My M&M OCD

Yoni

01 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

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
##
## Attaching package: 'MASS'
##
##
## The following object is masked from 'package:dplyr':
##
##     select
##
##
##
```

```
## Attaching package: 'scales'
##
##
## The following object is masked from 'package:purrr':
##
##     discard
##
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

The basic parameters (will be changed later):

```r
#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 lentils per color is ", round(av_per_color,2))
```

```
## [1] "The avarage 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.

```r
#create bag using sample vector
create_bag<- function(x,tx)
  {
  x<- floor(x)+sample(0:1,1)
  res<- matrix(nrow = 1, ncol = tx)
  colnames(res)<- 1:tx
  S_res<- sample(tx,x,replace = T)
  while(sum(table(S_res)>x))
  {S_res<- S_res[-1]}
  for (i in 1:tx) {
    res[,i] <- sum(S_res==i)
    }
  res
}

#example
create_bag(100,6)
```

```
##       1  2  3  4  5  6
## [1,] 17 16 20 17 14 16
```

```
sample_MnM<-function(nn,x_units,t_colors)
  {
  res<- matrix(nrow = nn, ncol = t_colors)
  for (i in 1:nn)
    {res[i,]<- create_bag(x= x_units,tx=t_colors)}
  res
}
```

Preview Graph

Now will be creating nn bugs of M&M

```
MnM_sample<- sample_MnM(nn*100,n_unit,n_color)

MnM_sample<-
  MnM_sample %>% as_data_frame() %>%
  mutate(even_count= rowSums(across(everything()  , ~ .x %% 2 == 0)), #how many evens colors there are
         even_evens= rowSums(across(-c(even_count)  , ~ .x %% 2 == 1)) %% 2 ==0, #are the uneven colors
         var_col=    apply(across(-c(even_count, even_evens)), 1, var), #var of candy per color/ type
         low_col=   rowSums(across(-c(even_count, even_evens,var_col), ~ .x <= 0.6*av_per_color)))

MnM_sample %>% head(6)
```

```
## # A tibble: 6 x 10
##      V1    V2    V3    V4    V5    V6 even_count even_evens var_col low_col
##   <int> <int> <int> <int> <int> <int>      <dbl> <lgl>        <dbl>   <dbl>
## 1    50    39    39    55    51    41          1 FALSE         49.0       0
## 2    42    48    33    49    49    53          2 TRUE          51.1       0
## 3    50    44    37    50    56    38          5 FALSE         56.2       0
## 4    47    44    38    53    51    41          2 TRUE          33.5       0
## 5    47    45    48    39    48    48          3 FALSE         12.6       0
## 6    42    47    45    41    41    58          2 TRUE          42.3       0
```

```
MnM_sample$low_col %>% table()
```

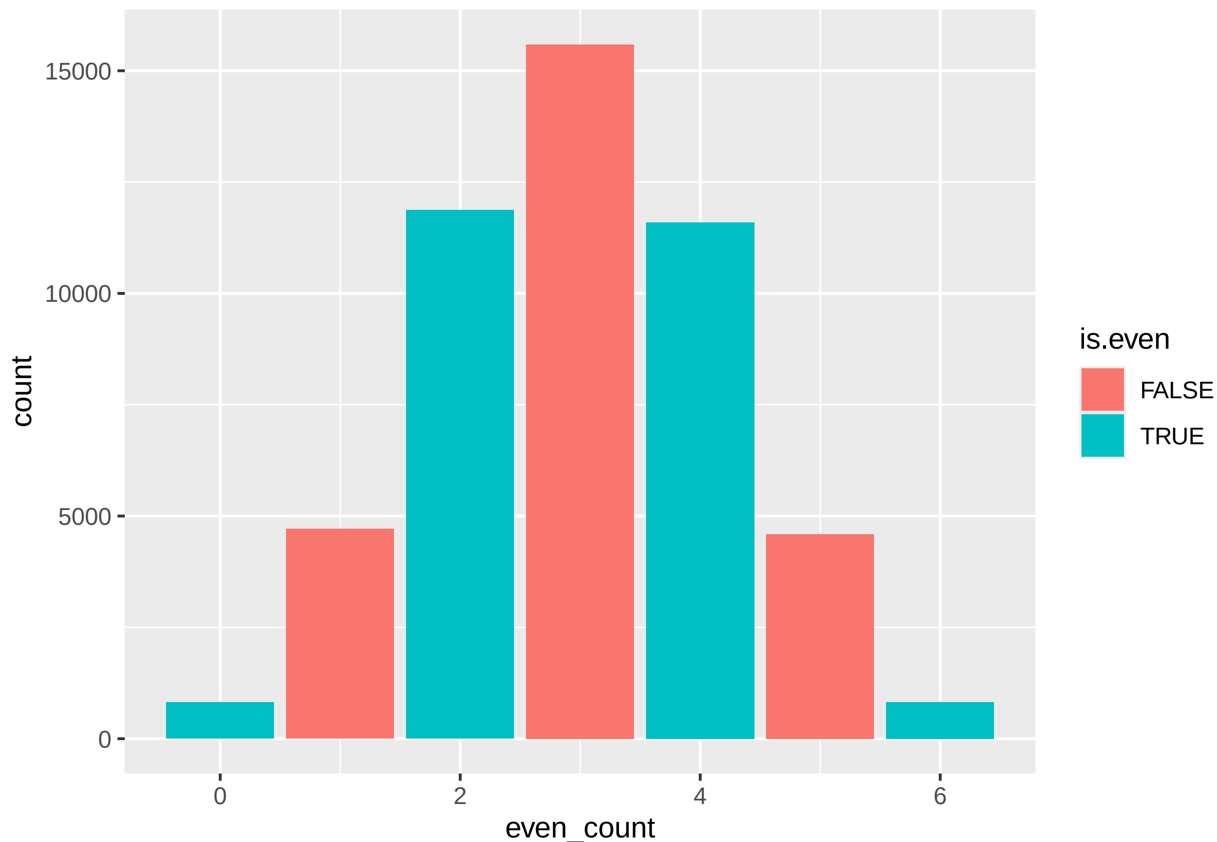
```
## .
##     0     1
## 49738   262
```

ggplot the MnM sample sample

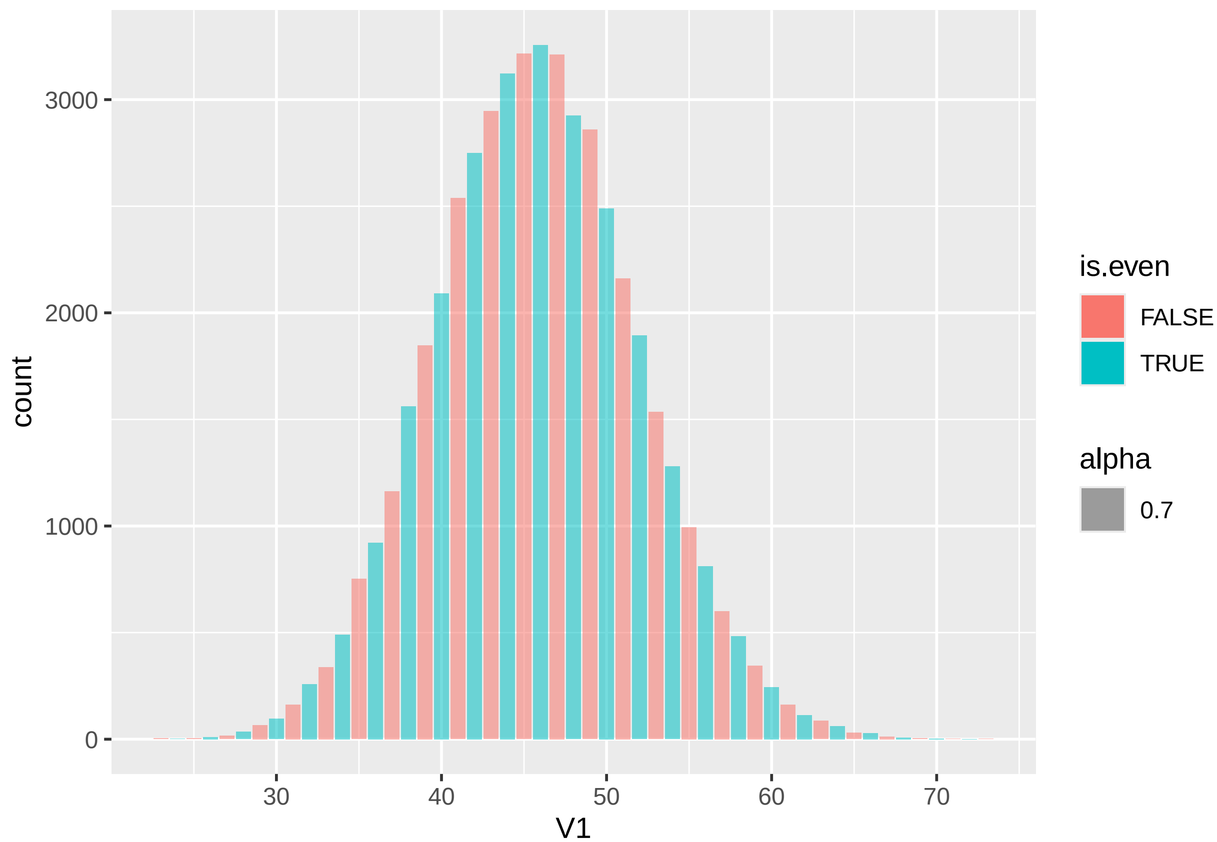
```
MnM_sample$even_count %>%
table()
```

```
## .
##     0     1     2     3     4     5     6
##   816  4710 11869 15593 11597  4590   825
```

```r
MnM_sample %>%
  mutate(is.even = even_count %%2 ==0) %>%
  ggplot(aes(x= even_count, fill= is.even))+
  geom_bar()
```




```r
MnM_sample %>%
  mutate(is.even = V1 %%2 ==0) %>%
  ggplot(aes(x= V1, fill= is.even, alpha= 0.7))+
  geom_bar()
```




```
#summary of all colors
rbind(
MnM_sample$V1 %>% summary(),
MnM_sample$V2 %>% summary(),
MnM_sample$V3 %>% summary(),
MnM_sample$V4 %>% summary(),
MnM_sample$V5 %>% summary(),
MnM_sample$V6 %>% summary()
) %>% as.data.frame() %>% cbind(sapply(MnM_sample[,1:6],var,na.rm=1)) %>%
  rename("Var" ="sapply(MnM_sample[, 1:6], var, na.rm = 1)")
```

```
##    Min. 1st Qu. Median     Mean 3rd Qu. Max.      Var
## V1   23      42     46 45.77100      50   73 37.89392
## V2   21      42     46 45.77948      50   71 38.07433
## V3   24      42     46 45.72078      50   72 37.99138
## V4   22      42     46 45.76166      50   75 37.98857
## V5   20      41     46 45.72052      50   74 38.65870
## V6   23      42     46 45.74442      50   78 37.74649
```

Test Expected Value

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

```r
#test mu is av_per_color
check_mean_hypothesis <- function(data, column_name, X) {
  test_result <- t.test(data[[column_name]], mu = X)
  return(test_result$p.value)
}

columns_to_test <- colnames(MnM_sample[,1:6])
test_results <- sapply(columns_to_test, function(col) {
  check_mean_hypothesis(MnM_sample, col, av_per_color)
})

t(t(test_results))
```

```
##          [,1]
## V1 0.54783052
## V2 0.77006526
## V3 0.01543359
## V4 0.34767501
## V5 0.01593519
## V6 0.11651895
```

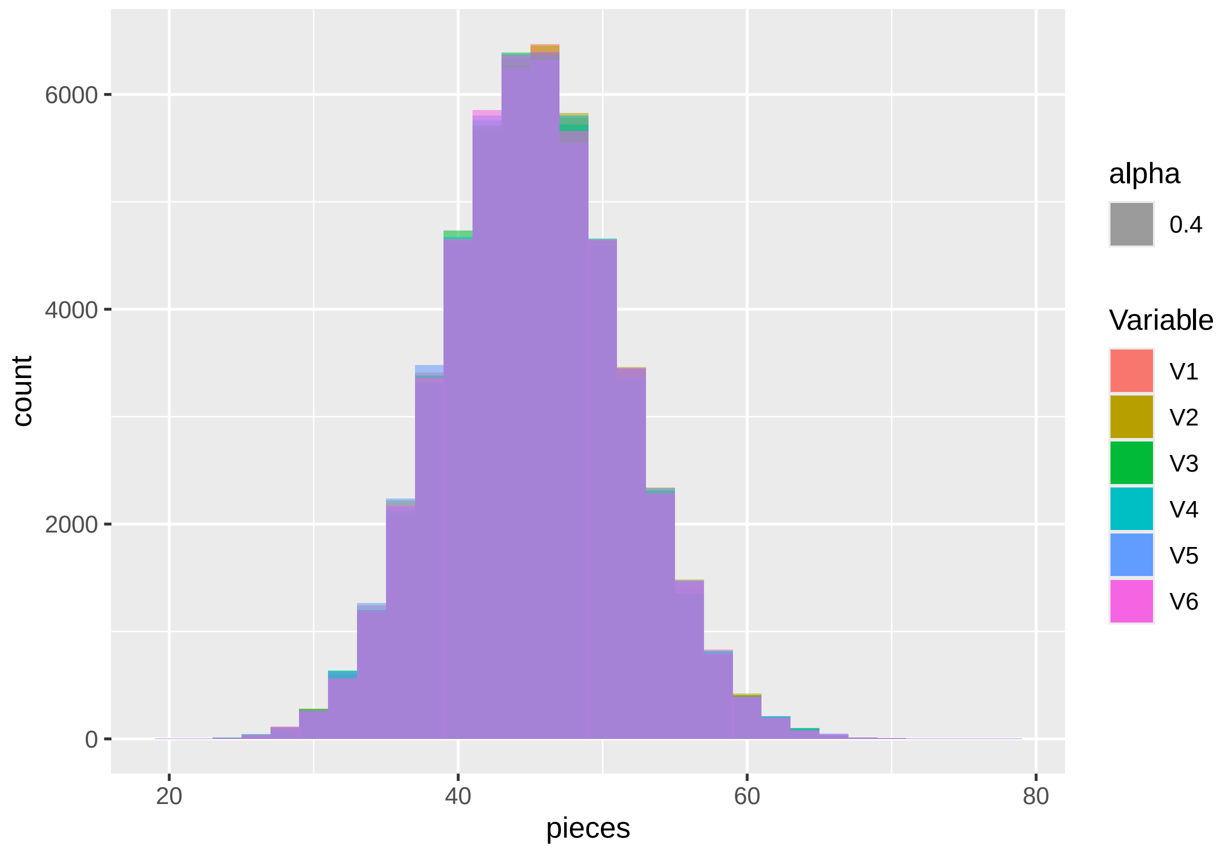
```r
check_mean_hypothesis <- function(data, column_name, X) {
  t.test(data[[column_name]], mu = X)$p.value}

check_two_sample_t_test <- function(data, col1, col2) {
  t.test(data[[col1]], data[[col2]])$p.value}

columns_to_test <- colnames(MnM_sample[,1:6])
num_cols <- length(columns_to_test)
p_value_matrix <- matrix(NA, nrow = num_cols, ncol = num_cols, dimnames = list(columns_to_test, columns_
for (i in 1:num_cols) {
  for (j in 1:num_cols) {
    if (i == j) { #diagonal test of mu
      p_value_matrix[i, j] <- check_mean_hypothesis(MnM_sample, columns_to_test[i], av_per_color)
    } else { # 2 sample t.test
      p_value_matrix[i, j] <- check_two_sample_t_test(MnM_sample, columns_to_test[i], columns_to_test[j]
    }
  }
}

round(as.data.frame(p_value_matrix),4)
```

```
##        V1     V2     V3     V4     V5     V6
## V1 0.5478 0.8278 0.1974 0.8105 0.1970 0.4944
## V2 0.8278 0.7701 0.1323 0.6478 0.1323 0.3679
## V3 0.1974 0.1323 0.0154 0.2943 0.9947 0.5436
## V4 0.8105 0.6478 0.2943 0.3477 0.2934 0.6578
## V5 0.1970 0.1323 0.9947 0.2934 0.0159 0.5409
## V6 0.4944 0.3679 0.5436 0.6578 0.5409 0.1165
```

```r
MnM_sample %>%
    pivot_longer(cols = 1:6, names_to = "Variable", values_to = "pieces") %>%
```

```
    ggplot(aes(fill=Variable ,x= pieces, alpha= 0.4))+
  geom_histogram(position = "identity")
```

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



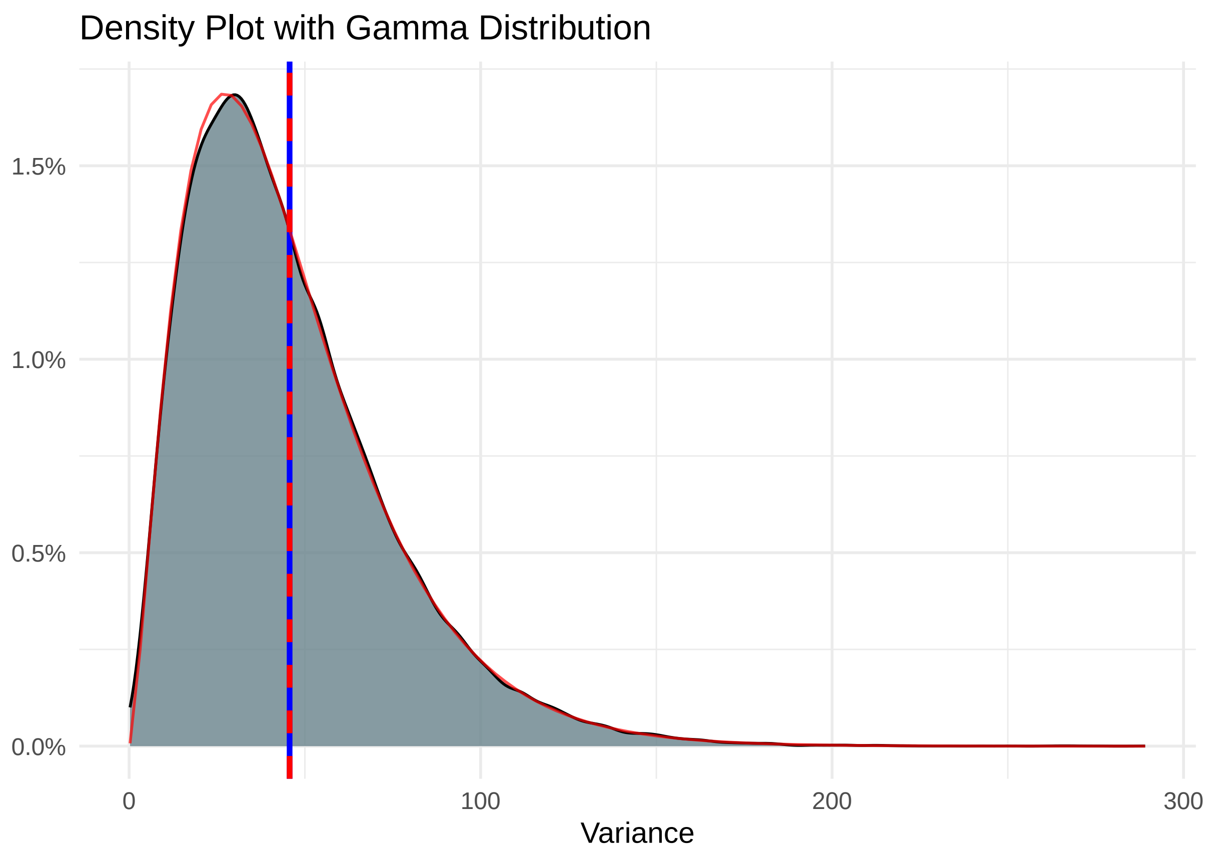
we can see that the variance distribution is Gamma like with shape and rate as seen below

```
gamma_params <- fitdistr(MnM_sample$var_col, "gamma")$estimate
paste0("the parameters of the gamma shaped var is shape ", round(gamma_params[1],3)," and rate ", round
```

```
## [1] "the parameters of the gamma shaped var is shape 2.493 and rate 0.055"
```

```
shape_est <- gamma_params["shape"]
scale_est <- 1 / gamma_params["rate"] #using Scale= 1/Rate

MnM_sample %>% #plot density
  ggplot(aes(x = var_col, fill = "orange")) +
  geom_density(fill= "lightblue4", alpha= 0.8) +
  geom_vline(xintercept = mean(MnM_sample$var_col), color = "blue", size=1) +
  geom_vline(xintercept = scale_est*shape_est, color = "red", linetype = "dashed", size=1) +
  stat_function(fun = dgamma, args = list(shape = shape_est, scale = scale_est),
                color = "red", alpha= 0.7) +
  labs(title = "Density Plot with Gamma Distribution",
       x = "Variance",
```

```
      y = "") +
  scale_y_continuous(label=scales::label_percent(.1)) +
  theme_minimal()
```



#check too low color (under 10%) and sample by n number

#use statistics to sample better low chance cases

optimizing best package

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

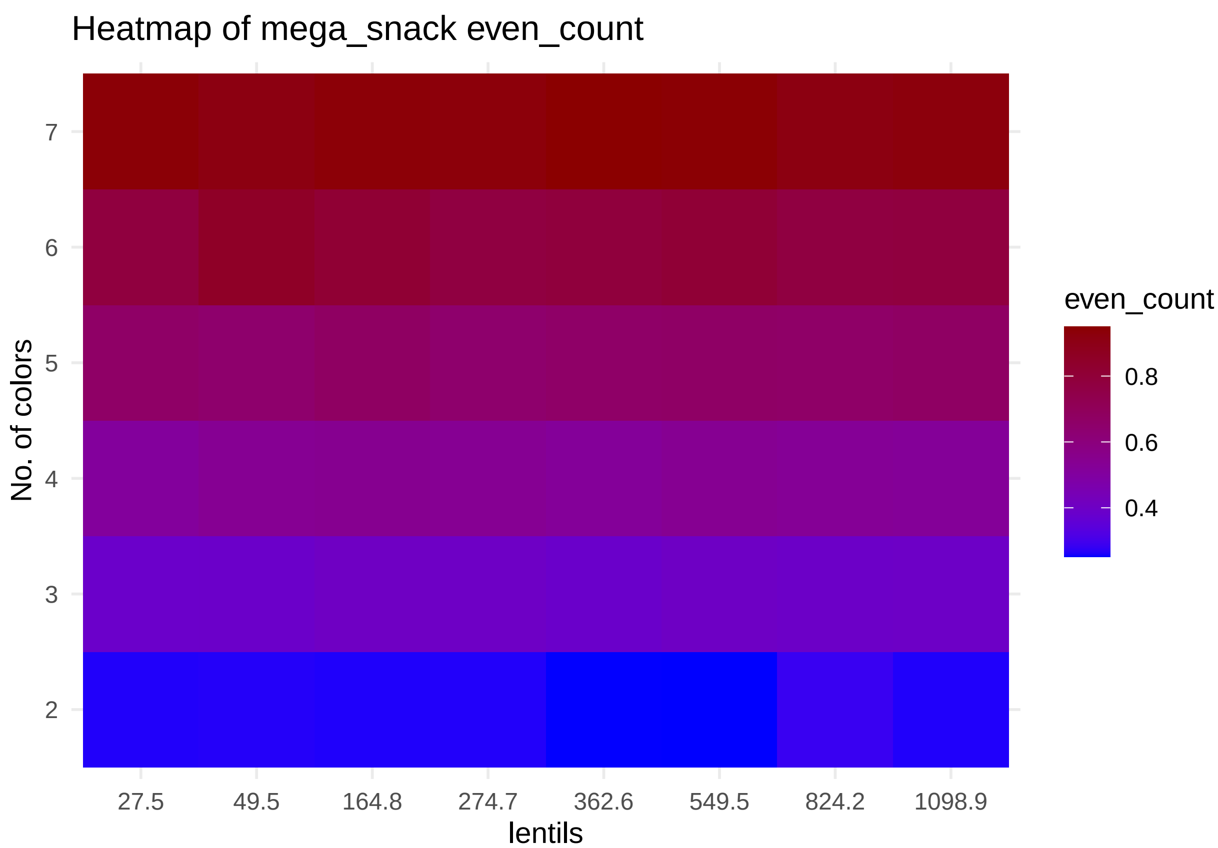
```
mega_snack<- function(nn,n_unit,n_color)
  {
  m_sample<- length(n_unit)*length(n_color)
  nul_mat= matrix(,nrow = m_sample, ncol = 5)
  res<- cbind(rep(n_unit,length(n_color)),sort(rep(n_color,length(n_unit))),
                                               nul_mat)
  for (i in 1:(dim(res)[1]))
  {
    #print(c(res[i,1],res[i,2]))
```

```r
    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 color
             even_evens= (rowSums(across(-c(even_count)  , ~ .x %% 2 == 1)) %% 2 ==0)/n_color, #are the
             var_col=    apply(across(c(-even_count, even_evens)), 1, var), #var of candy per color/ ty
             all_even=   rowSums(across(everything()  , ~ .x %% 2 == 0))== n_color,
             low_col=   rowSums(across(-c(even_count, even_evens,var_col,all_even), ~ .x <= low_color )
    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)
      }
  colnames(res)<- c("n_unit", "n_color", "even_count", "even_evens", "var_col","all_even","low_color")
  res
  }
```

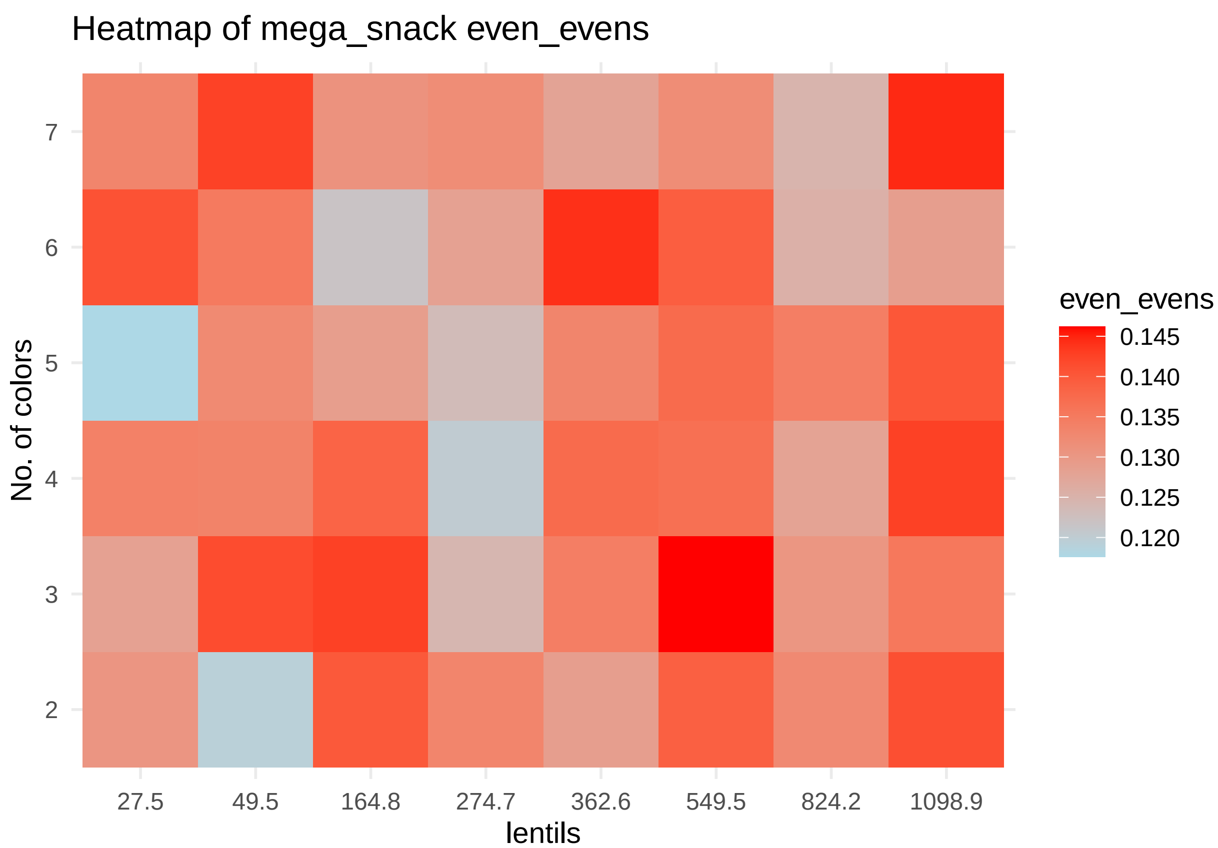
```r
color_op<- 2:7
grams_op<- c(25,45,150,250,330,500,750,1000)
n_unit_op<- grams_op/gram

mega_snack_1<-
  mega_snack(500,n_unit_op,color_op) %>% as.data.frame() %>%
  mutate(n_unit= round(n_unit,1))

mega_snack_1 %>% head(6)
```

```
##   n_unit n_color even_count even_evens     var_col all_even low_color
## 1   27.5       2  0.2640857  0.1304429    68.95187    0.140     0.124
## 2   49.5       2  0.2664667  0.1193381   215.47709    0.132     0.030
## 3  164.8       2  0.2629143  0.1399952  2284.09289    0.120     0.000
## 4  274.7       2  0.2650190  0.1333571  6332.78797    0.118     0.000
## 5  362.6       2  0.2530095  0.1288810 11014.58442    0.126     0.000
## 6  549.5       2  0.2524714  0.1390286 25287.04647    0.118     0.000
```

```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()
```

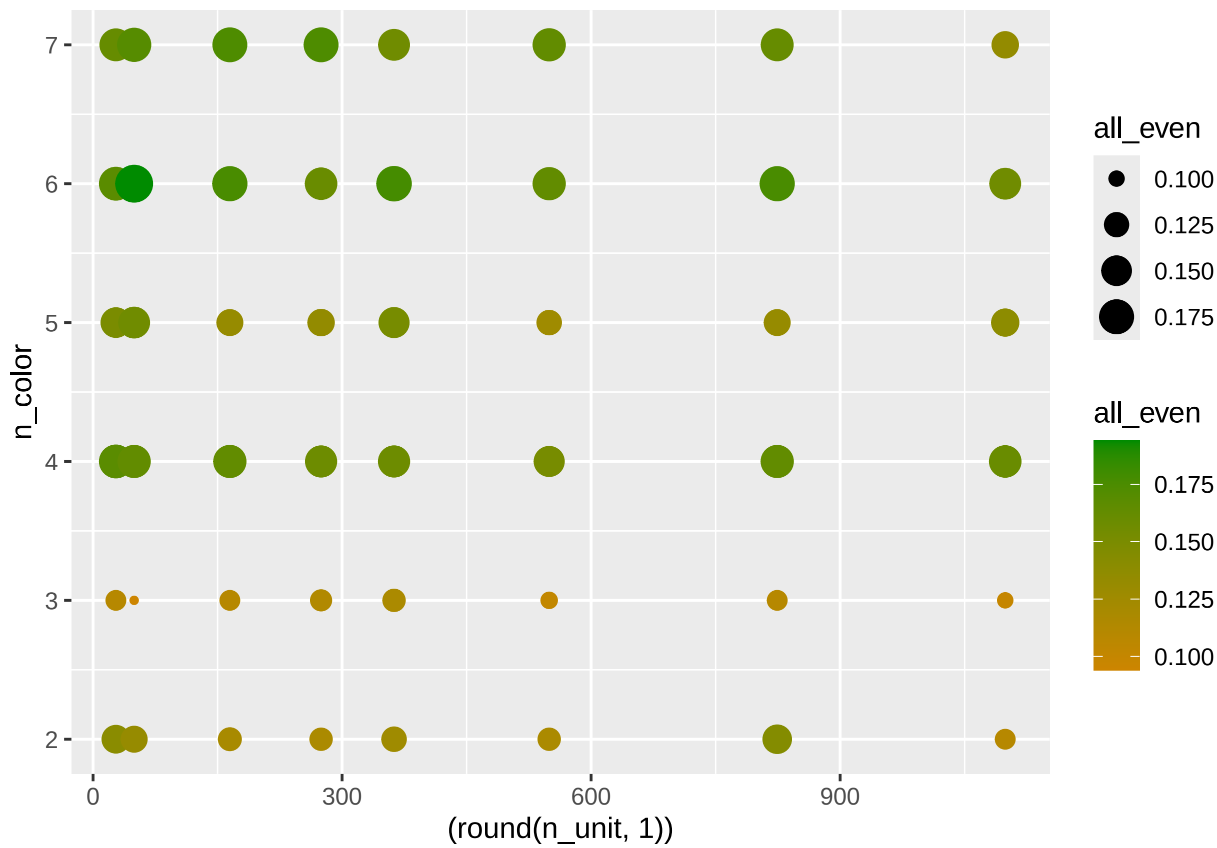



```
mega_snack_1 %>%
  ggplot(aes(x = factor((round( n_unit,1) )), y = factor(n_color ), fill = even_evens )) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "red")+
  labs(title = "Heatmap of mega_snack even_evens",
       x = "lentils",
       y = "No. of colors",
       fill = "even_evens")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



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

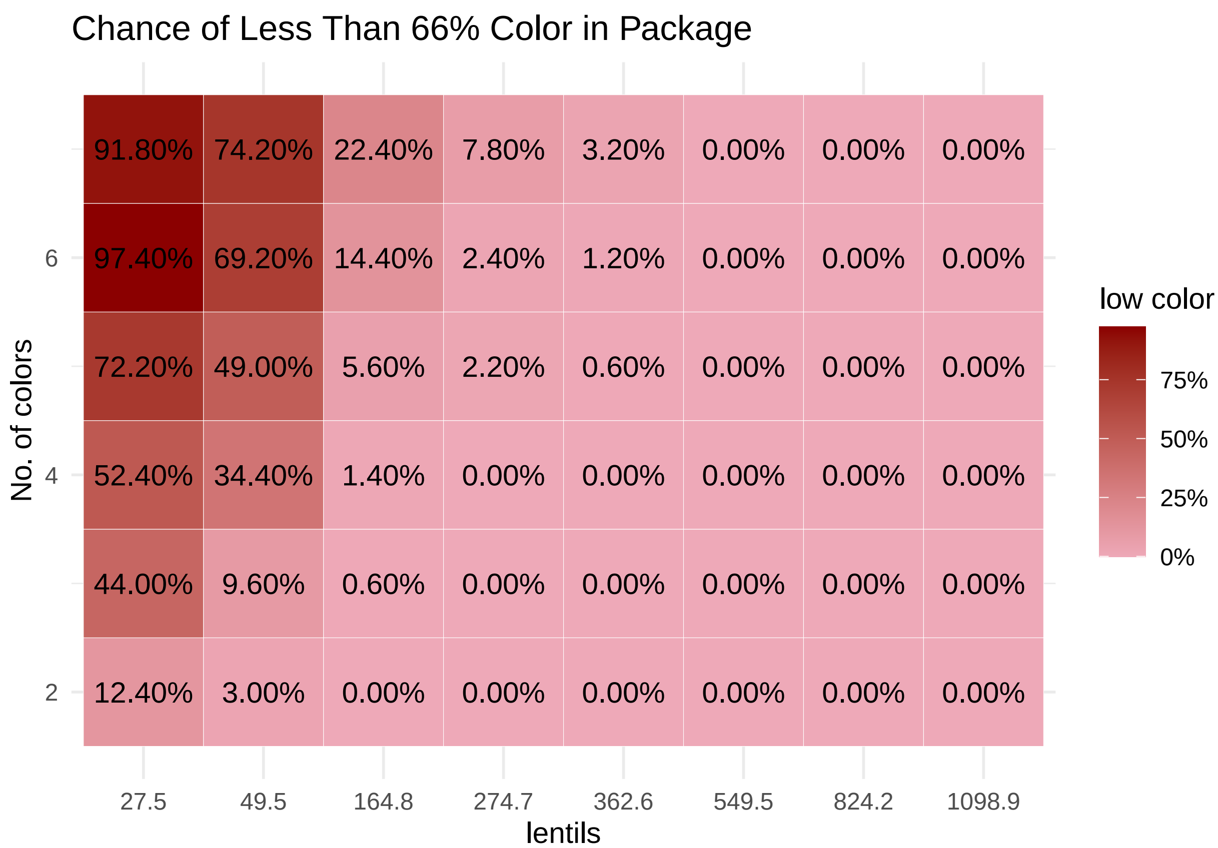
```
mega_snack_1 %>%
  ggplot(aes(x = (round( n_unit,1) ), y = n_color , color = all_even,size = all_even )) +
  geom_point() +
  scale_color_gradient(low = "orange3", high = "green4")
```




```
  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()
```



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.