CSDA1040_Lab1_Group2

Aaron

9/16/2020

Load the libraries

Load the data

Peek at the dataset 1

```
kable(head(orders,10))
```

order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
2539329	1	prior	1	2	8	NA
2398795	1	prior	2	3	7	15
473747	1	prior	3	3	12	21
2254736	1	prior	4	4	7	29
431534	1	prior	5	4	15	28
3367565	1	prior	6	2	7	19
550135	1	prior	7	1	9	20
3108588	1	prior	8	1	14	14
2295261	1	prior	9	1	16	0
2550362	1	prior	10	4	8	30

```
kable(head(products,10))
```

product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich Cookies	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Unsweetened Oolong Tea	94	7
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce	38	1
5	Green Chile Anytime Sauce	5	13
6	Dry Nose Oil	11	11
7	Pure Coconut Water With Orange	98	7
8	Cut Russet Potatoes Steam N' Mash	116	1
9	Light Strawberry Blueberry Yogurt	120	16
10	Sparkling Orange Juice & Prickly Pear Beverage	115	7

```
kable(head(order_products,10))
```

order_id	product_id	add_to_cart_order	reordered
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1
1	13176	6	0
1	47209	7	0
1	22035	8	1
36	39612	1	0
36	19660	2	1

```
kable(head(order_products_prior,10))
```

order_id	product_id	add_to_cart_order	reordered
2	33120	1	1
2	28985	2	1
2	9327	3	0
2	45918	4	1
2	30035	5	0
2	17794	6	1
2	40141	7	1
2	1819	8	1
2	43668	9	0
3	33754	1	1

```
kable(head(aisles,10))
```

aisle_id	aisle
1	prepared soups salads
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat preparation
6	other
7	packaged meat
8	bakery desserts
9	pasta sauce
10	kitchen supplies

```
kable(head(departments,10))
```

department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol
6	international
7	beverages
8	pets
9	dry goods pasta
10	bulk

Recode variables

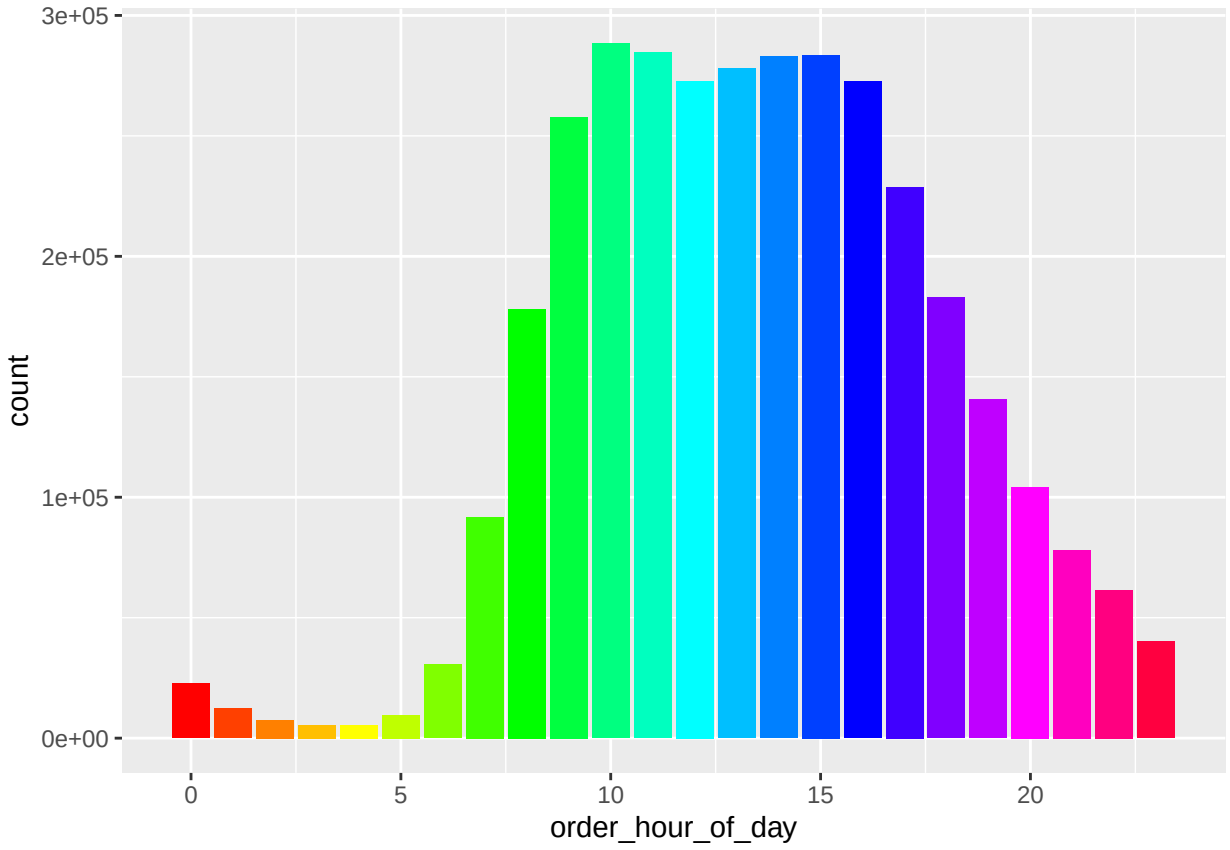
```
orders <- orders %>% mutate(order_hour_of_day = as.numeric(order_hour_of_day),
                            eval_set = as.factor(eval_set))
products <- products %>% mutate(product_name = as.factor(product_name))
aisles <- aisles %>% mutate(aisle = as.factor(aisle))
departments <- departments %>% mutate(department = as.factor(department))
```

The order behaivor

Hour of Day

```
orders %>%
  ggplot(aes(x=order_hour_of_day)) +
  geom_histogram(stat="count",fill=rainbow(24))
```

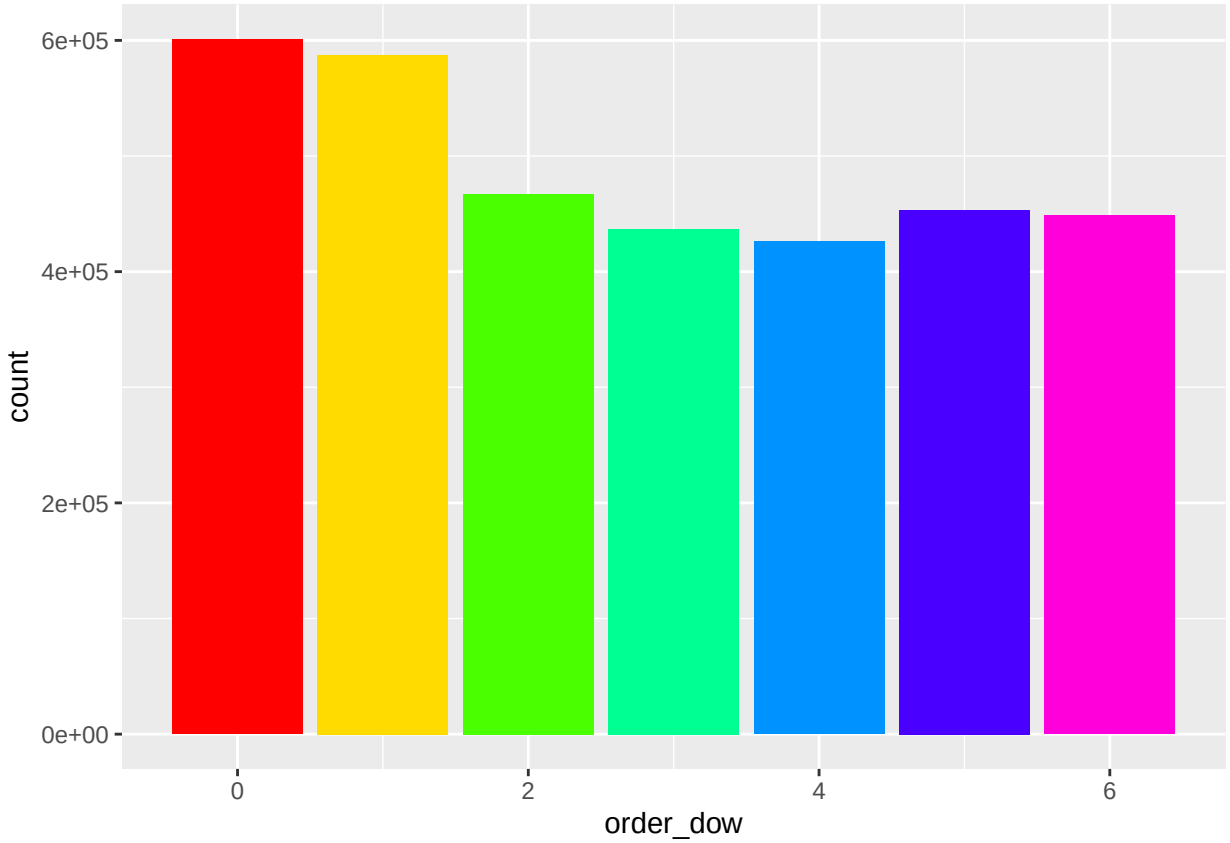
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



Day of Week

```
orders %>%
  ggplot(aes(x=order_dow)) +
  geom_histogram(stat="count",fill=rainbow(7))
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

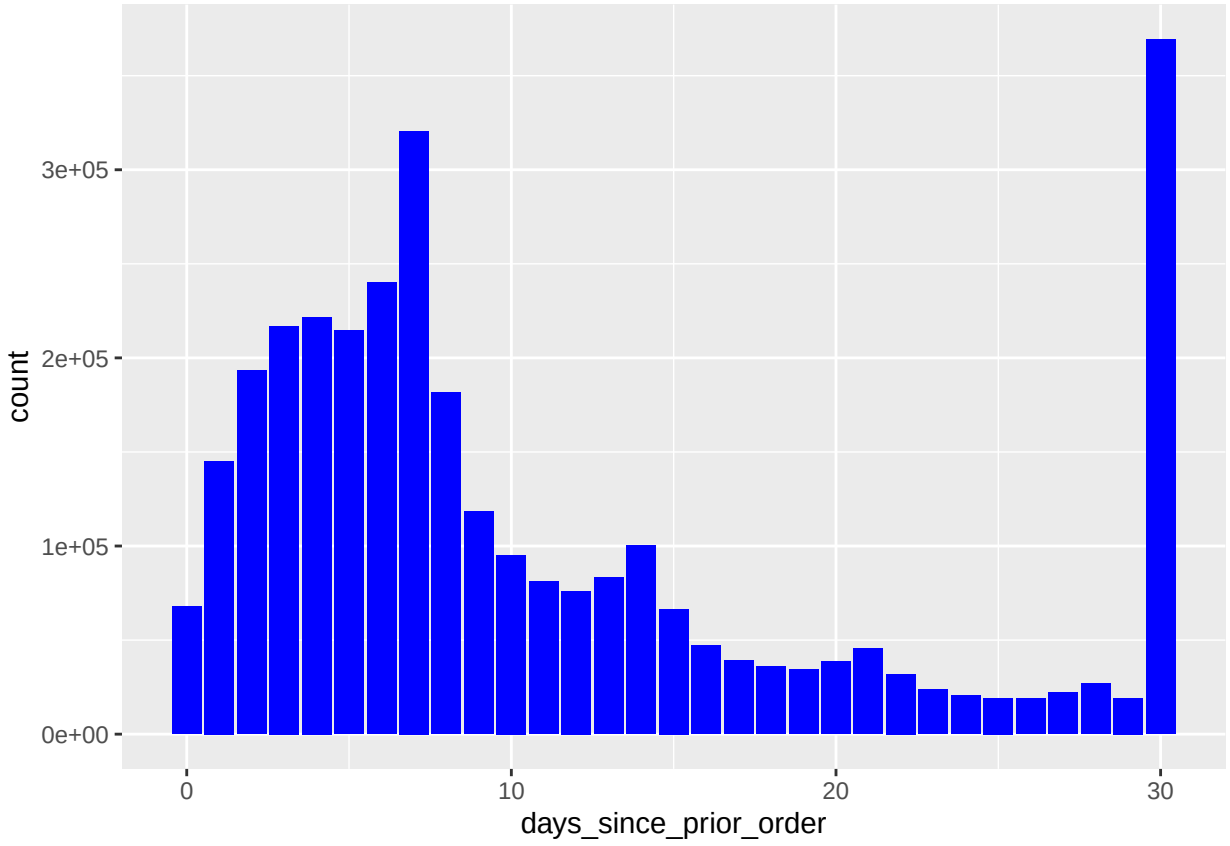


When do they order again

```
orders %>%
  ggplot(aes(x=days_since_prior_order)) +
  geom_histogram(stat="count",fill="blue")
```

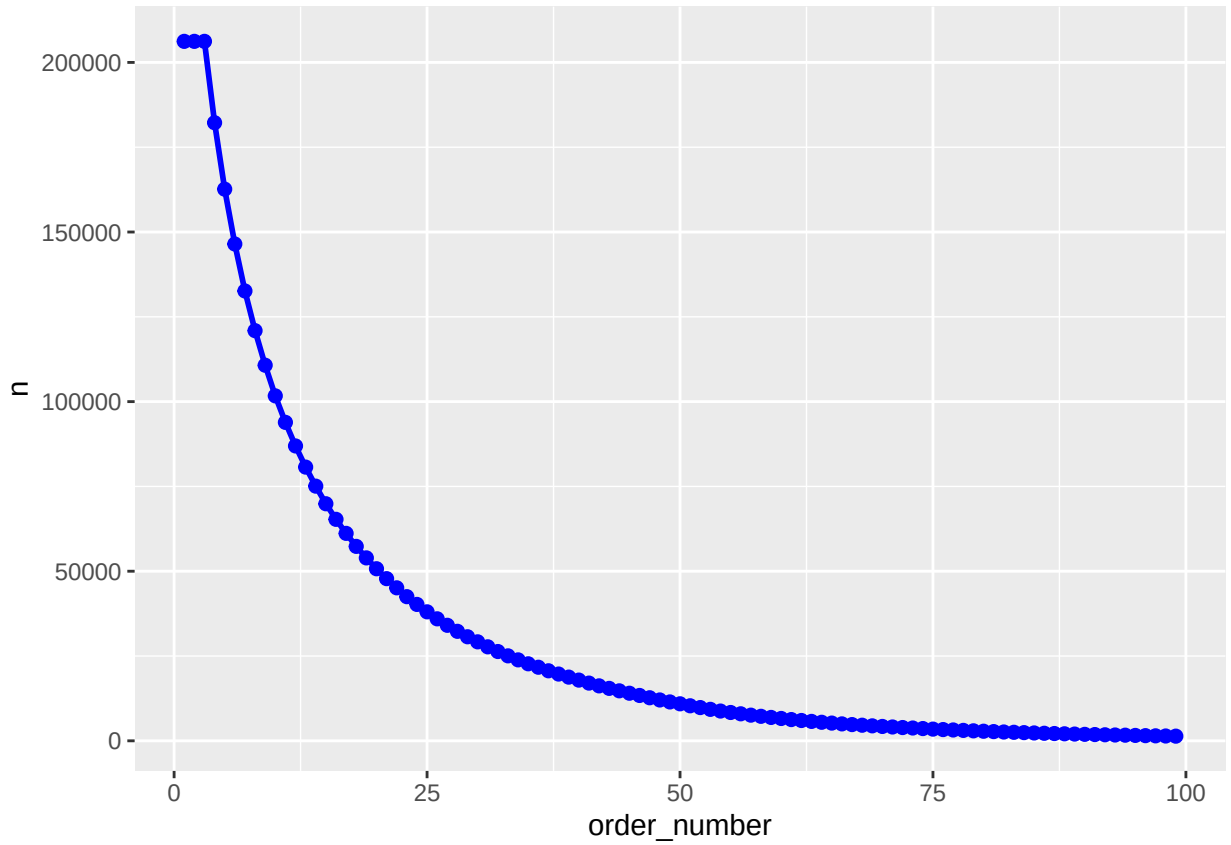
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
## Warning: Removed 206209 rows containing non-finite values (stat_count).
```



How many prior orders are there?

```
prior_order = orders %>%
  filter(eval_set=="prior") %>%
  count(order_number)
ggplot(data = prior_order, aes(order_number,n)) +
  geom_line(color="blue", size=1) +
  geom_point(color="blue", size=2)
```



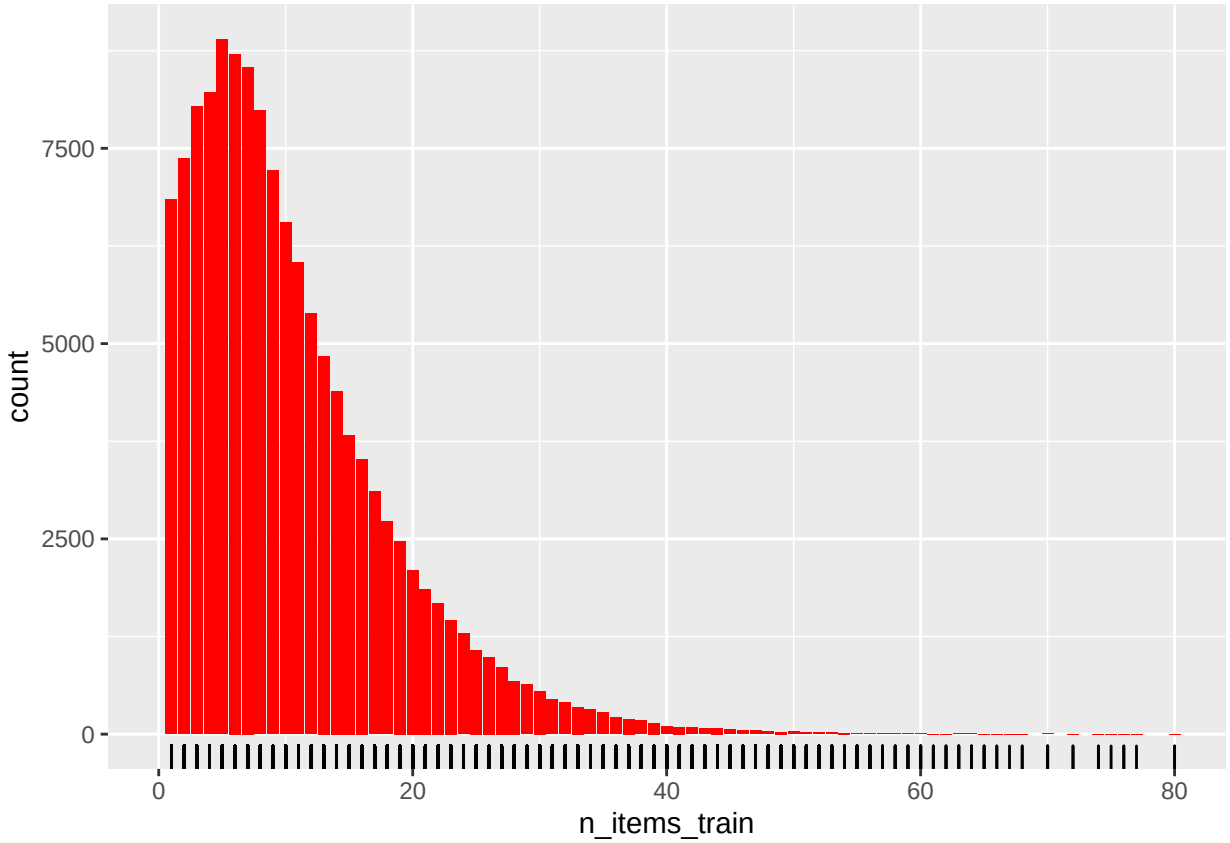
The distributions of how many items are in the orders

```
train_1 <- order_products %>%
  group_by(order_id) %>%
  summarize(n_items_train = last(add_to_cart_order))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(data=train_1, aes(x=n_items_train),col="red",alpha = 0.3) +
  geom_histogram(stat="count",fill="red") +
  geom_rug() +
  coord_cartesian(xlim=c(0,80))
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



How often do people order the same items again

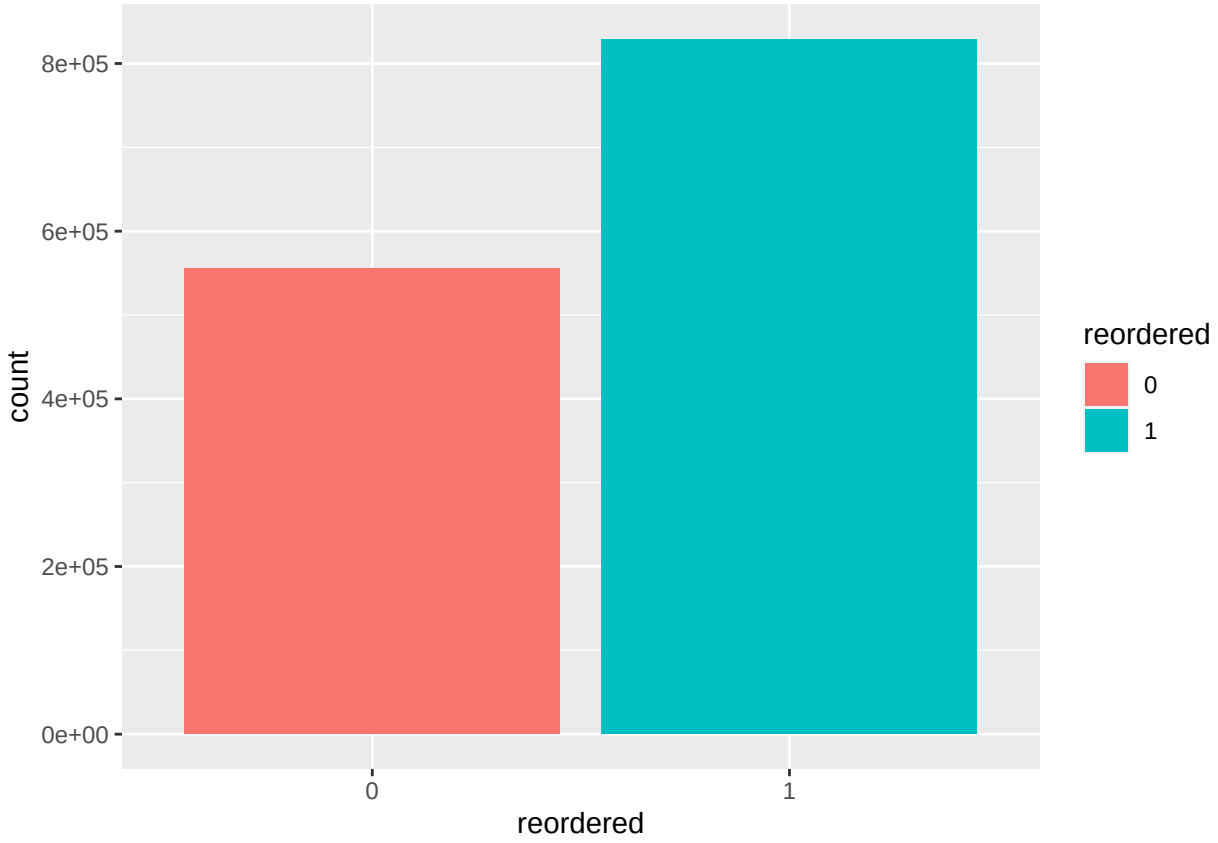
```
re_order <- order_products %>%
  group_by(reordered) %>%
  summarize(count = n()) %>%
  mutate(reordered = as.factor(reordered)) %>%
  mutate(proportion = count/sum(count))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(re_order)
```

```
## # A tibble: 2 x 3
##   reordered  count proportion
##   <fct>      <int>      <dbl>
## 1 0         555793      0.401
## 2 1         828824      0.599
```

```
ggplot(re_order, aes(x=reordered,y=count,fill=reordered))+
  geom_bar(stat="identity")
```



Most Popular Products Sold

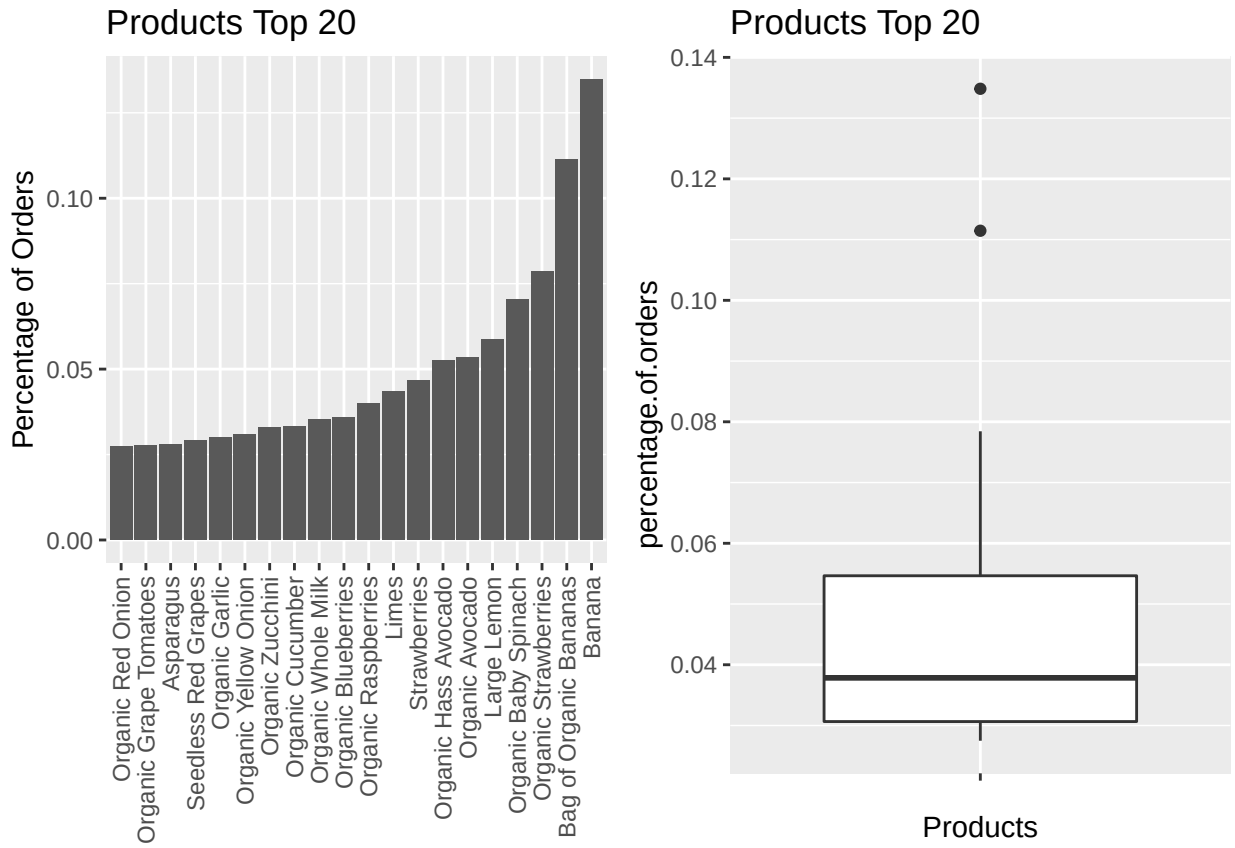
```
tmp1 <- order_products %>%
  left_join(products) %>%
  group_by(product_name) %>%
  summarize(count=n()) %>%
  top_n(n=20, wt=count) %>%
  mutate(percentage=count/sum(count))
```

```
## Joining, by = "product_id"
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p1 = ggplot (tmp1, aes(x=reorder(product_name,count), y=percentage)) +
  geom_col() +
  ggtitle('Products Top 20') +
  ylab('Percentage of Orders') +
  theme (axis.text.x=element_text(angle=90, hjust=1, vjust=0.5),
         axis.title.x = element_blank())
p2 = ggplot (data = tmp1, aes( x= '', y=percentage )) +
  ggtitle('Products Top 20') +
  ylab('percentage.of.orders') +
  geom_boxplot() +
```

```
  xlab('Products')
grid.arrange(p1, p2, ncol = 2)
```



Most Popular Department Sold

```
tmp2 <- order_products %>%
  left_join(products) %>%
  left_join(departments) %>%
  group_by(department) %>%
  summarize(count=n()) %>%
  mutate(percentage=count/sum(count))
```

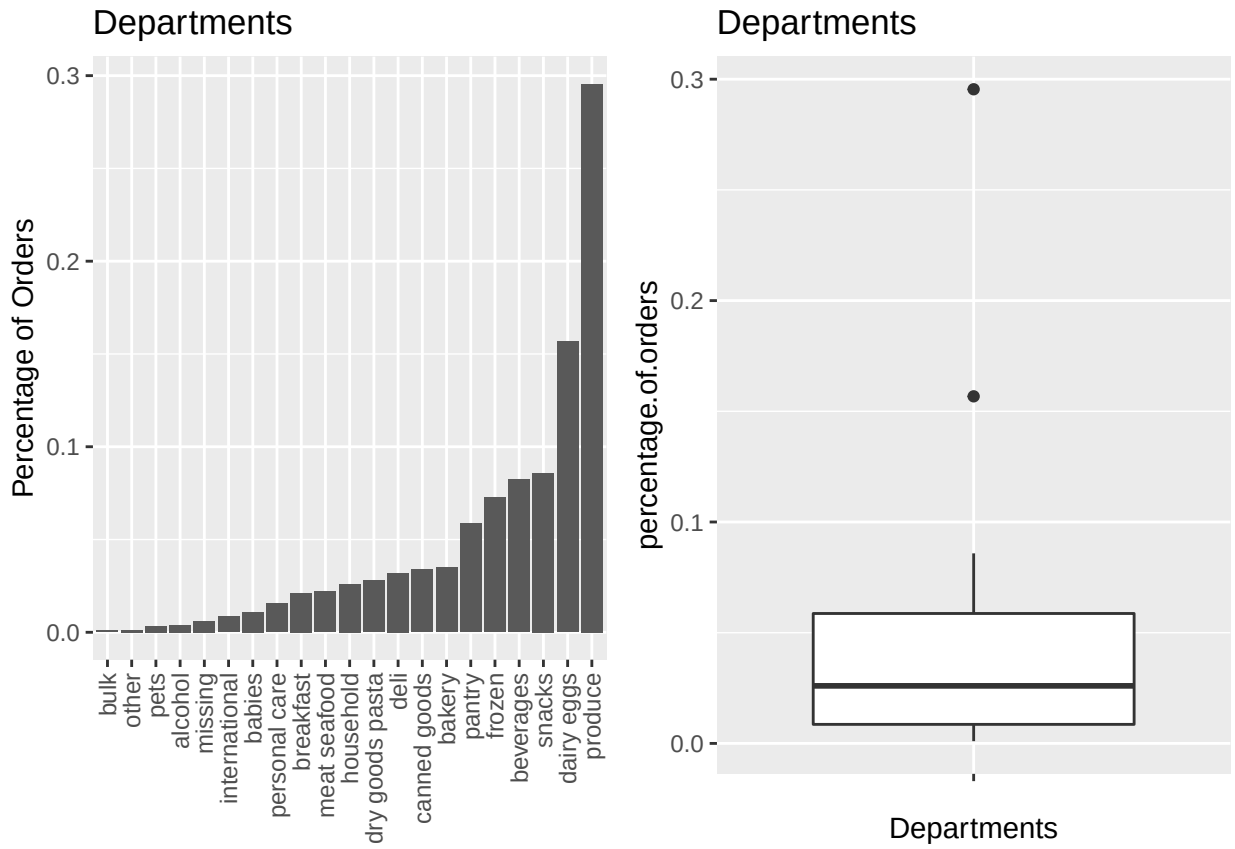
```
## Joining, by = "product_id"
```

```
## Joining, by = "department_id"
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p1 = ggplot (tmp2, aes(x=reorder(department,count), y=percentage)) +
  geom_col() +
  ggtitle('Departments') +
  ylab('Percentage of Orders') +
```

```
  theme (axis.text.x=element_text(angle=90, hjust=1, vjust=0.5),
         axis.title.x = element_blank())
p2 = ggplot (data = tmp2, aes( x= '', y=percentage )) +
  ggtitle('Departments') +
  ylab('percentage.of.orders') +
  geom_boxplot() +
  xlab('Departments')
grid.arrange(p1, p2, ncol = 2)
```



Most Popular Aisles Sold

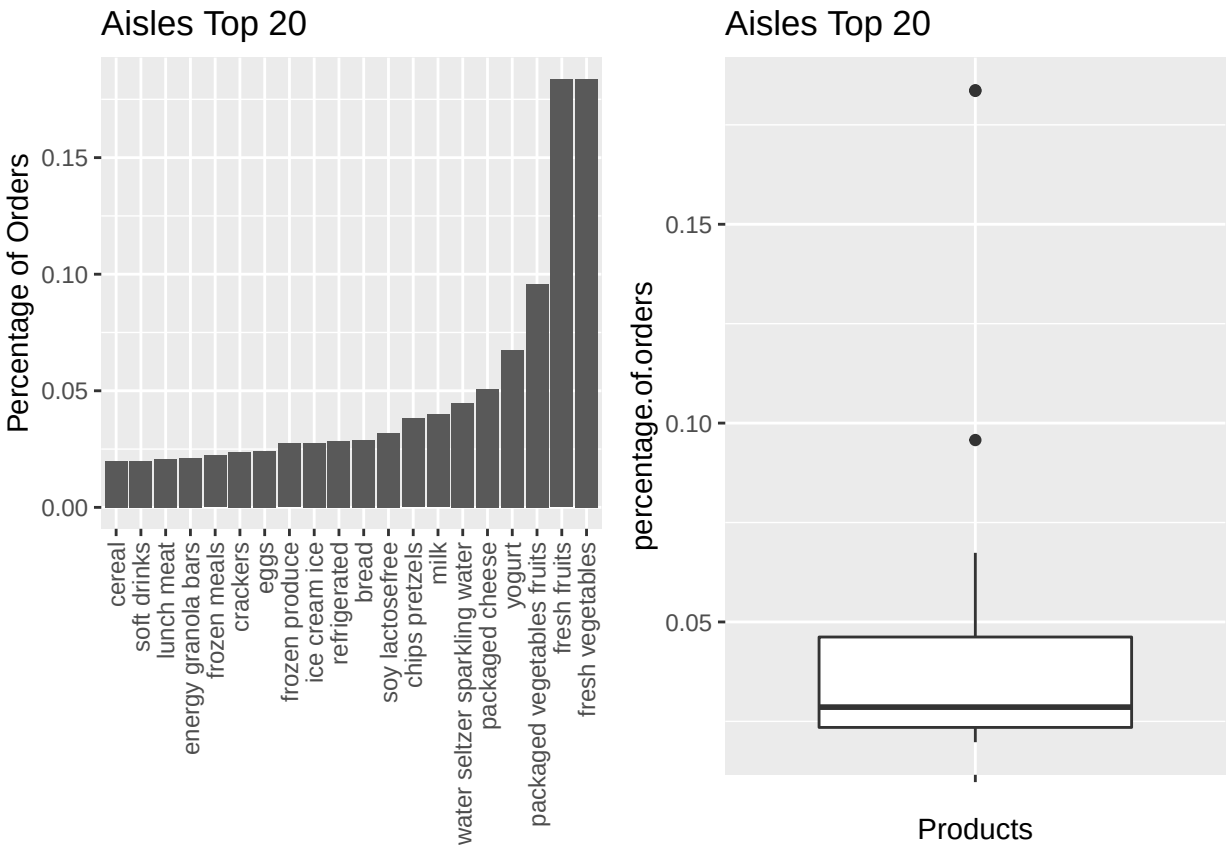
```
tmp3 <- order_products %>%
  left_join(products) %>%
  left_join(aisles) %>%
  group_by(aisle) %>%
  summarize(count=n()) %>%
  top_n(n=20, wt=count) %>%
  mutate(percentage=count/sum(count))
```

```
## Joining, by = "product_id"
```

```
## Joining, by = "aisle_id"
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p5 = ggplot (tmp3, aes(x=reorder(aisle,count), y=percentage)) +
  geom_col() +
  ggtitle('Aisles Top 20') +
  ylab('Percentage of Orders') +
  theme (axis.text.x=element_text(angle=90, hjust=1, vjust=0.5),
         axis.title.x = element_blank())
p6 = ggplot (data = tmp3, aes( x= '', y=percentage )) +
  ggtitle('Aisles Top 20') +
  ylab('percentage.of.orders') +
  geom_boxplot() +
  xlab('Products')
grid.arrange(p5, p6, ncol = 2)
```



Top ten products ordered daily contributes between 7% to 8%.

```
order_products_prior %>%
  left_join(orders) %>% left_join(products) %>%
  group_by(order_dow, product_name) %>%
  summarize(n=n()) %>%
  mutate(percentage=n/sum(n)) %>%
  top_n(10, wt=n) %>%
```

```
  ggplot (aes(x=as.factor(order_dow), y=percentage, fill=product_name)) +
  geom_col() + ylab('Proprtion of Orders') + ggtitle('Daily Top 10 Products Ordered') +
  theme(legend.position="bottom",legend.direction="horizontal")
```

```
## Joining, by = "order_id"
```

```
## Joining, by = "product_id"
```

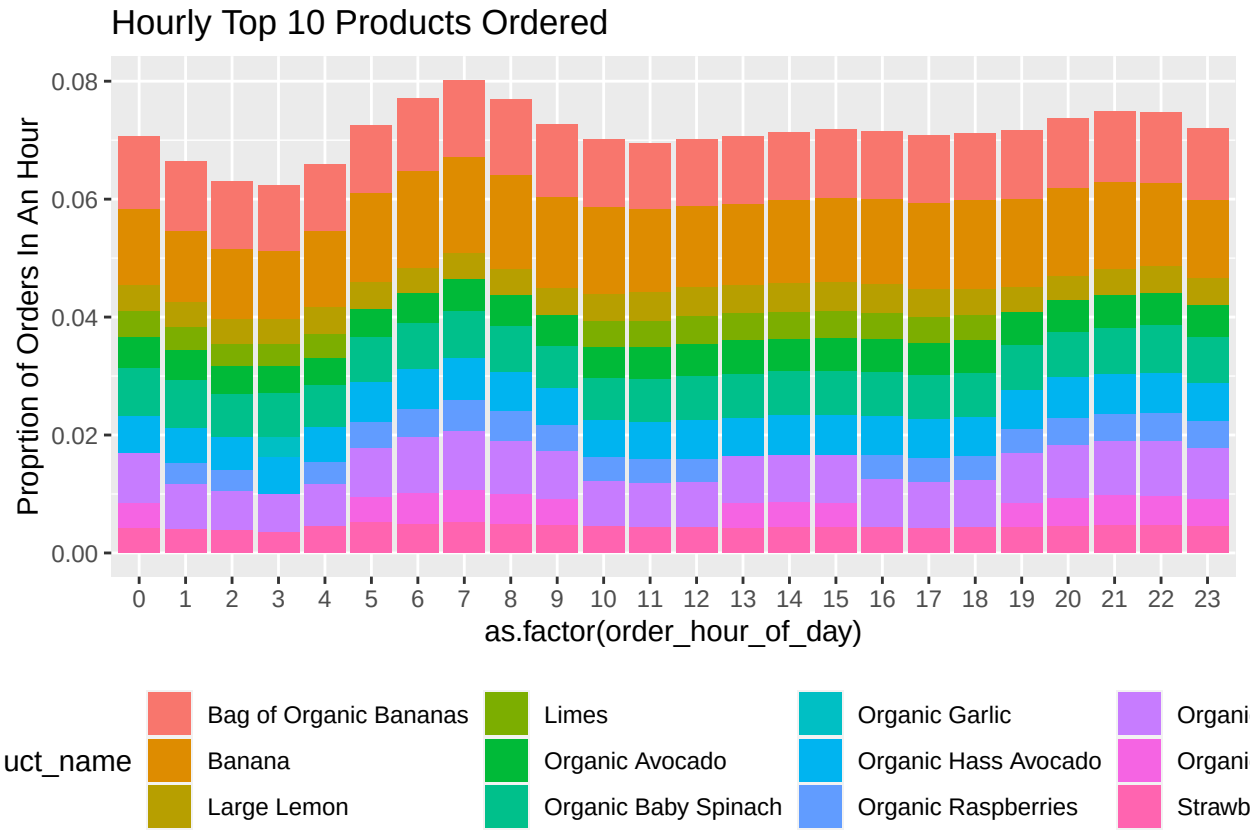
```
## `summarise()` regrouping output by 'order_dow' (override with `.groups` argument)
```


```
order_products_prior %>%
  left_join(orders) %>% left_join(products) %>%
  group_by(order_hour_of_day, product_name) %>%
  summarize(n=n()) %>%
  mutate(percentage=n/sum(n)) %>%
  top_n(10, wt=n) %>%
  ggplot(aes(x=as.factor(order_hour_of_day), y=percentage, fill=product_name)) +
  geom_col() + ylab('Proprtion of Orders In An Hour') +
  ggtitle('Hourly Top 10 Products Ordered') +
  theme(legend.position="bottom",legend.direction="horizontal")
```

```
## Joining, by = "order_id"
```

```
## Joining, by = "product_id"
```

```
## `summarise()` regrouping output by 'order_hour_of_day' (override with `.groups` argument)
```



Visualizing the Product Portfolio

use treemap package to visualize the structure of instacarts product portfolio,

```
tmp4 <- products %>%
  group_by(department_id, aisle_id) %>%
  summarize(n=n()) %>%
  left_join(departments, by="department_id") %>%
  left_join(aisles, by="aisle_id")
```

```
## `summarise()` regrouping output by 'department_id' (override with `.groups` argument)
```

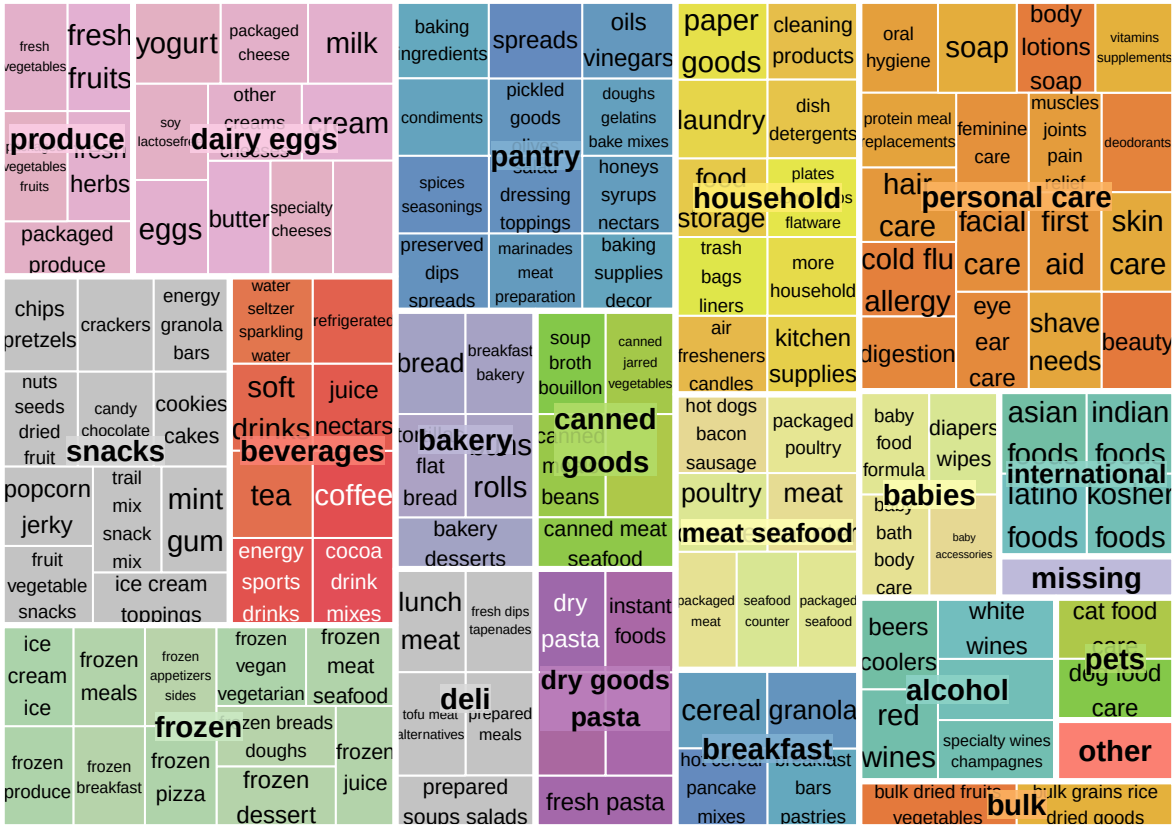
```
tmp5 <-order_products %>%
  group_by(product_id) %>%
  summarize(count=n()) %>%
  left_join(products, by="product_id") %>%
  ungroup() %>%
  group_by(department_id, aisle_id) %>%
  summarize(sumcount = sum(count)) %>%
  left_join(tmp4, by = c("department_id", "aisle_id")) %>%
  mutate(onesize = 1)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## `summarise()` regrouping output by 'department_id' (override with `.groups` argument)
```

Visualize in aisles organized within departments?

```
treemap(tmp5,index=c("department","aisle"), vSize="onesize",vColor="department",
        palette="Set3", title="", sortID="-sumcount", border.col="#FFFFFF")
```

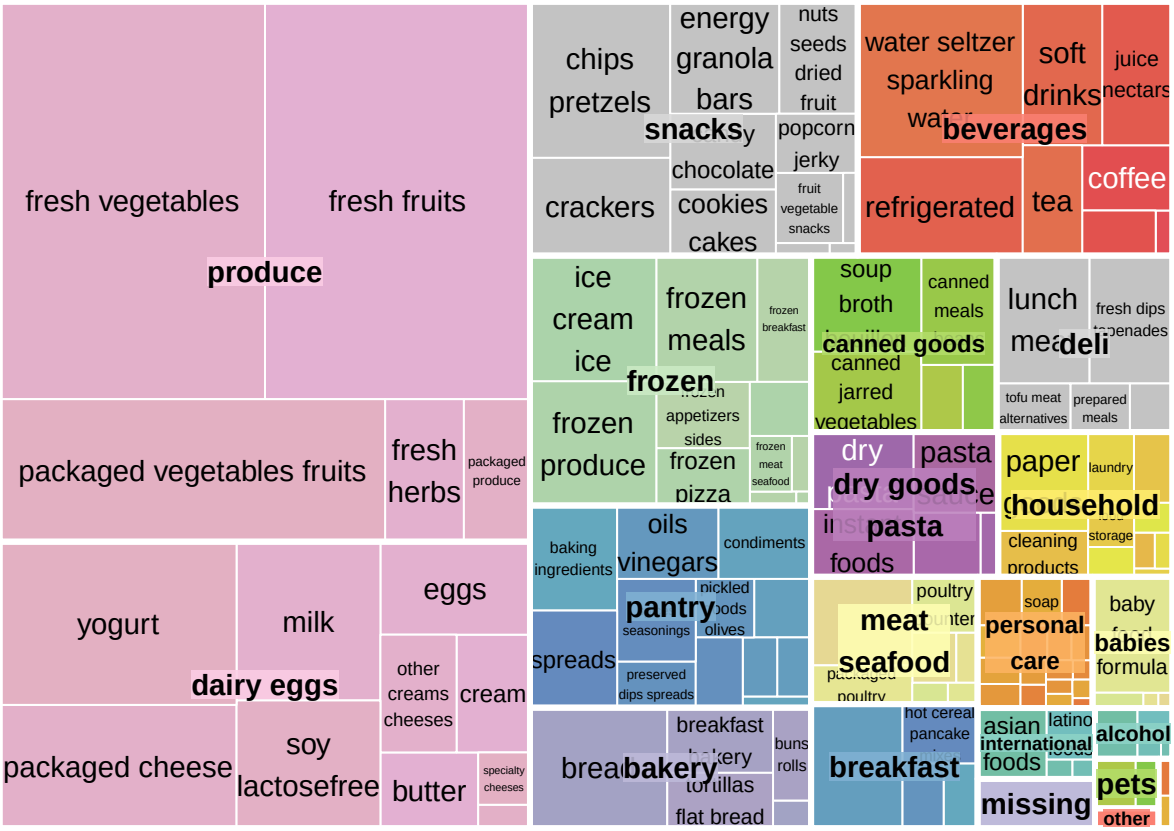


How many unique products are offered in each department/aisle?

```
# The size of the boxes shows the number of products in each category.
treemap(tmp5,index=c("department","aisle"),vSize="n",title="",palette="Set3",border.col="#FFFFFF")
```

How often are products from the department/aisle sold?

```
# The size of the boxes shows the number of sales.
treemap(tmp5,index=c("department","aisle"),vSize="sumcount",title="",palette="Set3",border.col="#FFFFFF
```



Predictive Analysis

only select orders contains >= 4 items

```
order_pro4 <- order_products %>%
  group_by(order_id) %>%
  mutate(n_items = last(add_to_cart_order))
order_pro4 <- order_pro4 %>%
  filter(n_items >= 5 & n_items <=10) # select part of the total data
```

Create a training label, which is 1 or 0, to indicate the actual basket content.

```
data_train = orders %>%
  filter(eval_set=='train') %>%
  inner_join(order_pro4) %>%
  left_join(products) %>%
  mutate(actual = as.integer(1)) %>%   #this is training label
  select(user_id, order_id, product_id, actual)
```

```
## Joining, by = "order_id"

## Joining, by = "product_id"
```

Since the data is too large for R to proceed, sampling 12.5% data

```
# need enough memory to run the matrix
gc()
```

```
##             used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   3098375 165.5    9586736  512.0   9586736  512.0
## Vcells 100670290 768.1  547833828 4179.7 684673536 5223.7
```

```
memory.size()
```

```
## [1] 1016.66
```

```
memory.limit(80000)
```

```
## [1] 80000
```

create the rating matrix

```
ratings_matrix <- data_train %>%
# Select only needed variables
  select(user_id, product_id, actual) %>%
# Spread into user-item format
  spread(product_id, actual, fill = 0) %>%
  select(-user_id) %>%
# Convert to matrix
  as.matrix() %>%
# Convert to recommenderlab class 'binaryRatingsMatrix'
  as("binaryRatingMatrix")
ratings_matrix
```

```
## 47897 x 27846 rating matrix of class 'binaryRatingMatrix' with 350857 ratings.
```

Evaluation Scheme and Model Validation

```
scheme <- ratings_matrix %>%
  evaluationScheme(method = "split",
                   k      = 2,
                   train  = 0.8,
                   given  = -1)
scheme
```

```
## Evaluation scheme using all-but-1 items
## Method: 'split' with 2 run(s).
## Training set proportion: 0.800
## Good ratings: NA
## Data set: 47897 x 27846 rating matrix of class 'binaryRatingMatrix' with 350857 ratings.
```

Set up List of Algorithms

```
algorithms <- list(
  "association rules" = list(name  = "AR",
                        param = list(supp = 0.01, conf = 0.01)),
  "random items"      = list(name  = "RANDOM",  param = NULL),
  "popular items"     = list(name  = "POPULAR", param = NULL),
  # "item-based CF"     = list(name  = "IBCF", param = list(k = 2)),
  # this model takes twice time than "UBCF", not able to run it
  "user-based CF"     = list(name  = "UBCF",
                        param = list(method = "Cosine", nn = 20))
                  )
```

Estimate the Models

```
results <- recommenderlab::evaluate(scheme,
                                    algorithms,
                                    type  = "topNList",
                                    n     = c(3, 5, 10, 15)
                                    )
```

```
## AR run fold/sample [model time/prediction time]
##   1  [0.28sec/121.98sec]
##   2  [0.25sec/118.31sec]
## RANDOM run fold/sample [model time/prediction time]
##   1  [0sec/405.81sec]
##   2  [0sec/393.48sec]
## POPULAR run fold/sample [model time/prediction time]
##   1  [0sec/404.94sec]
##   2  [0sec/424.39sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.02sec/8632.09sec]
##   2  [0sec/8627.24sec]
```

Visualise the Results

arrange the confusion matrix output for one model in a convenient format

```
# Pull into a list all confusion matrix information for one model
tmp <- results$`user-based CF` %>%
  getConfusionMatrix()  %>%
  as.list()
# Calculate average value of 5 cross-validation rounds
  as.data.frame( Reduce("+",tmp) / length(tmp)) %>%
# Add a column to mark the number of recommendations calculated
  mutate(n = c(3, 5, 10, 15)) %>%
# Select only columns needed and sorting out order
  select('n', 'precision', 'recall', 'TPR', 'FPR')
```

```
##    n   precision      recall         TPR          FPR
## 1  3 0.001391867 0.004175365 0.004175365 0.0001075798
## 2  5 0.001377941 0.006889353 0.006889353 0.0001793022
## 3 10 0.001654570 0.016544885 0.016544885 0.0003585051
## 4 15 0.001805939 0.027087683 0.027087683 0.0005376743
```

```
# put the previous steps into a formula
avg_conf_matr <- function(results) {
  tmp <- results %>%
    getConfusionMatrix()  %>%
    as.list()
    as.data.frame(Reduce("+",tmp) / length(tmp)) %>%
    mutate(n = c(3, 5, 10, 15)) %>%
    select('n', 'precision', 'recall', 'TPR', 'FPR')
}
```

use the map() function from the purrr package to get all results in a tidy format, ready for charting.

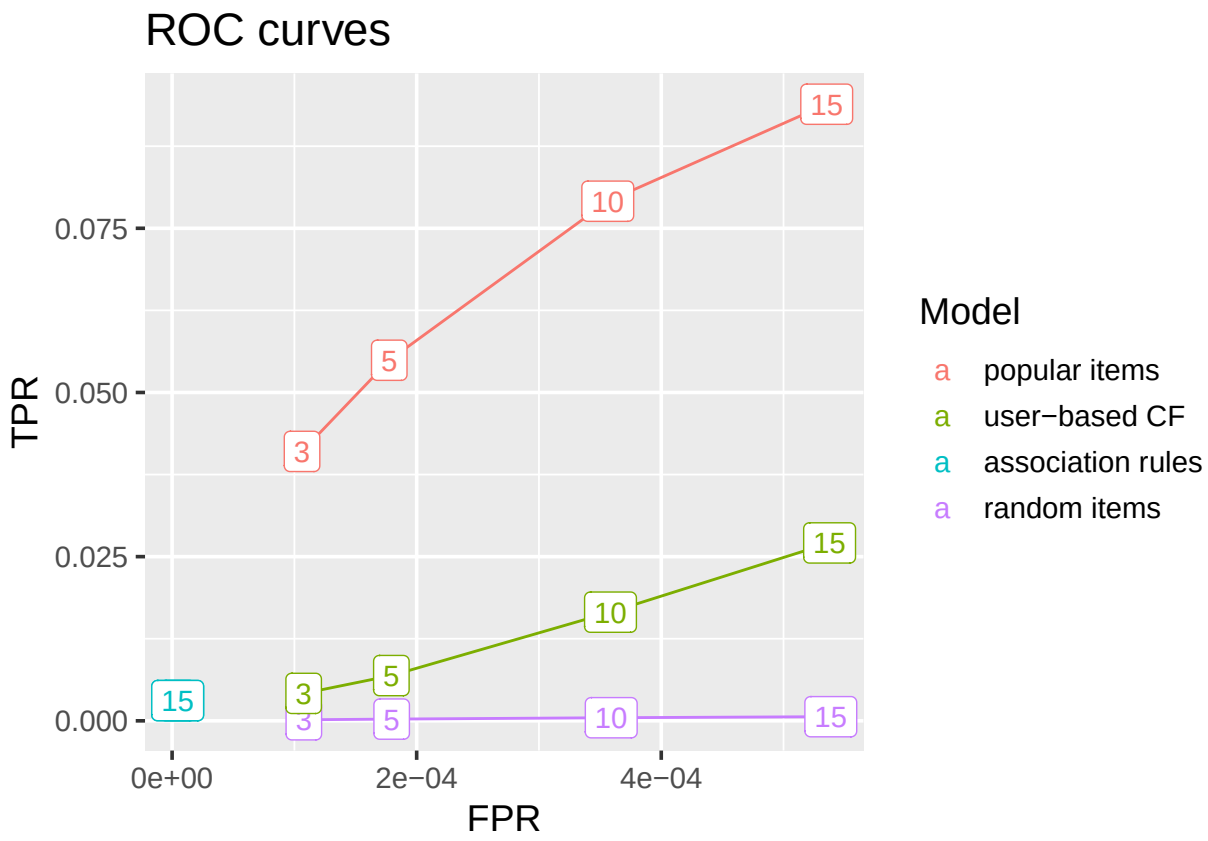
```
# Using map() to iterate function across all models
results_tbl <- results %>%
  map(avg_conf_matr) %>%
# Turning into an unnested tibble
  enframe() %>%
# Unnesting to have all variables on same level
  unnest(cols = c(value))
results_tbl
```

```
## # A tibble: 16 x 6
##    name                 n precision   recall      TPR        FPR
##    <chr>            <dbl>     <dbl>    <dbl>    <dbl>      <dbl>
##  1 association rules     3 0.0253    0.00308  0.00308  0.00000457
##  2 association rules     5 0.0253    0.00308  0.00308  0.00000457
##  3 association rules    10 0.0253    0.00308  0.00308  0.00000457
##  4 association rules    15 0.0253    0.00308  0.00308  0.00000457
##  5 random items          3 0.0000522 0.000157 0.000157 0.000108
##  6 random items          5 0.0000522 0.000261 0.000261 0.000180
##  7 random items         10 0.0000470 0.000470 0.000470 0.000359
##  8 random items         15 0.0000418 0.000626 0.000626 0.000539
##  9 popular items         3 0.0137    0.0410   0.0410   0.000106
## 10 popular items         5 0.0110    0.0549   0.0549   0.000178
## 11 popular items        10 0.00791   0.0791   0.0791   0.000356
## 12 popular items        15 0.00626   0.0939   0.0939   0.000535
## 13 user-based CF         3 0.00139   0.00418  0.00418  0.000108
## 14 user-based CF         5 0.00138   0.00689  0.00689  0.000179
## 15 user-based CF        10 0.00165   0.0165   0.0165   0.000359
## 16 user-based CF        15 0.00181   0.0271   0.0271   0.000538
```

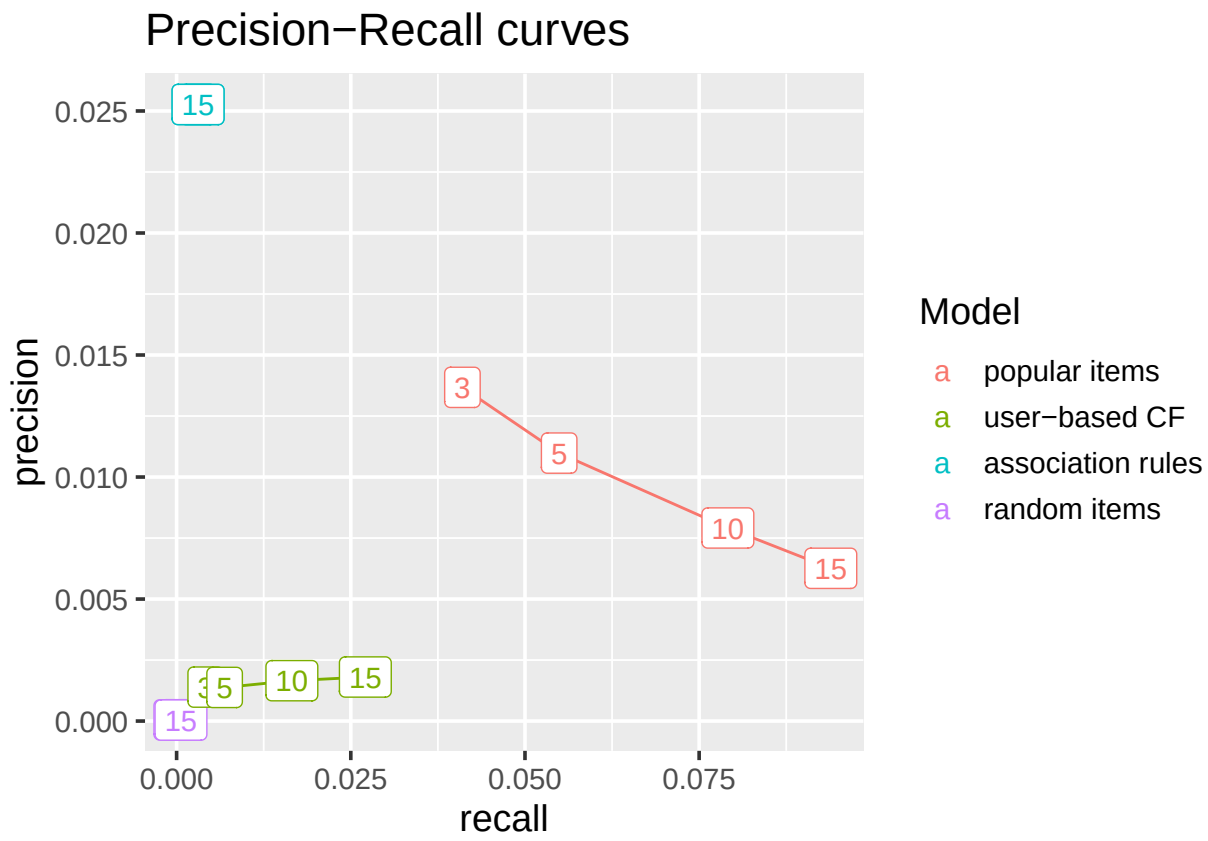
ROC curve

Classification models performance can be compared using the ROC curve

```
results_tbl %>%
  ggplot(aes(FPR, TPR,
             colour = fct_reorder2(as.factor(name),
                       FPR, TPR))) +
  geom_line() +
  geom_label(aes(label = n))  +
  labs(title = "ROC curves", colour = "Model") +
  theme_grey(base_size = 14)
```




```
results_tbl %>%
  ggplot(aes(recall, precision,
             colour = fct_reorder2(as.factor(name),
                       precision, recall))) +
  geom_line() +
  geom_label(aes(label = n))  +
  labs(title = "Precision-Recall curves", colour = "Model") +
  theme_grey(base_size = 14)
```



Prediction for a new user

create a string containing 5 products selected at random

```
set.seed(64)

sample_order <- data_train %>%
  left_join(products) %>%
  select(product_id,product_name) %>%
  sample_n(5) # ramdom pick n items
```

```
## Joining, by = "product_id"
```

```
sample_order
```

```
##    product_id                        product_name
##         <int>                              <fctr>
## 1:      39821 S.O.S Reusable Steel Wool Soap Pads
## 2:      31562                       Sweet Onions
## 3:       6844       Organic Unsulphured Molasses
## 4:      30960          Regular Pork Sausage Tube
## 5:      13914      Cheez-It Baked Snack Crackers
```

```
customer_order <- c(39821, 31562, 6844, 30960, 13914)
```

convert the order in a format that recommenderlab accept

```
gc() # clean the memory
```

```
##             used  (Mb) gc trigger    (Mb)   max used    (Mb)
## Ncells   3259713 174.1    9586737   512.0    9586737   512.0
## Vcells 101541548 774.8 3277776816 25007.5 5334824599 40701.5
```

```
new_order_rat_matrx <- data_train %>%
  select(product_id) %>%
  group_by(product_id) %>%
  unique() %>%
# Add a 'ref' column with 1 or 0 depends on whether it is in co
  mutate(ref = as.numeric(product_id %in% customer_order)) %>%
  spread(product_id, ref) %>%
  as.matrix() %>%
  as("binaryRatingMatrix")
```

create a Recommender by using getData to retrieve training data and set method = "UBCF" to select the best performing model.

```
recomm <- Recommender(getData(scheme, 'train'),
                      method = "UBCF",
                      param = list(k = 5))
```

```
## Warning: Unknown parameter: k
```

```
## Available parameter (with default values):
## method    =  jaccard
## nn    =  25
## weighted  =  TRUE
## sample    =  FALSE
## min_matching_items    =  0
## min_predictive_items  =  0
## verbose   =  FALSE
```

```
recomm
```

```
## Recommender of type 'UBCF' for 'binaryRatingMatrix'
## learned using 38317 users.
```

```
# if use "popular items", the result won't change
```

pass the Recommender and the made-up order to the predict function to create a top 5 recommendation list for the new customer.

```
# need to ensure enough memory before running
pred <- predict(recomm,
                newdata = new_order_rat_matrx,
                n       = 5)
```

the suggested items can be inspected as a list

```
as(pred, 'list')
```

```
## $`1`
## [1] "196"  "890"  "1940" "2876" "4037"
```

convert to product_name to have a better idea for the items

```
result_name <- products %>%
  filter(product_id == "1215" | product_id == "1940" | product_id == "4037" | product_id == "4658" | pr
head(result_name)
```

```
##    product_id                          product_name aisle_id department_id
##         <int>                                <fctr>    <int>         <int>
## 1:       1215 Kidz All Natural Baked Chicken Nuggets      129             1
## 2:       1940            Organic 2% Reduced Fat Milk       84            16
## 3:       4037                  Blackberry Preserves       88            13
## 4:       4658                Imported Mineral Water      115             7
## 5:       4913                  Table Water Crackers       78            19
```