Data Exploratory

Eyal Grinberg & Yam Rozen

2023-08-14

Libraries

```
library(tidyverse)
library(dplyr)
library(ggplot2)
library(tidytext)
library(forcats)
library(tm)
library(stringdist)
library(imager)
```

```
set.seed(37) # for sampling
```

Reading Data

```
data_food_train <- read_csv("data/food_train.csv")
data_nutrients <- read_csv("data/nutrients.csv")
data_food_nutrients <- read_csv("data/food_nutrients.csv")
data_food_test <- read_csv("data/food_test.csv")
```

NA values

```
# After visualizing NA values (see section 1.1 in additional_material) we see that only
# data_food_train and data_food_test contain NA values, let's replace them with "NA".

data_food_train$ingredients <- data_food_train$ingredients %>% replace_na("NA ingredients")
data_food_train$household_serving_fulltext <- data_food_train$household_serving_fulltext %>%
  replace_na("NA household")
data_food_test$ingredients <- data_food_test$ingredients %>%
  replace_na("NA ingredients")
data_food_test$brand <- data_food_test$brand %>% replace_na("NA brand")
```

Food Train Data

We will start by taking a look at each variable, and see if it benefits the model we wand to build.

household_serving_fulltext

```
household_category_data <- data_food_train %>% select(household_serving_fulltext, category)
head(household_category_data)
```

```
## # A tibble: 6 x 2
##   household_serving_fulltext category
##   <chr>                      <chr>
## 1 1 onz                      chocolate
## 2 1 cookie                   cookies_biscuits
## 3 2 cookies                  cookies_biscuits
## 4 5 pieces                   cookies_biscuits
## 5 4 pieces                   chocolate
## 6 3 cookies                  cookies_biscuits
```

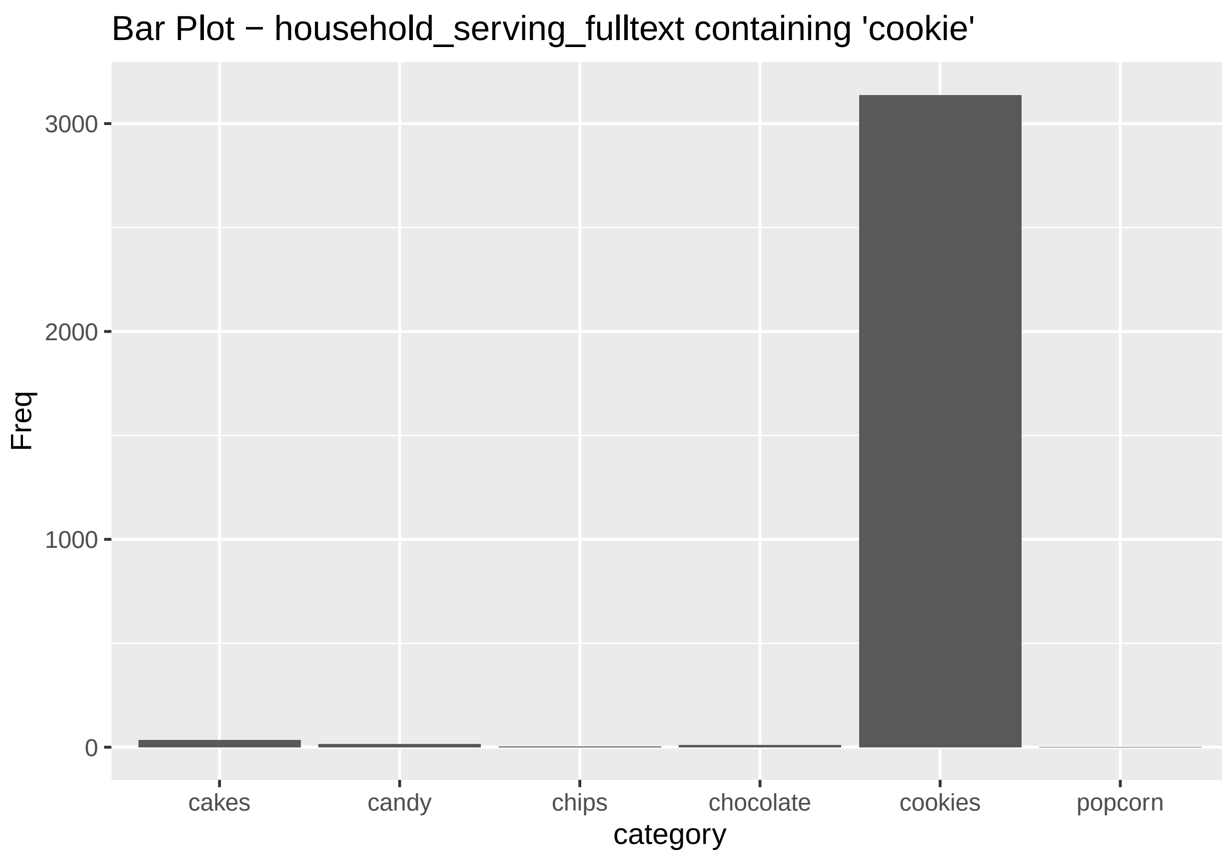
```r
# We can see that there is a serving unit "cookie" which can be very useful for our
# classification task (there is actually a category for it).
# We can see there are similarities in terms of the units of serving size.
# So we want to treat this variable as a categorial variable.

# Let's check how can we model this variable.

unique_household <- unique(data_food_train$household_serving_fulltext)
head(unique_household)
```

```
## [1] "1 onz"     "1 cookie"  "2 cookies" "5 pieces"  "4 pieces"  "3 cookies"
```

```r
# It seems that if we categorize the household_serving_fulltext as it is right now,
# it might be problematic in terms of bias variance trade-off, since the same
# household serving unit has multiple different appearances in different amounts.
# Thus, we want to get a simpler representation.

# Let's take as an example the cookies:

household_category_data$cookie_ind <-
  str_detect(household_category_data$household_serving_fulltext, "cookie")
table_bool_cookies <- table(household_category_data$category,
                            household_category_data$cookie_ind)
table_bool_cookies
```

```
##
##                                      FALSE TRUE
##   cakes_cupcakes_snack_cakes          3751   35
##   candy                               7569   15
##   chips_pretzels_snacks               3676    4
##   chocolate                           3762   10
##   cookies_biscuits                    2146 3138
##   popcorn_peanuts_seeds_related_snacks  7644    1
```

```r
freq_df <- data.frame(table(category = household_category_data$
                              category[household_category_data$cookie_ind]))
ggplot(freq_df, aes(x = category, y = Freq)) +
  geom_bar(stat = "identity") +
  scale_x_discrete(labels = c("cakes", "candy", "chips", "chocolate",
                              "cookies", "popcorn")) +
  labs(title = "Bar Plot - household_serving_fulltext containing 'cookie'")
```




```r
# As we suspected, there is great dominance of the cookies category.

# Let's check randomly a few examples that were marked as 'TRUE'

sample(unique(household_category_data$household_serving_fulltext
              [household_category_data$cookie_ind]), 6)
```

```
## [1] "0.333 cookie"       "14 cookies | about" "30 cookies"
## [4] "0.5 cookies"        "3 cookies"          "2 cookies."
```

```r
# We can see that not only the amounts are different, but also there are expressions
# that contain other words in addition to "cookie".
# We would like to combine those to just "cookie" from the reasons mentioned above.
# Our solution in the model to the mentioned issues would be to lemmatize the expressions.

# We now want to inspect the mistakes, i.e. the FALSE + cookies category,
# and TRUE + other categories.
# let's take as an example the cakes category observations that contained the word "cookie"

sample(household_category_data$household_serving_fulltext[household_category_data$
    cookie_ind & household_category_data$category == "cakes_cupcakes_snack_cakes"] , 6)
```

```
## [1] "1 cookie"     "5 cookies"    "0.125 cookie" "1 cookie"     "1 cookie"
## [6] "2 cookies"
```

```r
# Unfortunately, it seems more or less the same as the household_serving_fulltext
# description of the real cookies category, there's not much we can do about it.

sample(household_category_data$household_serving_fulltext[household_category_data$
    cookie_ind == FALSE & household_category_data$category == "cookies_biscuits"], 6)
```

```
## [1] "14 grm"     "2 biscuits" "1.05 onz"   "1 onz"      "6 pieces"
## [6] "4 waffles"
```

```r
# Not very surprising, we can see cookie's related words like "biscuits" and "waffles".
# so in our model we can check these words also.
```

serving_size_unit

```r
table(data_food_train$serving_size_unit)
```

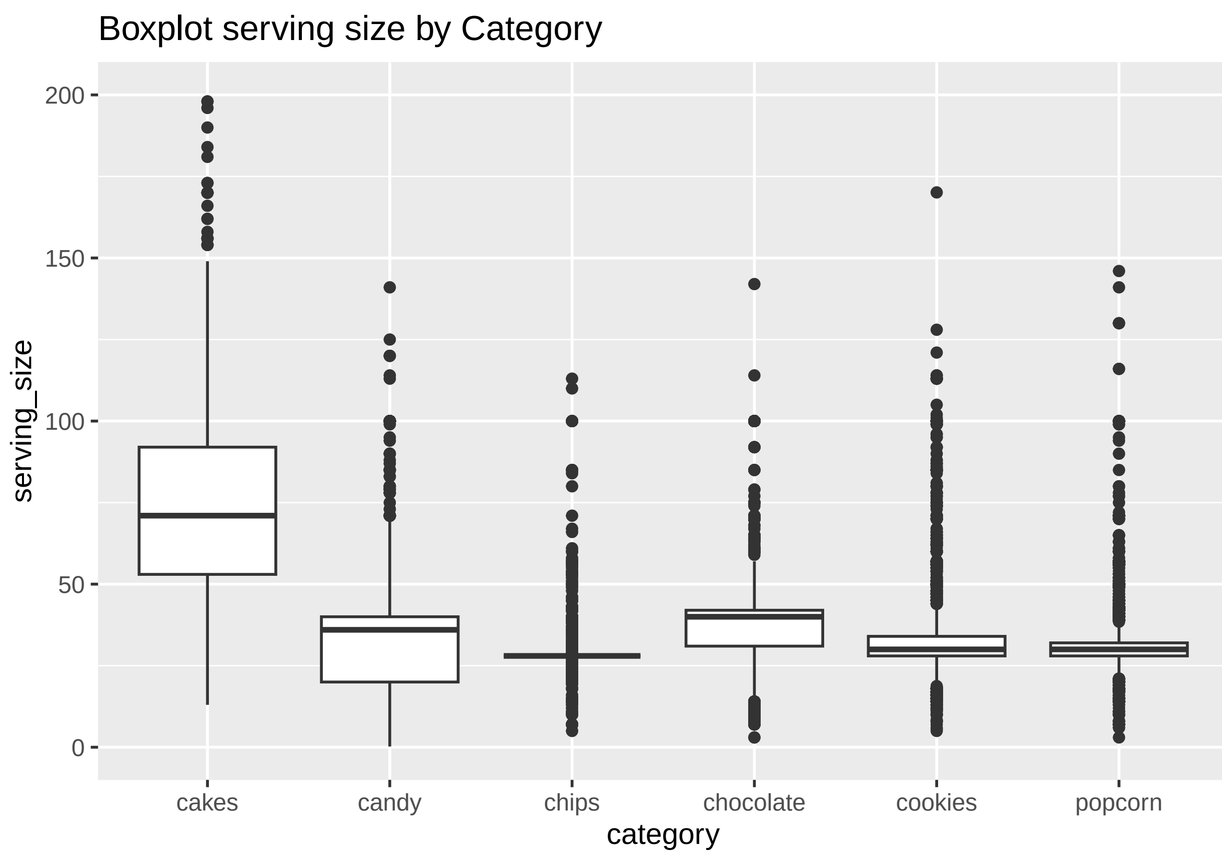
```
##
##     g    ml
## 31743     8
```

```r
# It looks like there's nothing to be done here, we won't use this variable in the model
```

Serving_size

```r
# Boxplot of serving size by category:

ggplot(data_food_train, aes(x = category, y = serving_size)) +
  geom_boxplot() + ylim(0,200) + scale_x_discrete(labels =
            c("cakes", "candy", "chips", "chocolate", "cookies", "popcorn")) +
  labs(title = "Boxplot serving size by Category")
```

```
## Warning: Removed 10 rows containing non-finite values (`stat_boxplot()`).
```




```r
# Let's see clearly the important values

serving_size_grouped_by_cat <- data_food_train %>% group_by(category) %>%
  summarise(mean = mean(serving_size), median = median(serving_size),
            min = min(serving_size), max = max(serving_size), sd = sd(serving_size))
serving_size_grouped_by_cat
```

```
## # A tibble: 6 x 6
##   category                             mean median   min   max    sd
##   <chr>                               <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 cakes_cupcakes_snack_cakes           75.1     71 13      480  32.2
## 2 candy                                32.0     36  0.225  350  14.4
## 3 chips_pretzels_snacks                29.3     28  5      113   6.67
## 4 chocolate                            38.1     40  3      246  13.5
## 5 cookies_biscuits                     33.0     30  5      170. 13.1
## 6 popcorn_peanuts_seeds_related_snacks 31.7     30  3      146   7.92
```

```r
# This leads us to the following insights:
# 1. the serving size of cakes is much bigger.
# 2. candies can be served in very small sizes, unlike the other categories
# (which makes sense according to our knowledge about candies).
# 3. the variance of cakes is much bigger than the other categories.
# Overall it looks like a good feature to use in our model.
```

Description

```r
# This is a textual variable, we first start by cleaning the strings.

text_clean <- function(x)
{
  x  =  gsub("<.*?>", " ", x)
  x  =  iconv(x, "latin1", "ASCII", sub="")
  x  =  gsub("[^[:alnum:]]", " ", x)
  x  =  tolower(x)
  x  =  stripWhitespace(x)
  x  =  gsub("^\\s+|\\s+$", "", x)
  return(x)
}
```

```r
data_food_train$description <- text_clean(data_food_train$description)
data_food_train$ingredients <- text_clean(data_food_train$ingredients)
data_food_train$brand <- text_clean(data_food_train$brand)
data_food_train$household_serving_fulltext <-
  text_clean(data_food_train$household_serving_fulltext)
```

```r
# Now we want to tokenize the descriptions into words.

data_train_tokenized_description <- data_food_train %>%
  unnest_tokens(word, description) %>% count(category, word, sort = TRUE)
head(data_train_tokenized_description, 7)
```

```
## # A tibble: 7 x 3
##   category                             word          n
##   <chr>                                <chr>     <int>
## 1 cookies_biscuits                     cookies    3155
## 2 chocolate                            chocolate  3142
## 3 candy                                candy      2825
## 4 chips_pretzels_snacks                chips      2516
## 5 chips_pretzels_snacks                potato     1387
## 6 popcorn_peanuts_seeds_related_snacks roasted    1334
## 7 cookies_biscuits                     chocolate  1295
```

```r
# As shown, the word "chocolate" appears in the chocolate category,
# but also in the cookies_biscuits category.

cookies_with_word_chocolate <- data_food_train %>% subset(category ==
      "cookies_biscuits") %>% select(description) %>%
      filter(str_detect(description, "chocolate"))
sample(cookies_with_word_chocolate$description, 4)
```

```
## [1] "thin crispy chocolate sandwich cookies"
## [2] "spartan middleo s chocolate sandwich cookies double filled mint"
## [3] "rite aid pantry chocolate chip cookies"
## [4] "michel et augustin milk chocolate and grains"
```

```r
# An important insight from the sample, can be that the phrase
# 'chocolate chip' is relevant for the cookies category.
```

```r
# These results led us to try tokenizing based on 2 words

data_train_tokenized_description_2_tokens <- data_food_train %>%
  unnest_tokens(bigram, description, token = "ngrams", n = 2) %>%
  count(category, bigram, sort = TRUE)
head(data_train_tokenized_description_2_tokens)
```

```
## # A tibble: 6 x 3
##   category                           bigram              n
##   <chr>                              <chr>           <int>
## 1 chips_pretzels_snacks              potato chips     1170
## 2 chocolate                          milk chocolate   1042
## 3 chocolate                          dark chocolate   1014
## 4 popcorn_peanuts_seeds_related_snacks trail mix        710
## 5 chips_pretzels_snacks              tortilla chips    609
## 6 cookies_biscuits                   chocolate chip    462
```

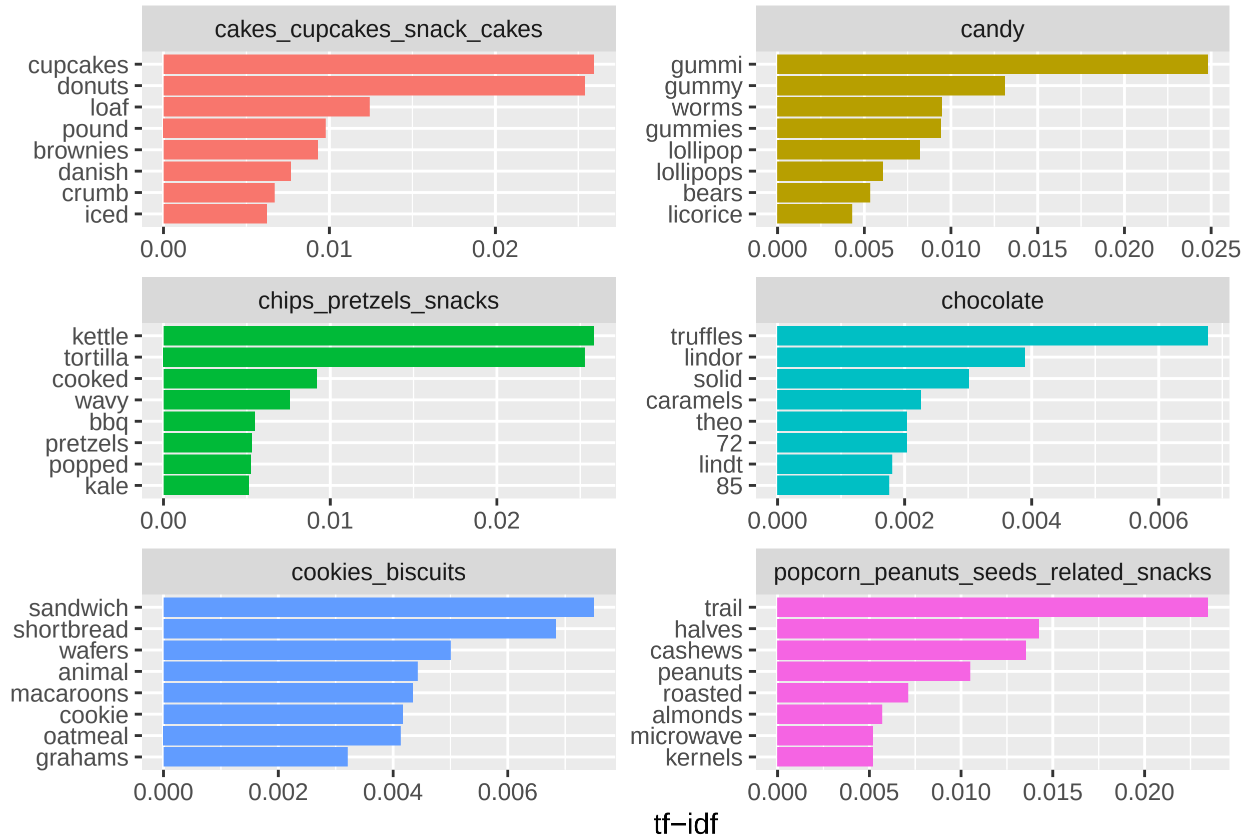
```r
# Indeed when we tokenize for 2 word there's less overlap between
# the different categories.
# for our phrase 'chocolate chip' - it appears in 462 of the
# cookies_biscuits observations.
```

```r
# Let's try to find another method for finding the "unique" words that
# separate between the different categories.
# for this purpose, we used the NLP tool TF-IDF.

data_train_tokenized_description <- data_train_tokenized_description %>%
  group_by(category) %>% mutate(total = sum(n)) %>% bind_tf_idf(word, category, n)
head(data_train_tokenized_description)
```

```
## # A tibble: 6 x 7
## # Groups:   category [5]
##   category                           word          n total     tf   idf  tf_idf
##   <chr>                              <chr>     <int> <int>  <dbl> <dbl>   <dbl>
## 1 cookies_biscuits                   cookies    3155 24331 0.130  0     0
## 2 chocolate                          chocola~   3142 19337 0.162  0     0
## 3 candy                              candy      2825 33870 0.0834 0     0
## 4 chips_pretzels_snacks              chips      2516 18213 0.138  0     0
## 5 chips_pretzels_snacks              potato     1387 18213 0.0762 0     0
## 6 popcorn_peanuts_seeds_related_snacks roasted   1334 34110 0.0391 0.182 0.00713
```

```r
# Explanation:
# tf = n / total
# idf(term) = ln(num of categories(6) / num of categories that term appears in)
# tf_idf = tf * idf

# Visualization

data_train_tokenized_description <- data_train_tokenized_description %>%
  select(-total) %>% arrange(desc(tf_idf))
data_train_tokenized_description %>%
  group_by(category) %>% slice_max(tf_idf, n = 8) %>% ungroup() %>%
```

```r
ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = category)) +
  geom_col(show.legend = FALSE) + facet_wrap(~category, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```




```r
# The following plots present the unique words for each category (in terms of highest tf-idf values)

# Insights:
# 1. We still have some "meaningwize duplicates", for example in the candy
# category we have: "gummi", "gummy", "gummies".
# 2. numbers are related to the chocolate category, which makes sense because
# in many chocolate snacks the description states the percentage of cocoa.
# 3. the word "chocolate" does not appear in any of the categories above, as mentioned
# before, this word is problematic because it has multiple appearances also in
# categories which are not chocolate. Hence, it's not considered as a unique word that
# differentiate between the different categories.

# We will consider using the unique words based on the tf-idf score in our model.
```

Ingredients

```r
# This is also a textual variable, therefore we used the same tools for analysis.

data_train_tokenized_ingredients <- data_food_train %>%
  unnest_tokens(word, ingredients) %>% count(category, word, sort = TRUE)
head(data_train_tokenized_ingredients)
```

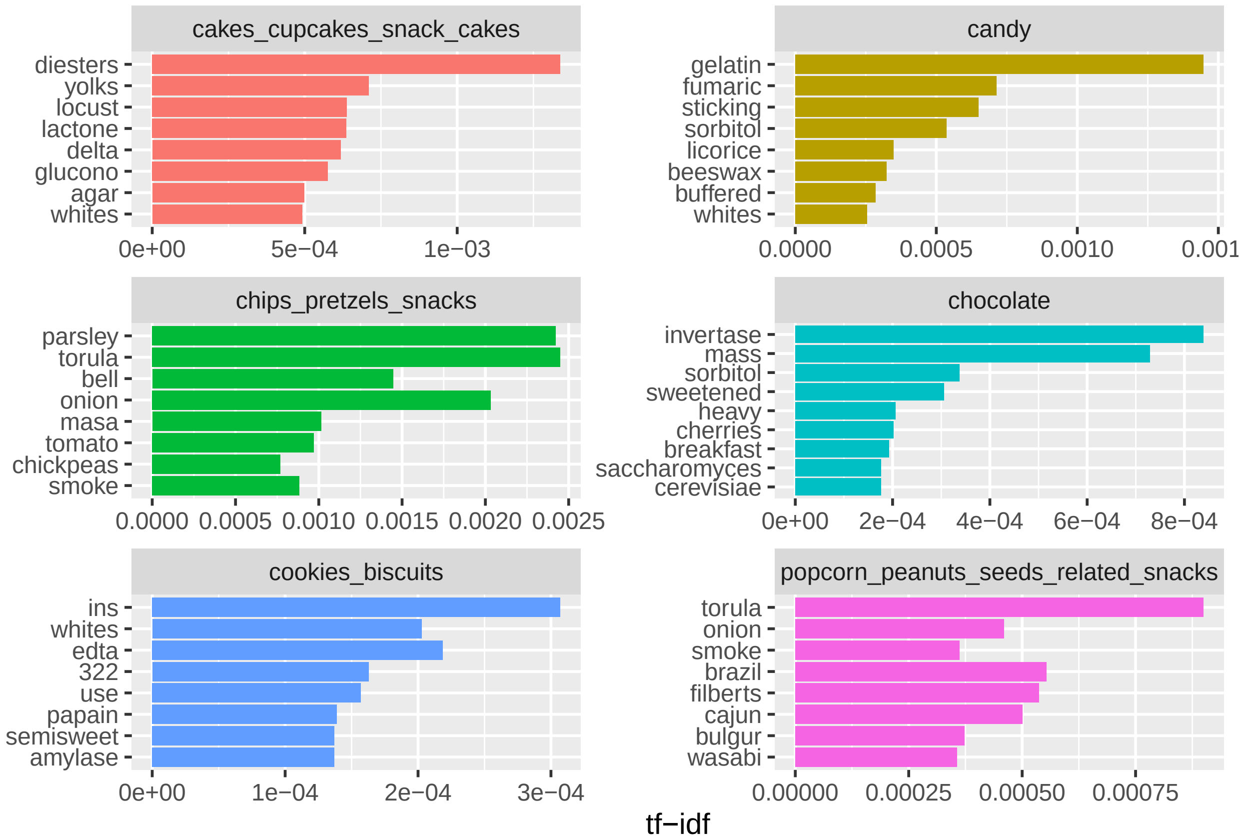
```
## # A tibble: 6 x 3
##   category                           word      n
##   <chr>                              <chr> <int>
## 1 cakes_cupcakes_snack_cakes         and   11004
## 2 cakes_cupcakes_snack_cakes         oil   10557
## 3 cookies_biscuits                   flour 10425
## 4 popcorn_peanuts_seeds_related_snacks oil    9788
## 5 cakes_cupcakes_snack_cakes         flour  9737
## 6 cookies_biscuits                   sugar  9643
```

```r
# as expected, there are common ingredients (such as oil) in different categories,
# and also the word "and" appears multiple times which is not good.
```

```r
# After failing to acheive any useful insights when tokenizing based on more than
# one word (see section 1.4 in the additional_material), we used again the tf-idf.

data_train_tokenized_ingredients <- data_train_tokenized_ingredients %>%
  group_by(category) %>% mutate(total = sum(n)) %>% bind_tf_idf(word, category, n)
head(data_train_tokenized_ingredients)
```

```
## # A tibble: 6 x 7
## # Groups:   category [3]
##   category                           word      n  total     tf   idf tf_idf
##   <chr>                              <chr> <int>  <int>  <dbl> <dbl>  <dbl>
## 1 cakes_cupcakes_snack_cakes         and   11004 385760 0.0285     0      0
## 2 cakes_cupcakes_snack_cakes         oil   10557 385760 0.0274     0      0
## 3 cookies_biscuits                   flour 10425 296895 0.0351     0      0
## 4 popcorn_peanuts_seeds_related_snacks oil    9788 206374 0.0474     0      0
## 5 cakes_cupcakes_snack_cakes         flour  9737 385760 0.0252     0      0
## 6 cookies_biscuits                   sugar  9643 296895 0.0325     0      0
```

```r
# Visualization

data_train_tokenized_ingredients <- data_train_tokenized_ingredients %>%
  select(-total) %>% arrange(desc(tf_idf))
data_train_tokenized_ingredients %>%
  group_by(category) %>% slice_max(tf_idf, n = 8) %>% ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = category)) +
  geom_col(show.legend = FALSE) + facet_wrap(~category, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```




```r
# As shown above, most tf-idf values are relatively small in comparison to the
# description analysis.
# The reason for it is that the tf values are small because there are much more
# ingredients in each category (in comparison to the description).
# Also, there are many tf-idf values that are exactly 0, that is because their
# idf value is 0 (ln(6/6)) which means the specific ingredient appears in every
# category (such as oil) and thus it is not a useful ingredient in terms of prediction.
```

Brand

```r
# We would like to use brand as a categorial variable, let's check how many
# unique brands are there.

length(unique(data_food_train$brand)) # 4714
```

```
## [1] 4714
```

```r
# We can see that there are many different brands, we will try to reduce that number.

appearances_df <- as.data.frame(table(data_food_train$brand))
sum(appearances_df$Freq <= 5) # 3861
```

```
## [1] 3861
```

```r
# It seems that many of the brands have a small number of appearances,
# from considerations of bias variance trade-off we will merge them into one category.

# note: We tried using more advanced tools such as similarities/distances matrices
# between strings, but eventually we decided to not use them in a model because we think
# there is a greater predictive power in the other textual variables.
# most of the frequent brands are retail stores or big companies that are likely
# to sell items from all the 6 categories.
# also there are 467 snacks that are not even branded.

sorted_appearances_df <- appearances_df[order(-appearances_df$Freq), ]
head(sorted_appearances_df)
```

```
##                       Var1 Freq
## 4475   wal mart stores inc  579
## 4069         target stores  540
## 1320 ferrara candy company  506
## 2971    not a branded item  467
## 2694            meijer inc  463
## 945       cvs pharmacy inc  342
```

Data nutrients + Data food nutrients

These two data sets are used together because they both provide nutrients data about the observations where the connecting link is the 'nutrient_id'.

So we decided to merge them into one data set.

```r
merged_df_nutrients <- merge(data_food_nutrients, data_nutrients,
                             by = "nutrient_id", all.x = TRUE) %>% arrange(idx)

# We also can drop the nutrient_id variable because we have the 'name' variable

merged_df_nutrients <- merged_df_nutrients[-1]
glimpse(merged_df_nutrients)
```

```
## Rows: 493,054
## Columns: 4
## $ idx       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, ~
## $ amount    <dbl> 7.14, 35.71, 53.57, 536.00, 3.60, 143.00, 5.14, 89.00, 0.00,~
## $ name      <chr> "Protein", "Total lipid (fat)", "Carbohydrate, by difference~
## $ unit_name <chr> "G", "G", "G", "KCAL", "G", "MG", "MG", "MG", "IU", "MG", "M~
```

```r
# We wanted to add the categories to the data,but since the nutrients data
# contains test observations, we decided to add both train and test categories.

data_food_test$category <- "unknown"
data_food <- rbind(data_food_train, data_food_test)
merged_df_nutrients <- merge(merged_df_nutrients, data_food[, c(1,8)],
                             by = "idx", all.x = TRUE)
```

```r
# Grouping by 'category' and 'name', then adding the mean of 'amount'

df_nutrients_mean_by_cat <- merged_df_nutrients %>%
  group_by(category, name) %>% mutate(mean_amount = mean(amount))
head(df_nutrients_mean_by_cat)
```

```
## # A tibble: 6 x 6
## # Groups:   category, name [6]
##     idx amount name                       unit_name category  mean_amount
##   <dbl>  <dbl> <chr>                      <chr>     <chr>           <dbl>
## 1     1   7.14 Protein                    G         chocolate        5.93
## 2     1  35.7  Total lipid (fat)          G         chocolate       31.8
## 3     1  53.6  Carbohydrate, by difference G         chocolate       55.8
## 4     1 536    Energy                     KCAL      chocolate      516.
## 5     1   3.6  Fiber, total dietary       G         chocolate        4.56
## 6     1 143    Calcium, Ca                MG        chocolate       98.9
```

```r
# The data is combined for all the categories, so we splitted it by the categories

nuts_splitted_by_cat <- split(merged_df_nutrients , merged_df_nutrients$category)

# creating a df for each category

cakes_nutrients <- nuts_splitted_by_cat$cakes_cupcakes_snack_cakes
choco_nutrients <- nuts_splitted_by_cat$chocolate
popcorn_nutrients <- nuts_splitted_by_cat$popcorn_peanuts_seeds_related_snacks
candy_nutrients <- nuts_splitted_by_cat$candy
chips_nutrients <- nuts_splitted_by_cat$chips_pretzels_snacks
cookies_nutrients <- nuts_splitted_by_cat$cookies_biscuits
test_nutrients <- nuts_splitted_by_cat$unknown
head(cakes_nutrients)
```

```
##     idx amount                        name unit_name                   category
## 121   9   4.69                     Protein         G cakes_cupcakes_snack_cakes
## 122   9  12.50           Total lipid (fat)         G cakes_cupcakes_snack_cakes
## 123   9  42.19 Carbohydrate, by difference         G cakes_cupcakes_snack_cakes
## 124   9 297.00                      Energy      KCAL cakes_cupcakes_snack_cakes
## 125   9   1.60        Fiber, total dietary         G cakes_cupcakes_snack_cakes
## 126   9  20.00        Total sugar alcohols         G cakes_cupcakes_snack_cakes
```

```r
cakes_mean_by_nut <- cakes_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
choco_mean_by_nut <- choco_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
popcorn_mean_by_nut <- popcorn_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
candy_mean_by_nut <- candy_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
chips_mean_by_nut <- chips_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
cookies_mean_by_nut <- cookies_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
```

```r
test_mean_by_nut <- test_nutrients %>%
  group_by(name) %>% reframe(mean_amount = mean(amount))
head(cakes_mean_by_nut)
```

```
## # A tibble: 6 x 2
##   name                                  mean_amount
##   <chr>                                       <dbl>
## 1 Calcium, Ca                                 46.1
## 2 Carbohydrate, by difference                 51.7
## 3 Carbohydrate, other                         19.6
## 4 Cholesterol                                 99.8
## 5 Energy                                     378.
## 6 Fatty acids, total monounsaturated           5.21
```

```r
# let's see how many different nutrients are in each category

df_list <- list(cakes_mean_by_nut, candy_mean_by_nut, popcorn_mean_by_nut,
  choco_mean_by_nut, chips_mean_by_nut, cookies_mean_by_nut, test_mean_by_nut)
num_diff_nutrients_each_cat <- map_dbl(df_list, nrow)
min(num_diff_nutrients_each_cat) # 37
```

```
## [1] 37
```

```r
max(num_diff_nutrients_each_cat) # 43
```

```
## [1] 43
```

```r
# what about all the observations together?

length(unique(c(cakes_mean_by_nut$name, choco_mean_by_nut$name,
    popcorn_mean_by_nut$name, candy_mean_by_nut$name,chips_mean_by_nut$name,
    cookies_mean_by_nut$name, test_mean_by_nut$name))) # 47
```

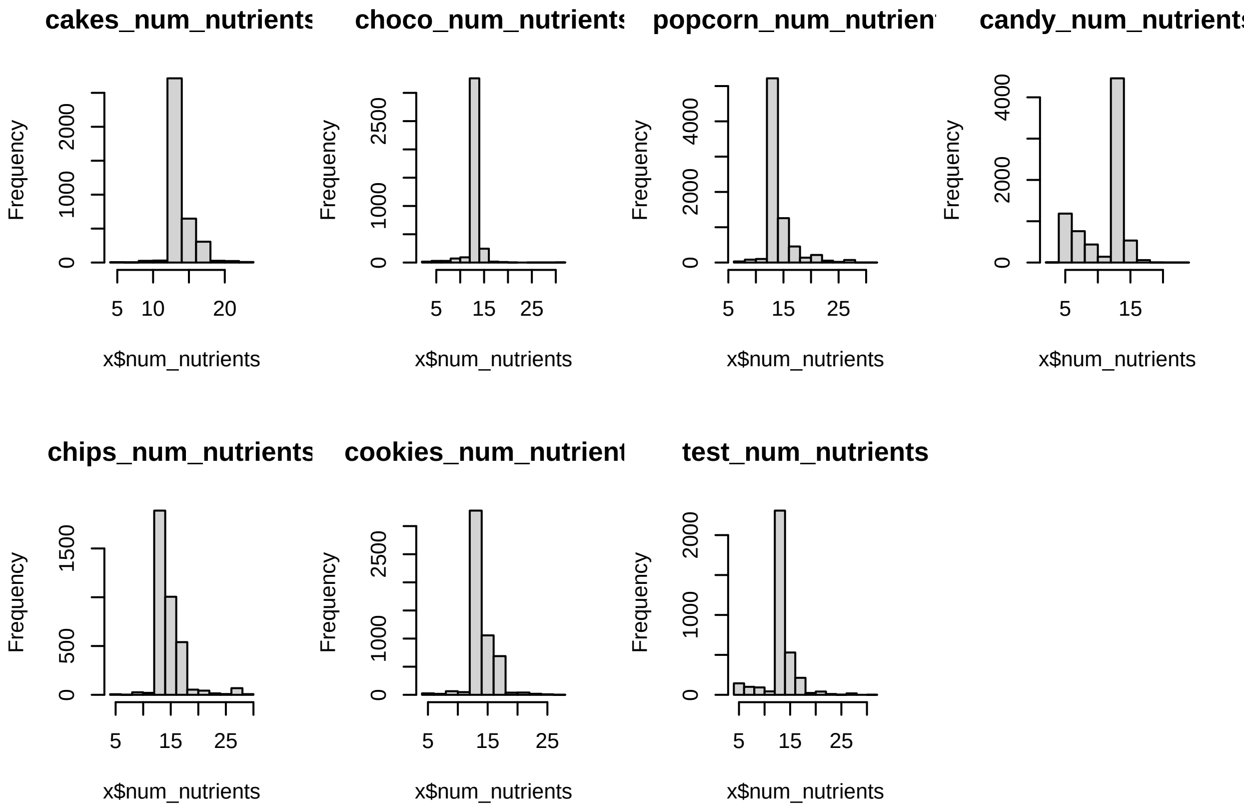
```
## [1] 47
```

```r
# Interesting, there are only 47 different nutrients in all the observations
# together (the original data contains 235 nutrients).
# Since there is a big overlap between the categories in terms of the number of
# different nutrients, it makes sense to compare the nutrients between the categories.

# Let's see how many nutrients each observation contains (for each category)

cakes_num_nutrients <- cakes_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
choco_num_nutrients <- choco_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
popcorn_num_nutrients <- popcorn_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
candy_num_nutrients <- candy_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
```

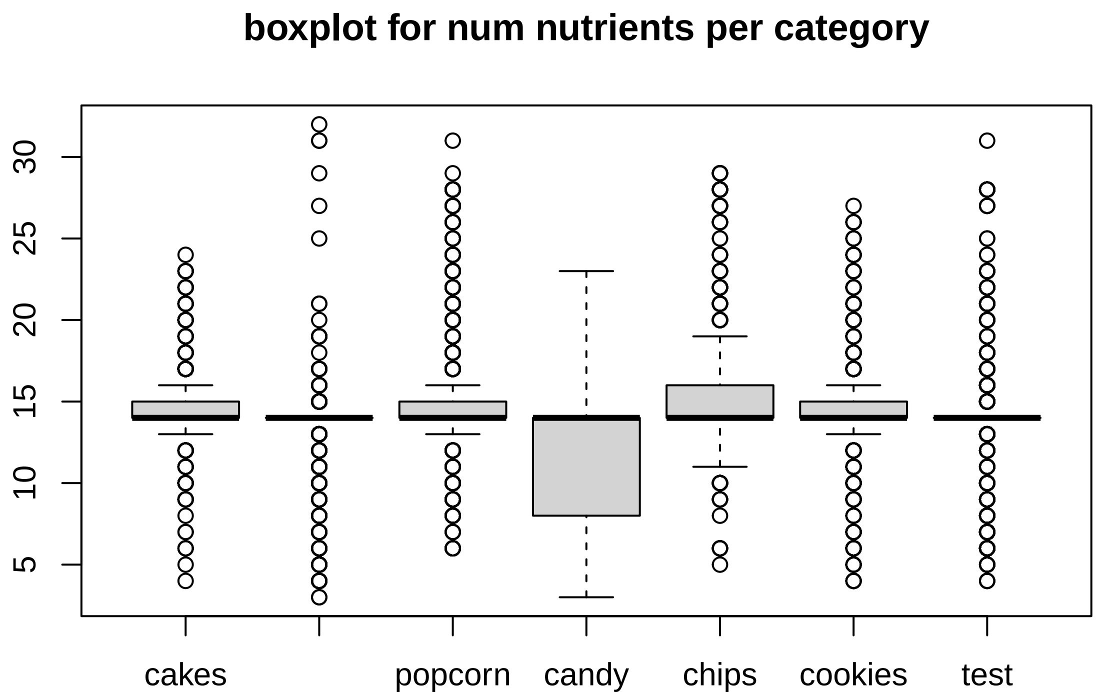
```r
chips_num_nutrients <- chips_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
cookies_num_nutrients <- cookies_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
test_num_nutrients <- test_nutrients %>%
  group_by(idx) %>% reframe(num_nutrients = length(name))
head(cakes_num_nutrients)
```

```
## # A tibble: 6 x 2
##     idx num_nutrients
##   <dbl>         <int>
## 1     9            15
## 2    12            14
## 3    14            14
## 4    22            14
## 5    49            14
## 6    70            17
```

```r
# num nutrients distribution for each category

plots_func_num_nutriens_by_cat <- function(x, title_name) {
  hist(x$num_nutrients, main = title_name)
}

par(mfrow = c(2,4))
plots_func_num_nutriens_by_cat(cakes_num_nutrients, "cakes_num_nutrients")
plots_func_num_nutriens_by_cat(choco_num_nutrients, "choco_num_nutrients")
plots_func_num_nutriens_by_cat(popcorn_num_nutrients, "popcorn_num_nutrients")
plots_func_num_nutriens_by_cat(candy_num_nutrients, "candy_num_nutrients")
plots_func_num_nutriens_by_cat(chips_num_nutrients, "chips_num_nutrients")
plots_func_num_nutriens_by_cat(cookies_num_nutrients, "cookies_num_nutrients")
plots_func_num_nutriens_by_cat(test_num_nutrients, "test_num_nutrients")
```



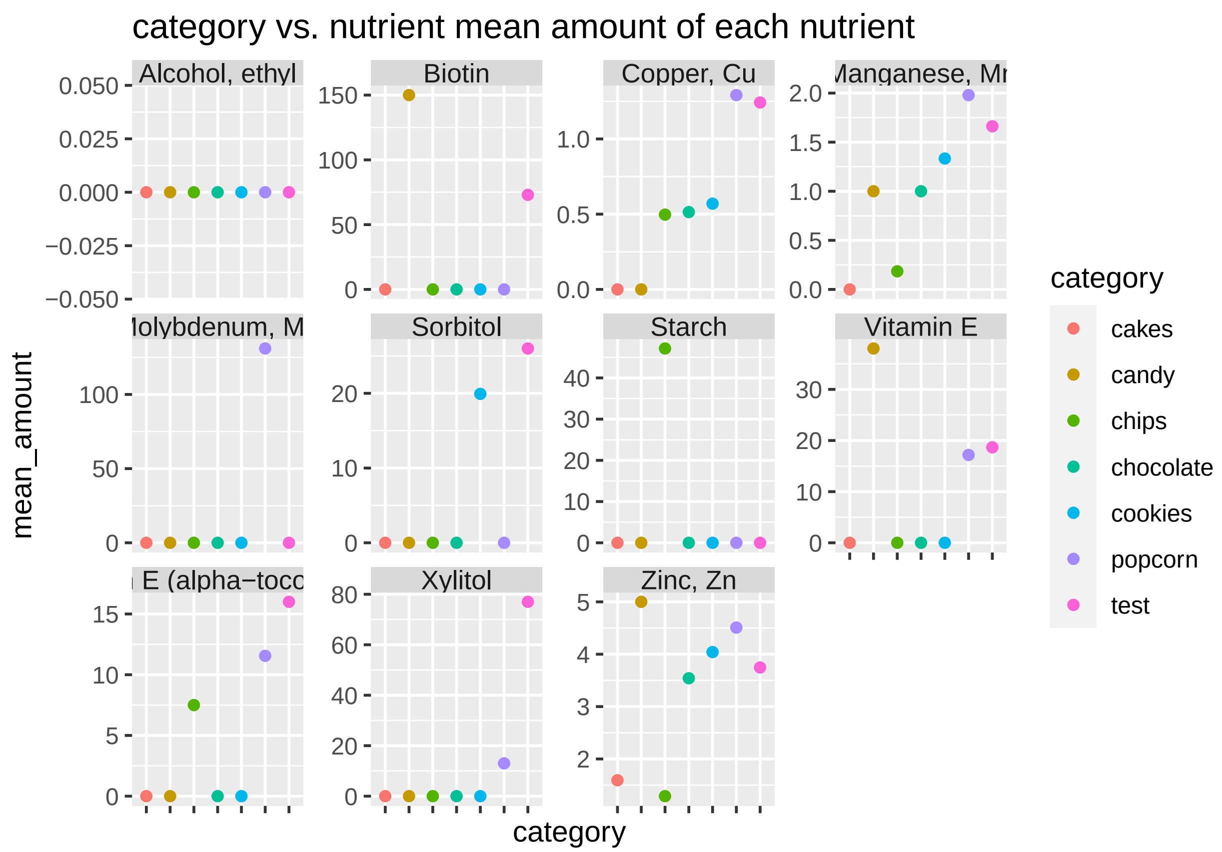

```r
# the distributions look pretty much similar, maybe except the candy category.
# let's use boxplots

boxplot(cakes_num_nutrients$num_nutrients, choco_num_nutrients$num_nutrients,
        popcorn_num_nutrients$num_nutrients, candy_num_nutrients$num_nutrients,
        chips_num_nutrients$num_nutrients, cookies_num_nutrients$num_nutrients,
        test_num_nutrients$num_nutrients,
        main = "boxplot for num nutrients per category", names =
          c("cakes", "chocolate", "popcorn", "candy", "chips", "cookies", "test"))
```




```r
# Indeed, there are quite a few candy observations that contain a 'low'
# number of nutrients.
```

```r
# We now want to create a df of the mean amount of the nutrients for each category.

nutrients_mean_amount_with_zero_amounts <- df_list %>% reduce(full_join, by = "name")
colnames(nutrients_mean_amount_with_zero_amounts) <-
  c("nutrient", "cakes", "candy", "popcorn", "chocolate", "chips", "cookies", "test")

# replace NAs with 0
nutrients_mean_amount_with_zero_amounts[is.na(
  nutrients_mean_amount_with_zero_amounts)] <- 0

# Visualization

ggplot_nutrients <- function(pivoted_df, start, end) {
  ggplot(data = pivoted_df[start:end,], mapping = aes(x = category, y = mean_amount,
  color = category)) + geom_point() + facet_wrap(. ~ nutrient, scales = "free_y") +
  labs(title = "category vs. nutrient mean amount of each nutrient",
  x = "category", y ="mean_amount") + theme(strip.text.x = element_text(
  size = 10, margin = margin()),axis.text.x.bottom = element_blank(),
  strip.text.y = element_text(size = 20, margin = margin()))
}

# We wanted to create a plot for each one of the 47 nutrients and also keeping it clear,
# that's the solution we chose.
```

```r
nuts_pivoted <- nutrients_mean_amount_with_zero_amounts %>%
  pivot_longer(!nutrient, names_to = "category", values_to = "mean_amount")

# We didn't want to exceed 20 pages in the PDF so we plot just once.
# see the rest of the plots in section 2.1 in the additional_material, they are very colorful
# ggplot_nutrients(nuts_pivoted, 1, 63)
# ggplot_nutrients(nuts_pivoted, 64, 126)
# ggplot_nutrients(nuts_pivoted, 127, 189)
# ggplot_nutrients(nuts_pivoted, 190, 252)
ggplot_nutrients(nuts_pivoted, 253, 329)
```

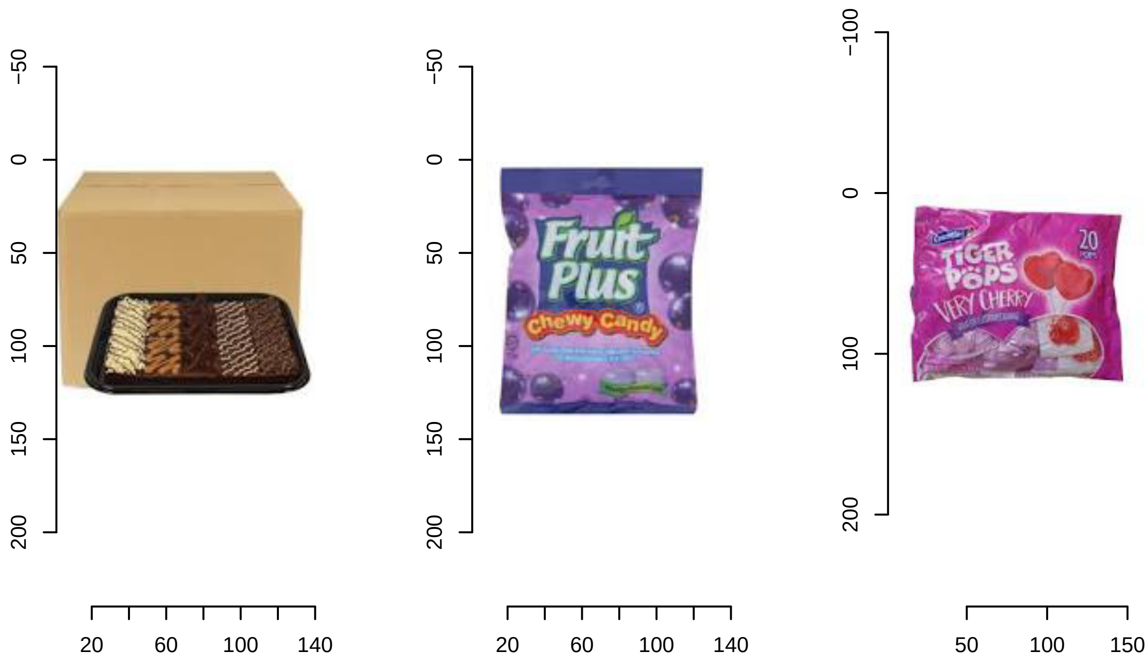



```r
# from the plots we can see that there are a few nutrients that are highly related
# to a certain category, for example for the nutrient Starch, it appears only in the
# chips category, which makes sense based on our basic knowledge.
# we will use each one of these 47 nutrients as a feature in the model.
```

Images data

```r
# train images
folder_list <- list.files("data/train/")
folder_path <- paste0("data/train/", folder_list, "/")
file_name <- map(folder_path, function(x) paste0(x, list.files(x))) %>% unlist()
```

```r
#Let's see some images
sample_image <- sample(file_name, 3)
img <- map(sample_image, load.image)
par(mfrow = c(1, 3))
map(img, plot)
```




```
## [[1]]
## Image. Width: 140 pix Height: 140 pix Depth: 1 Colour channels: 3
##
## [[2]]
## Image. Width: 140 pix Height: 140 pix Depth: 1 Colour channels: 3
##
## [[3]]
## Image. Width: 162 pix Height: 121 pix Depth: 1 Colour channels: 3
```

```r
# Dimensions

get_dim <- function(x){
  img <- load.image(x)
  df_img <- data.frame(height = height(img), width = width(img), filename = x)
  return(df_img)
}

sample_file <- sample(file_name, 100)
```

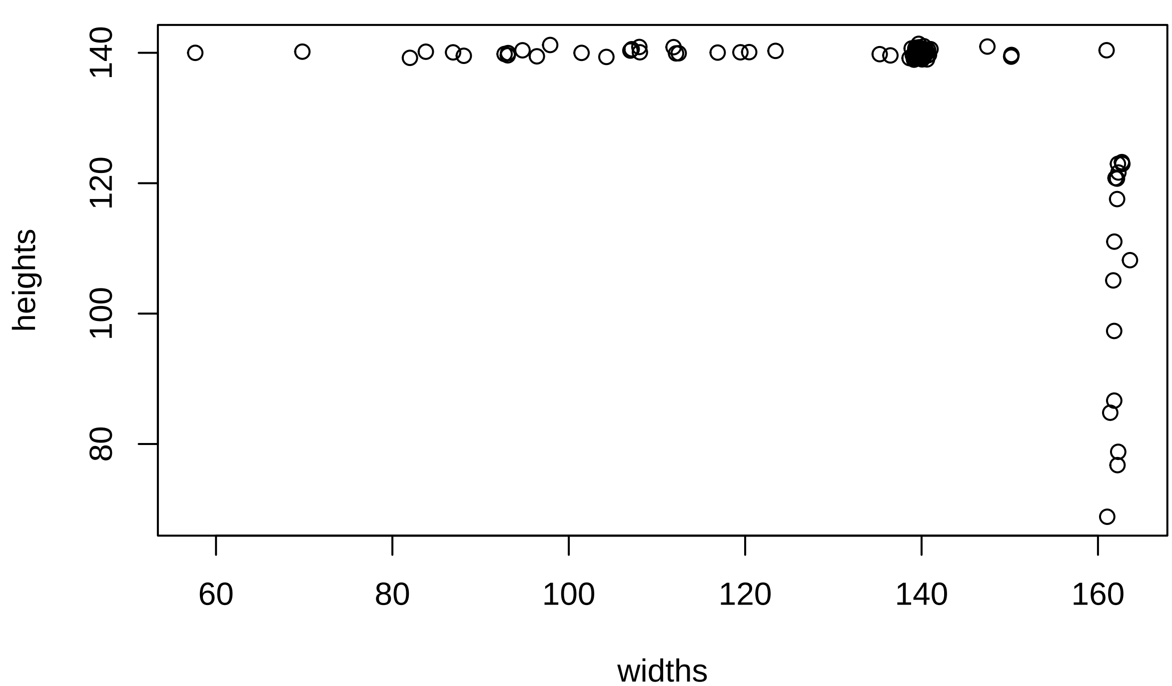
```r
file_dim <- map_df(sample_file, get_dim)
head(file_dim , 3)
```

```
##   height width                                       filename
## 1    140   140 data/train/cakes_cupcakes_snack_cakes/25862.jpg
## 2    140    87                  data/train/chocolate/10949.jpg
## 3    140    97                  data/train/chocolate/16369.jpg
```

```r
summary(file_dim)
```

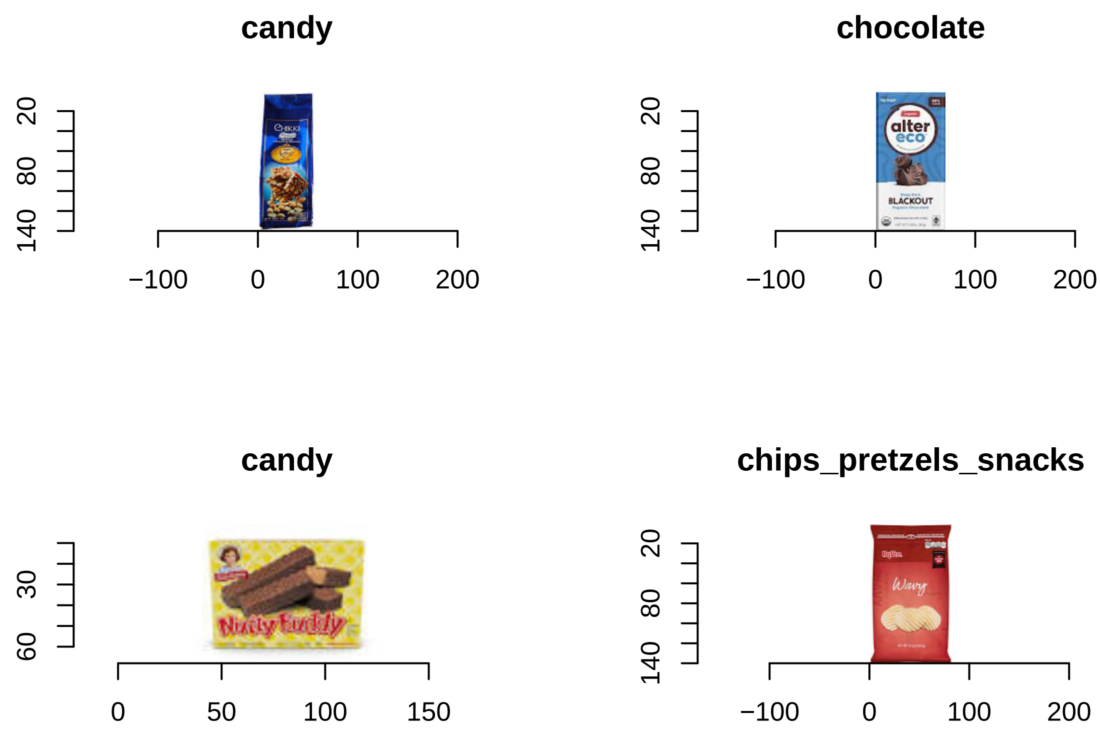
```
##      height          width          filename
##  Min.   : 68.0   Min.   : 57.0   Length:100
##  1st Qu.:140.0   1st Qu.:132.2   Class :character
##  Median :140.0   Median :140.0   Mode  :character
##  Mean   :134.2   Mean   :133.8
##  3rd Qu.:140.0   3rd Qu.:140.0
##  Max.   :140.0   Max.   :162.0
```

```r
# It looks like most images are of size 140X140, let's make a plot.
# We added normal noise so we would be able to distinguish between points with
# the exact same dimensions.

heights <- file_dim$height + rnorm(100, 0, 0.5)
widths <- file_dim$width + rnorm(100, 0, 0.5)
plot(widths, heights)
```



```r
# Indeed, we can see there is a huge black "point" at (140,140).
```

```r
# We wanted to take a look at some of the 'smaller' images (in terms of the number of pixels).

file_dim$num_pixels_in_img <- file_dim$height * file_dim$width
file_dim <- file_dim %>% arrange(num_pixels_in_img)
img_name <- file_dim$filename[1:4]
titles <- strsplit(img_name, split = "/") %>% unlist()
img <- map(img_name, load.image)
par(mfrow = c(2,2))
for (i in 1:4) {
  plot(img[[i]], main = paste(titles[4*i - 1]))
}
```




```r
# It seems like these images are not so bad, we will consider using them in the model.
```

THE END .