My M&M OCD

Yoni

05 04, 2025

Intro

The goal of this simulation is to test the statistics of M&M and other stacks even Chocolate lentils by color, I wanted to know, if I eat m&m package 2 by 2, separated by color, what is the chance of my finishing the package without mixing any color in one bite.

In addition, here are some BI incite that needed to be checked:

1. What is the probability of M&M packages packaged fairly?
2. What is the probability of M&M packages packaged without one color?
3. How does the size of the package or number of colors affect this probability?

The method is based of simulation of some M&M bags, according to the most common sizes. Each time we sample x lentils, name them by colors (represented as factorial numbers), and see the results for many packages as a statistic data.

Parameters

The basic parameters (will be changed later):

```
#parameters
nn<- 500            #numbers of bags per sample
n_color<- 6         #unique colors of M&M
gram<- 0.91         #weight of one M&M
bag_g<- 250         #common weight of M&M package
n_unit<- bag_g/gram #M&M per package
av_per_color= n_unit/n_color
paste0("The avarage number of lentils per color is ", round(av_per_color,2))
```

```
## [1] "The avarage number of lentils per color is 45.79"
```

Creating of Sample

General Sample

create_bag is a function to create one snack package as matrix.

sample_MnM is a function to create n bags from the create_bag function.

```
## [1] "One bag:"

##       1  2  3  4  5  6
## [1,] 18 19 13 18 14 18

## [1] "3 bags:"

##       1 2 3 4 5 6
## Bag_1 3 1 0 3 2 2
## Bag_2 0 3 3 1 3 0
## Bag_3 1 1 2 3 2 2
```

Preview Graph

Now will be creating nn bugs of M&M

columns:

1. V1:V6- the number of lentils per color
2. even_count- how many evens colors there are
3. even_evens- are the uneven colors even
4. Variance- variance of lentils per color
5. low_col-
6. min- the lowest color in each row

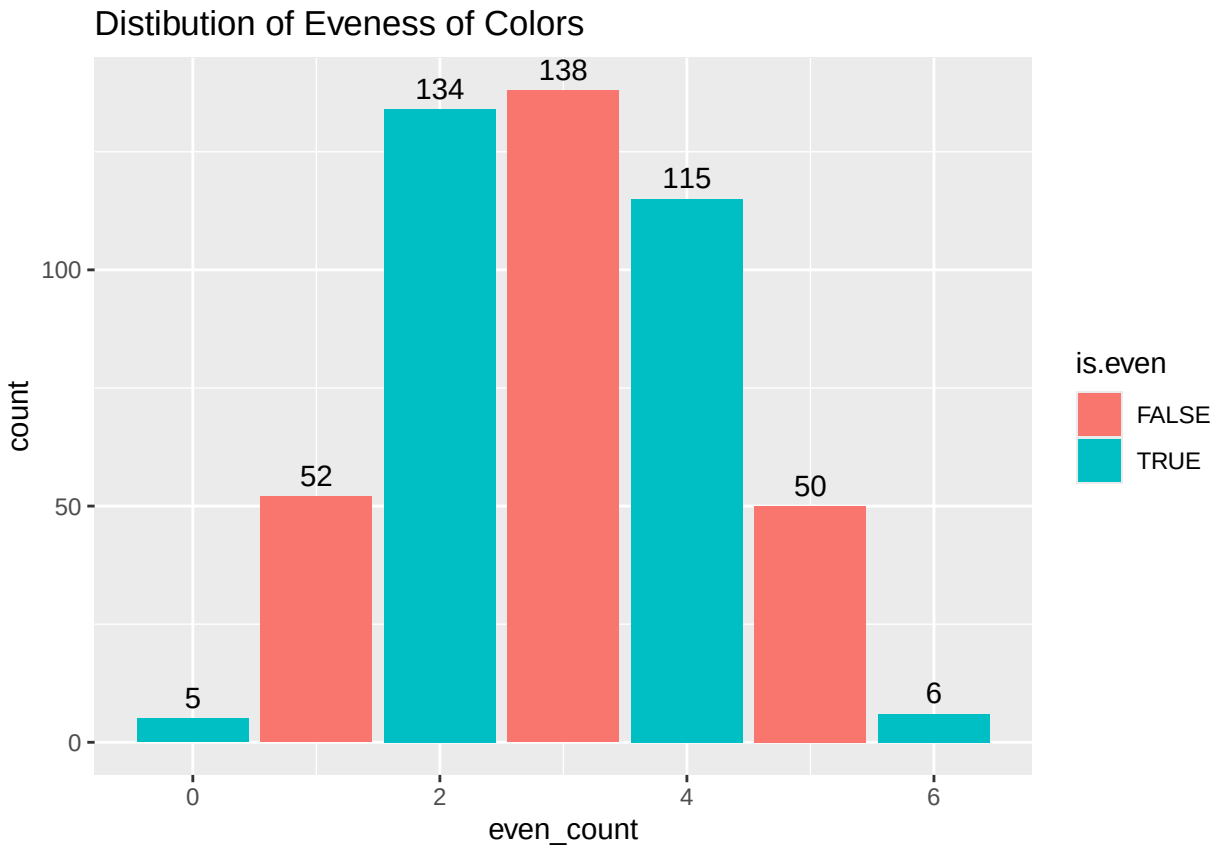
here are the first rows:

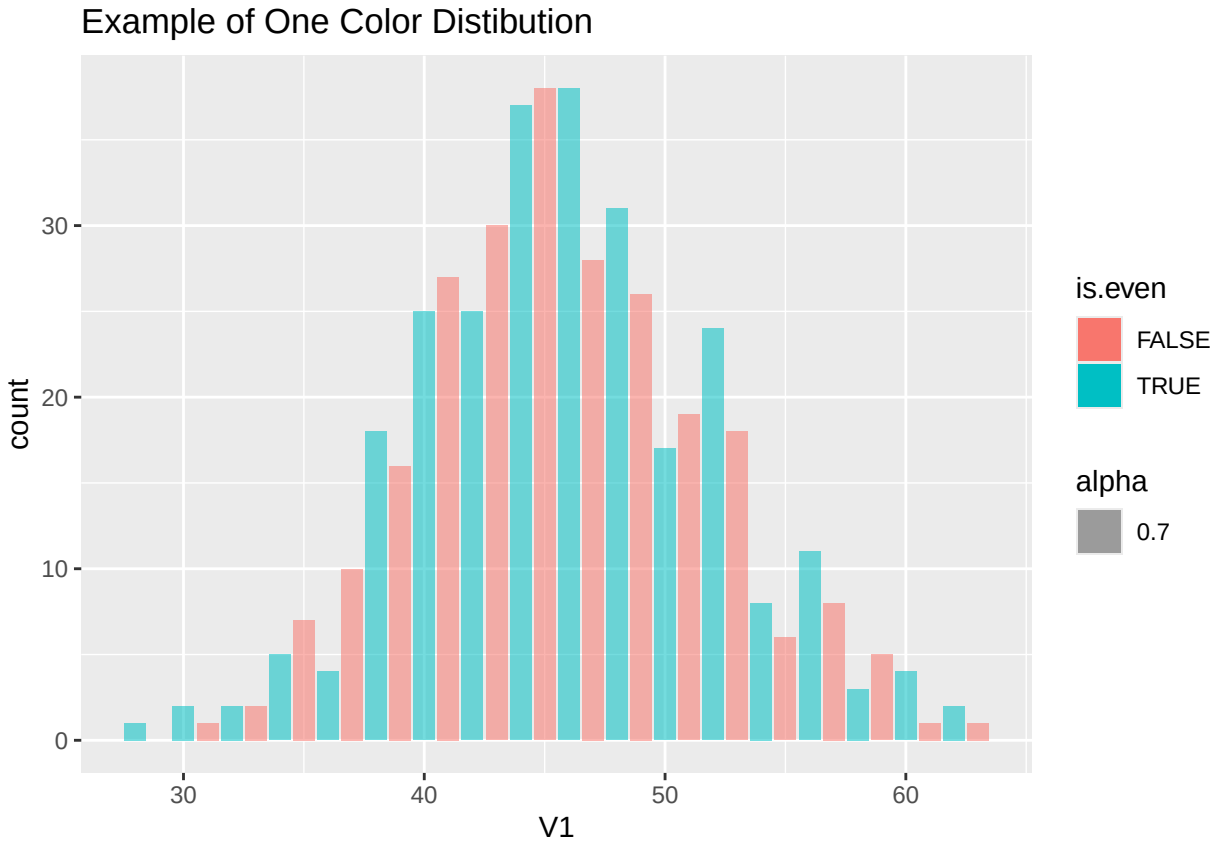
```
## # A tibble: 6 x 11
##      V1    V2    V3    V4    V5    V6 even_count even_evens Variance low_col
##   <int> <int> <int> <int> <int> <int>      <dbl> <lgl>         <dbl>   <dbl>
## 1    48    47    42    46    45    46          4 TRUE           4.27       0
## 2    55    36    38    46    49    50          4 TRUE          53.9        0
## 3    42    48    48    45    44    48          5 FALSE          6.57       0
## 4    47    46    45    53    43    41          1 FALSE         17.0        0
## 5    48    50    42    50    39    46          5 FALSE         20.2        0
## 6    52    49    44    35    54    40          4 TRUE          53.9        0
## # i 1 more variable: min <int>
```

plot the MnM sample sample

```
## [1] "summary of all colors Distibution:"

##    Min. 1st Qu. Median   Mean 3rd Qu. Max.      Var
## V1   28      42   45.5 45.818      50   63 35.97282
## V2   29      42   46.0 45.664      49   67 35.03718
## V3   28      42   46.0 45.882      50   64 37.08625
## V4   27      41   45.0 45.702      50   66 40.09739
## V5   29      41   45.0 45.702      50   66 38.63847
## V6   27      41   46.0 45.712      50   67 41.17941
```


Distibution of Eveness of Colors
count
100
50
0
5
52
134
138
115
50
6
0
2
4
6
even_count
is.even
FALSE
TRUE



Test Expected Value

to see is the mu of the lentils per color are fair, we will test it per column with t.test for each color.

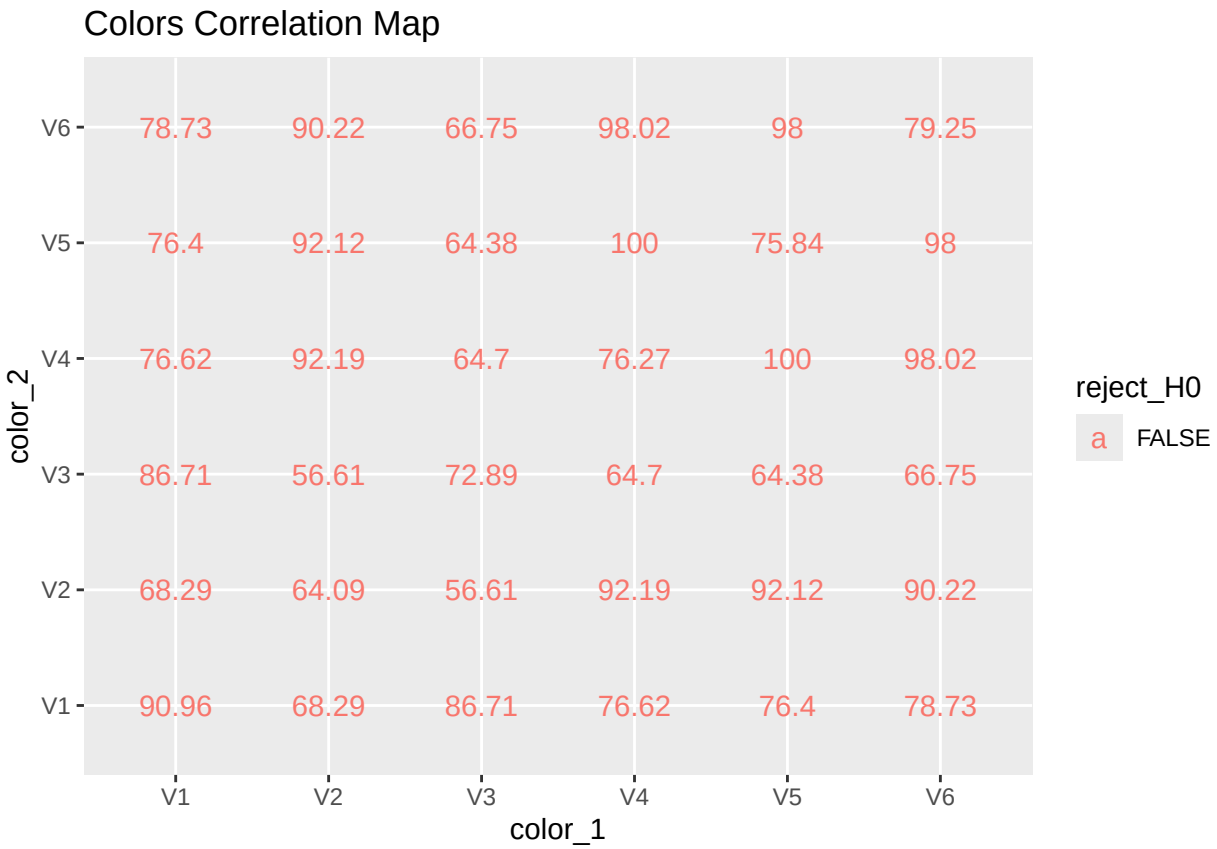
Here is the result, none of them bellow 5% P. value

```
##       V1       V2       V3       V4       V5       V6
## "90.96%" "64.09%" "72.89%" "76.27%" "75.84%" "79.25%"
```

Now we will do the same checking for 2 samples, to see whether there is correlation between each 2 colors distribution.

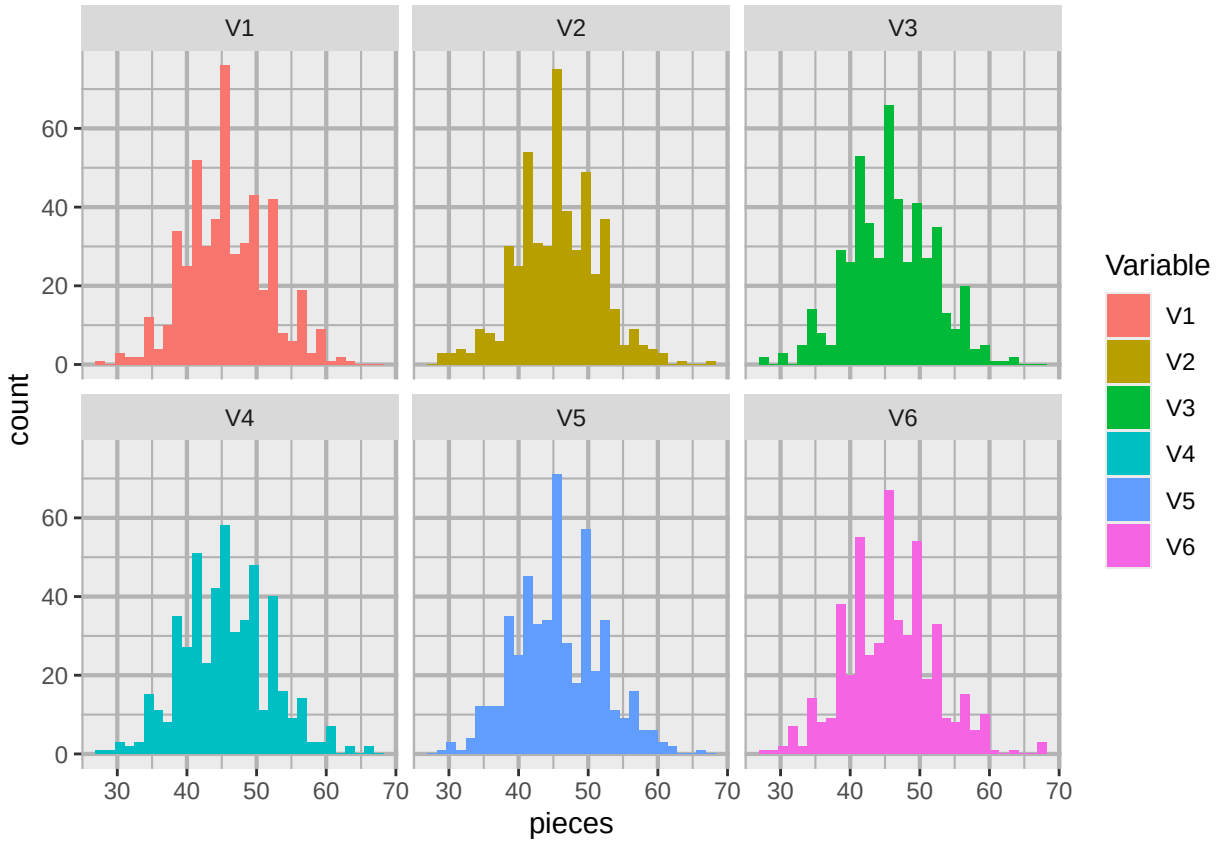
for each row i and column j, 1) if i==j, this it the check from before of the expected value to n_unit/n_color 2) if i!=j, this is two samples test of same expected value hypothesis

```
##         V1     V2     V3     V4     V5     V6
## V1 0.9096 0.6829 0.8671 0.7662 0.7640 0.7873
## V2 0.6829 0.6409 0.5661 0.9219 0.9212 0.9022
## V3 0.8671 0.5661 0.7289 0.6470 0.6438 0.6675
## V4 0.7662 0.9219 0.6470 0.7627 1.0000 0.9802
## V5 0.7640 0.9212 0.6438 1.0000 0.7584 0.9800
## V6 0.7873 0.9022 0.6675 0.9802 0.9800 0.7925
```



now here Is visualization of the actual data per color:

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



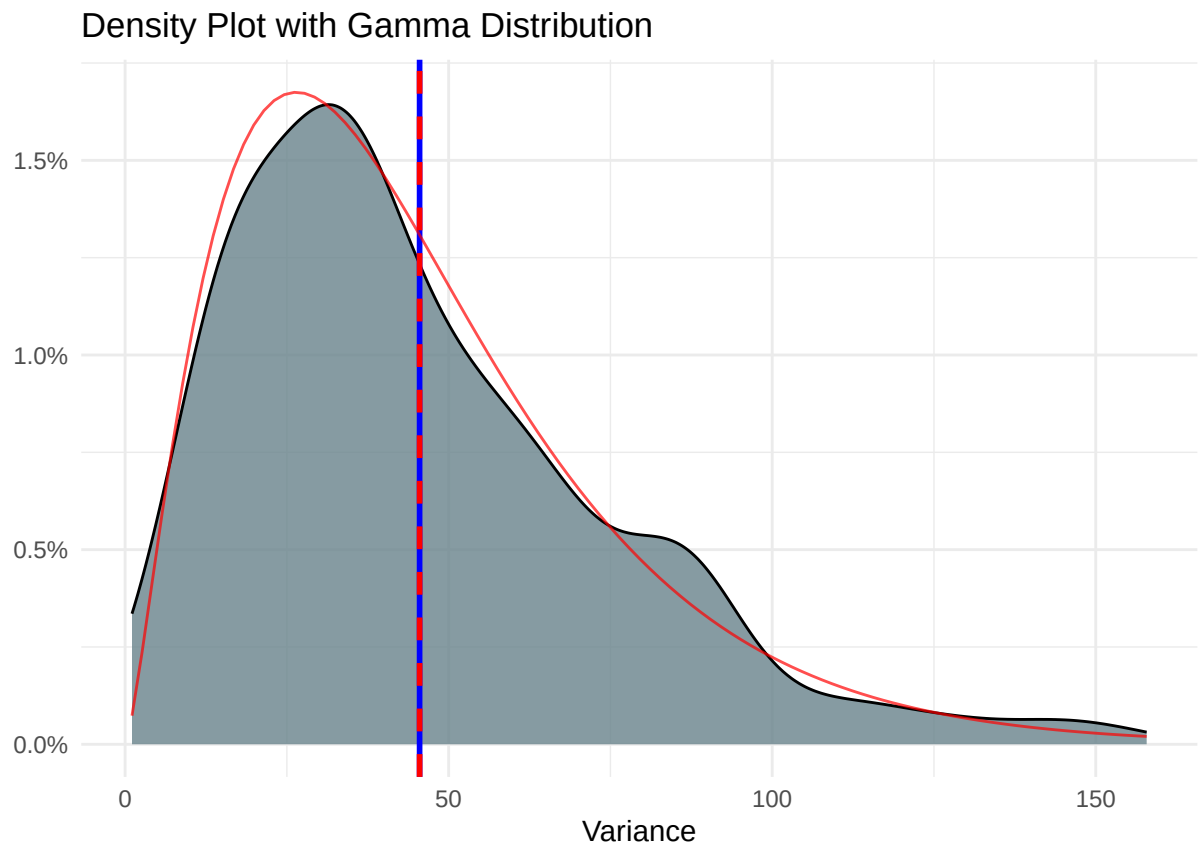
Variance Distribution Checking

We know that the distribution of variance is approximately Gamma distribution:

$$f(x) = \frac{1}{(\Gamma(\alpha)\theta^{\alpha})} x^{\alpha-1} e^{-x/\theta}$$

We can see that the variance distribution is Gamma like with shape and rate as seen below

```
## [1] "The parameters of the gamma shaped variance is shape 2.388 and rate 0.052"
```



#use statistics to sample better low chance cases

n*m types of snacks

We will create a function that create sample for each number of colors and package size we want, and then calculate some interesting parameters

```
mega_snack<- function(nn,n_unit,n_color)
  {
  m_sample<- length(n_unit)*length(n_color)
  nul_mat= matrix(nrow = m_sample, ncol = 6)
  res<- cbind(rep(n_unit,length(n_color)),sort(rep(n_color,length(n_unit))),
                                              nul_mat)
  for (i in 1:(dim(res)[1]))
  {
    #print(c(res[i,1],res[i,2]))
    low_color<- 0.666*res[i,1]/(res[i,2])
    small_sample<- sample_MnM(nn,res[i,1],res[i,2])
    small_sample<-
      small_sample %>% as_data_frame() %>%
      mutate(even_count= rowSums(across(everything()   , ~ .x %% 2 == 0))/n_color,
             #how many evens colors there are
             even_evens= (rowSums(across(c(1:n_color)   , ~ .x %% 2 == 1)) %% 2 ==0)/n_color,
             #are the uneven colors even
             var_col=    apply(across(c(1:n_color)), 1, var),
```

```
            #var of candy per color/ type
            min=        apply(across(c(1:n_color)), 1, min),
            #lowest value in color
            all_even=   rowSums(across(c(1:n_color)  , ~ .x %% 2 == 0))== n_color,
            low_col=    rowSums(across(c(1:n_color), ~ .x <= low_color ))>=1
           )
   res[i,3]<- mean(small_sample$even_count)
   res[i,4]<- mean(small_sample$even_evens)
   res[i,5]<- mean(small_sample$var_col)
   res[i,6]<- mean(small_sample$all_even)
   res[i,7]<- mean(small_sample$low_col)
   res[i,8]<- min(small_sample$min)
     }
 colnames(res)<- c("n_unit", "n_color", "even_count", "even_evens",
                   "var_col","all_even","low_color","smallest_col")
 res %>% as.data.frame()
 }
```

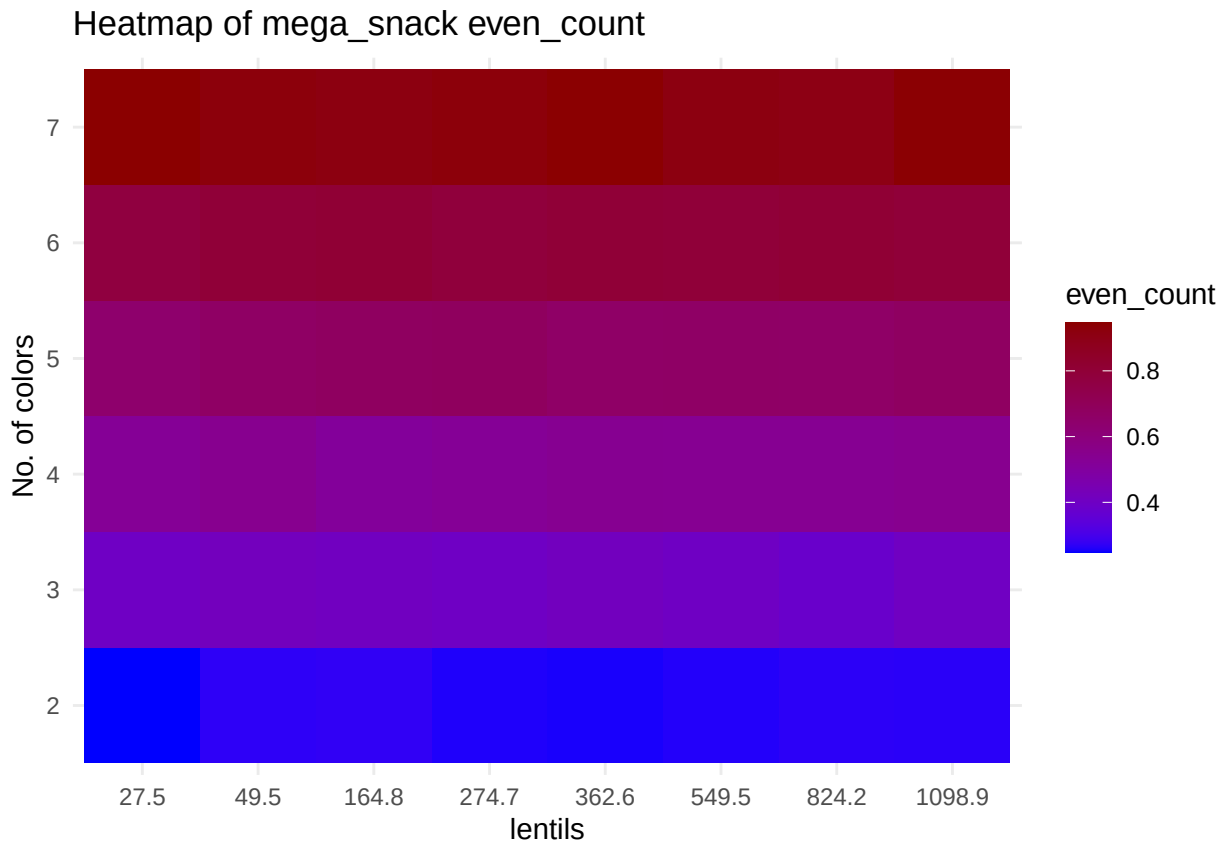
```
color_op<- 2:7
grams_op<- c(25,45,150,250,330,500,750,1000)
n_unit_op<- grams_op/gram
nn=700
```

We will make the multiple sample. Here is some random rows:

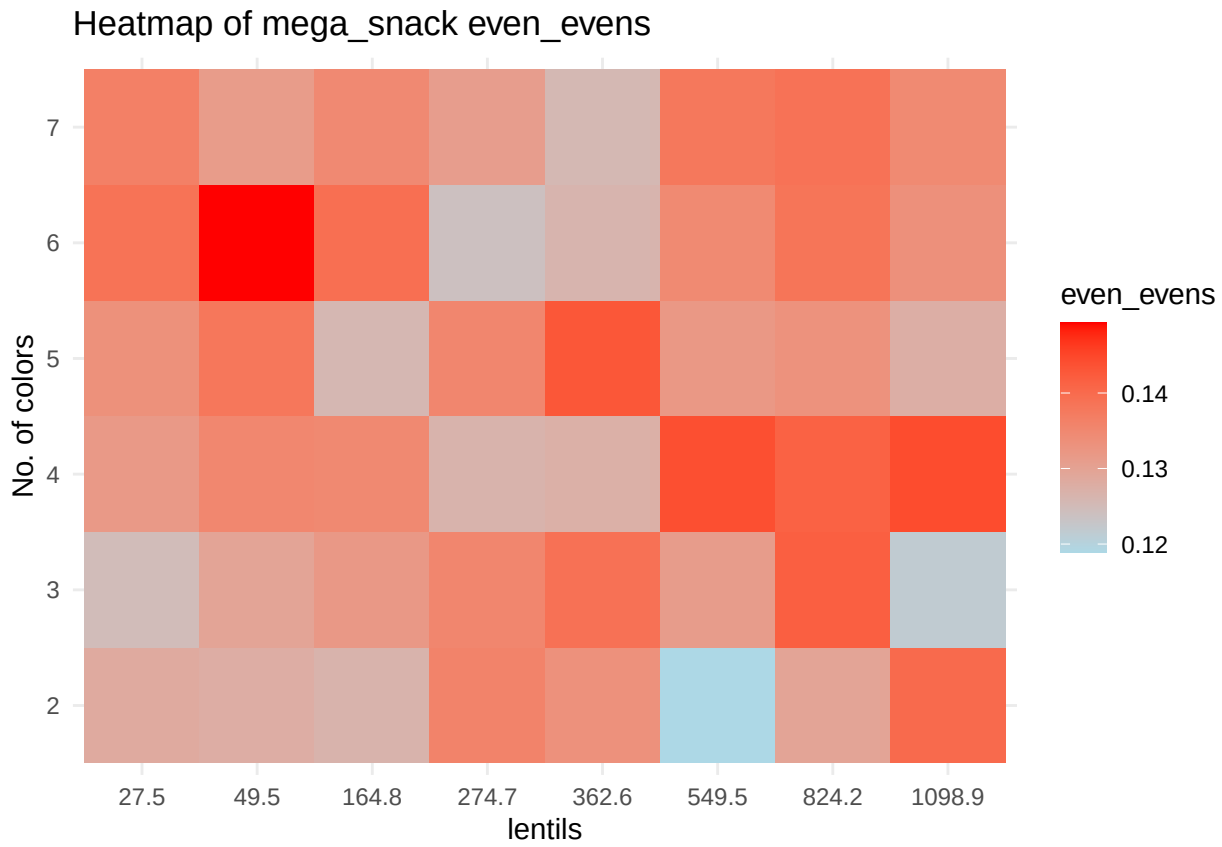
```
##    n_unit n_color even_count smallest_col    var_col
## 1    27.5       2  0.2482959            5  16.023571
## 2    49.5       2  0.2697959           13  23.111429
## 3   164.8       2  0.2728299           60  78.135714
## 4   274.7       2  0.2592075          111 141.951429
## 5   362.6       2  0.2558605          148 181.262143
## 6   549.5       2  0.2611973          239 273.902857
## 7   824.2       2  0.2678844          365 437.093571
## 8  1098.9       2  0.2662177          495 546.175000
## 9    27.5       3  0.4015306            2   8.802143
## 10   49.5       3  0.4182517            6  17.187143
## 11  164.8       3  0.4108503           35  55.534286
## 12  274.7       3  0.4008810           64  88.762143
## 13  362.6       3  0.4155374           95 120.732143
## 14  549.5       3  0.4032823          145 195.965714
## 15  824.2       3  0.3814694          240 270.817857
## 16 1098.9       3  0.4058367          309 394.915000
## 17   27.5       4  0.5264490            1   7.057857
## 18   49.5       4  0.5446939            5  12.705000
## 19  164.8       4  0.5144354           24  43.023571
## 20  274.7       4  0.5257517           44  72.627857
## 21  362.6       4  0.5382381           63  91.072143
## 22  549.5       4  0.5362823          100 141.614286
## 23  824.2       4  0.5349490          159 212.302857
## 24 1098.9       4  0.5452687          229 272.572857
## 25   27.5       5  0.6426190            0   5.405000
## 26   49.5       5  0.6660442            2   8.992143
## 27  164.8       5  0.6778265           17  32.012857
```

```
## 28  274.7       5  0.6862687           33  54.594286
## 29  362.6       5  0.6626667           47  70.809286
## 30  549.5       5  0.6646599           78 102.930000
## 31  824.2       5  0.6605714          130 165.097857
## 32 1098.9       5  0.6761429          175 224.266429
## 33   27.5       6  0.7737687            0   3.931429
## 34   49.5       6  0.7999796            1   8.316429
## 35  164.8       6  0.8078367           12  27.030714
## 36  274.7       6  0.7883401           28  41.461429
## 37  362.6       6  0.8013095           42  57.883571
## 38  549.5       6  0.7953707           65  93.350000
## 39  824.2       6  0.8081395          107 134.487143
## 40 1098.9       6  0.7971769          141 199.497143
## 41   27.5       7  0.9468571            0   3.980000
## 42   49.5       7  0.9275408            0   6.140000
## 43  164.8       7  0.9153673           11  23.394286
## 44  274.7       7  0.9276531           20  40.940714
## 45  362.6       7  0.9441599           30  55.522143
## 46  549.5       7  0.9165374           48  83.489286
## 47  824.2       7  0.9045408           87 120.330000
## 48 1098.9       7  0.9417551          114 173.919286
```

```r
mega_snack_1 %>%
  ggplot(aes(x = factor((round( n_unit,1) )), y = factor(n_color ), fill = even_count )) +
  geom_tile() +
  scale_fill_gradient(low = "blue", high = "red4")+
  labs(title = "Heatmap of mega_snack even_count",
       x = "lentils",
       y = "No. of colors",
       fill = "even_count") +
  theme_minimal()
```

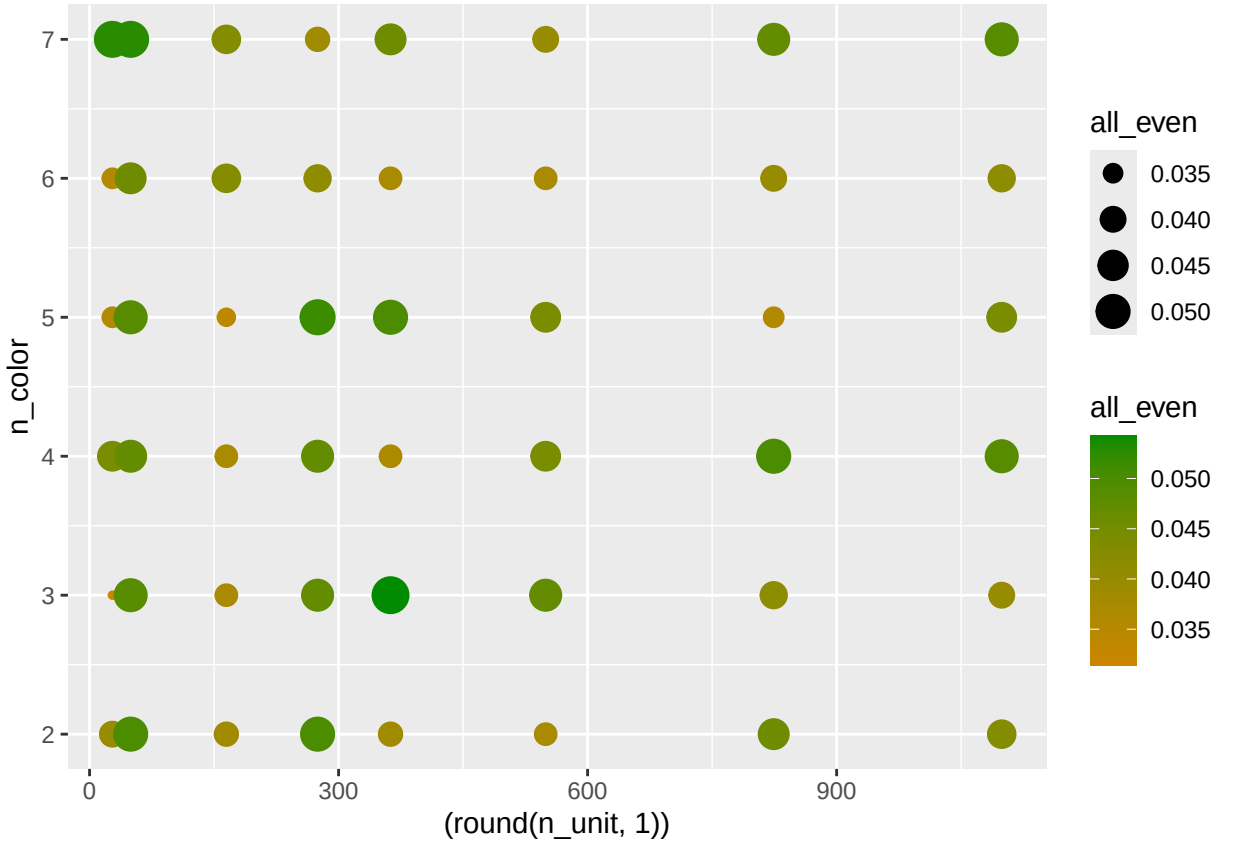



```
mega_snack_1 %>%
  ggplot(aes(x = factor((round( n_unit,1) )), y = factor(n_color ), fill = even_evens )) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "red")+
  labs(title = "Heatmap of mega_snack even_evens",
       x = "lentils",
       y = "No. of colors",
       fill = "even_evens") +
  theme_minimal()
```



now let us see the probability of all even, and whether there is pattern.

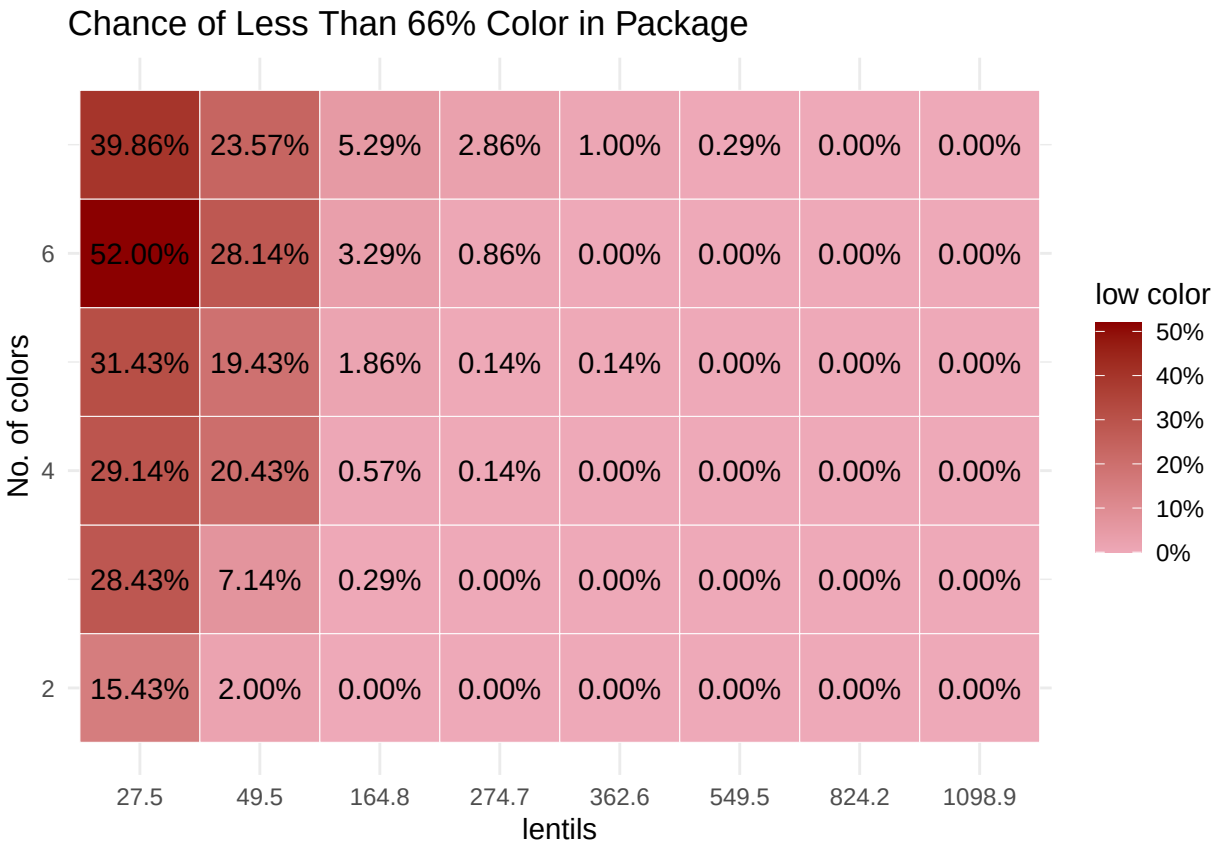
```
mega_snack_1 %>%
  ggplot(aes(x = (round( n_unit,1) ), y = n_color , color = all_even,size = all_even )) +
  geom_point() +
  scale_color_gradient(low = "orange3", high = "green4")
```

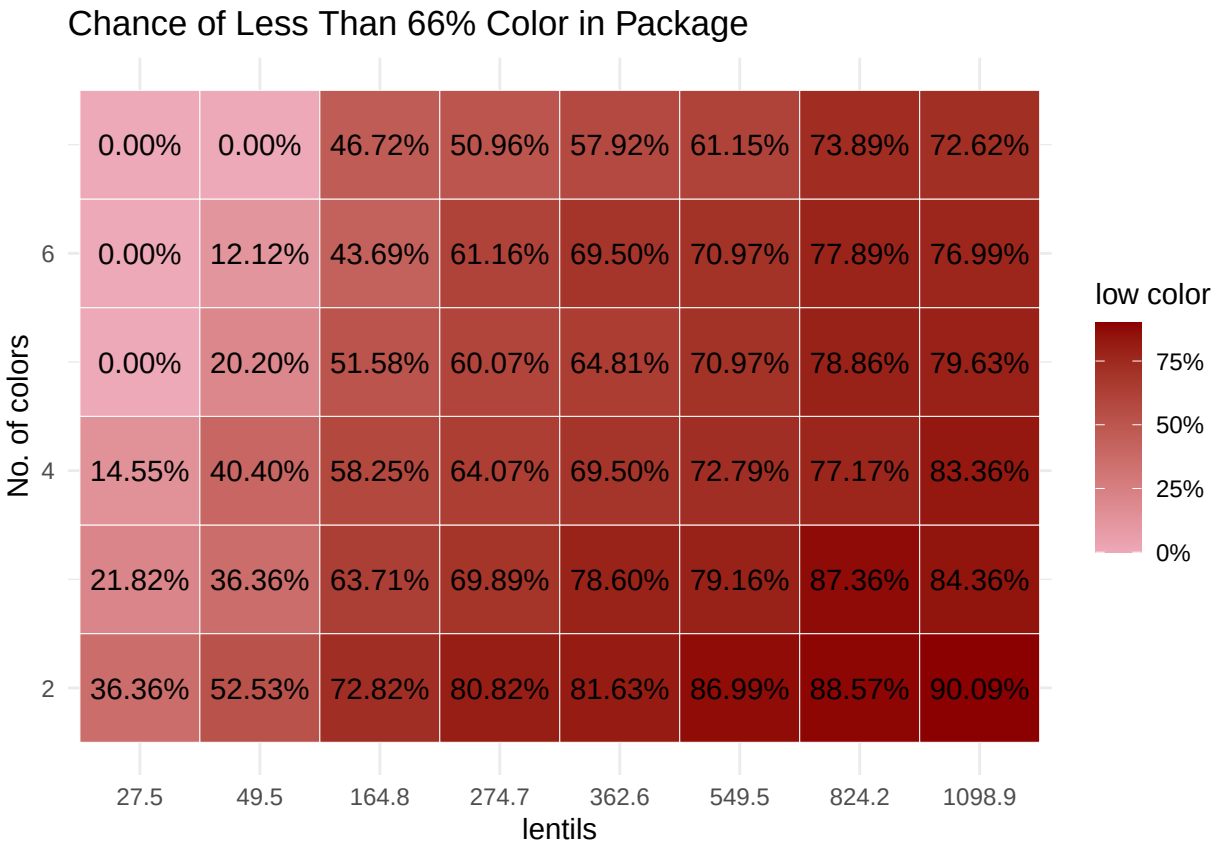



```
  labs(title = "Heatmap of mega_snack all evens",
       x = "lentils",
       y = "No. of colors",
       color = "all_even") +
  theme_minimal()
```

```
## NULL
```

```
mega_snack_1 %>%
  ggplot(aes(x = factor(round(n_unit, 1)), y = n_color, fill = low_color))+
  geom_tile(color = "white") +
    geom_text(aes(label = sprintf("%.2f%%", low_color * 100)), color = "black", size = 4)+
  scale_fill_gradient(low = "pink2", high = "red4",labels = scales::percent)+
  labs(title = "Chance of Less Than 66% Color in Package",
       x = "lentils",
       y = "No. of colors",
       fill = "low color")+
  theme_minimal()
```


Chance of Less Than 66% Color in Package
39.86%
23.57%
5.29%
2.86%
1.00%
0.29%
0.00%
0.00%
52.00%
28.14%
3.29%
0.86%
0.00%
0.00%
0.00%
0.00%
31.43%
19.43%
1.86%
0.14%
0.14%
0.00%
0.00%
0.00%
29.14%
20.43%
0.57%
0.14%
0.00%
0.00%
0.00%
0.00%
28.43%
7.14%
0.29%
0.00%
0.00%
0.00%
0.00%
0.00%
15.43%
2.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
No. of colors
6
4
2
27.5
49.5
164.8
274.7
362.6
549.5
824.2
1098.9
lentils
low color
50%
40%
30%
20%
10%
0%



As we can see, only the small package (less than 50 lentils) have high probability of at least one color to appear severely lower.

Therefore, splitting package by color on the big ones should be relatively even.