[length(ordered_results)*0.975]
```

```
## The 95% confidence interval for the true success rate is:[ 0 ; 2.671756 ]
```

Conclusion

What we can say is that if you want to have exciting games with players "winning" the roulette, the round size is not that important , but you need to have very low game sizes (<20). This maximises the chances of getting a few games where a player has all three spicy burgers. If spicy chilies are your kryptonite, you need not fear - you are very likely not going to end up with all three spicy burgers.

What I learnt

- Running simulations in parallel yields HUGE performance boosts. One just has to think through whether or not the problem is well suited to parallelisation.
- Outliers should always be examined. I am glad that I looked at my two outlying points. It is important to figure out whether or not outliers are because of measurement errors , pure chance or a faulty algorithm.
- I need to Git-good. This is the first mini-project that I used with Git. I use the Gitkraken client and it optimised my workflow nicely.