Data Analysis and Visualisation Project

Shraddha Lanka

Thursday, May 05, 2016

Contents

Data Analysis and Visualisation Project, HS-616.

Creating a new column category.

The following changes were made to the starter script. We select food_code to build the required dataframe.

```
#install.packages("tidyr")

setwd("F:/MSHI/616/DataAnaviz")
options(Warn=-1)

data_dir <- "FNDDS_2011"

fortification <- c(`0`="none", `1`="fortified_product", `2`="contains
fortified ingredients")

fndds_tables <- list(
  AddFoodDesc = list(
            title="Additional Food Descriptions",
            column_types=c(
                food_code="integer", # foreign key
                seq_num="integer",
                start_date="date",
                end_date="date",
                additional_food_description="text"),
            sep="^"
        ),
    FNDDSNutVal = list(
            title="FNDDS Nutrient Values",
            column_types=c(
                food_code="integer",
                nutrient_code="integer",    # Nutrient Descriptions table
                start_date="date",
                end_date="date",
                nutrient_value="double"
                ),
            sep="^"
        ),
    FNDDSSRLinks = list(
            title="FNDDS-SR Links", # see p34 of fndds_2011_2012_doc.pdf
            column_types=c(
                food_code="integer",
                start_date="date",
                end_date="date",
                seq_num="integer",
                sr_code="integer",
```

```
                sr_descripton="text",
                amount="double",
                measure="char[3]",  # lb, oz, g, mg, cup, Tsp, qt, fluid
ounce, etc
                portion_code="integer",
                retention_code="integer",
                flag="integer",
                weight="double",
                change_type_to_sr_code="char[1]",   # D=data change; F=food
change
                change_type_to_weight="char[1]",
                change_type_to_retn_code="char[1]"
                ),
            sep="^"
        ),
    FoodPortionDesc = list(
            title="Food Portion Descriptions",
            column_types=c(
                portion_code="integer",     # foreign key
                start_date="date",
                end_date="date",
                portion_description="text",
                change_type="char[1]"
            ),
            sep="^"
        ),
    FoodSubcodeLinks = list(
            title="Food code-subcode links",
            column_types=c(
                food_code="integer",
                subcode="integer",
                start_date="date",
                end_date="date"
                ),
            sep="^"
        ),
    FoodWeights = list(
            title="Food Weights",
            column_types=c(
                food_code="integer",    # foreign key
                subcode="integer",
                seq_num="integer",
                portion_code="integer", # food portion description id
                start_date="date",
                end_date="date",
                portion_weight="double",    # missing values = -9
                change_type="char[1]"   # D=data change, F=food change
                ),
            sep="^"
        ),
```

```
    MainFoodDesc = list(
            title="Main Food Descriptions",
            column_types=c(
                food_code="integer",
                start_date="date",
                end_date="date",
                main_food_description="character",
                fortification_id="integer"),
            sep="^"
        ),
    ModDesc = list(
            title="Modifications Descriptons",
            column_types=c(
                modification_code="integer",
                start_date="date",
                end_date="date",
                modification_description="text",
                food_code="integer"

                ),
            sep="^"
        ),
    ModNutVal = list(
            title="Modifications Nutrient Values",
            column_types=c(
                modification_code="integer",
                nutrient_code="integer",
                start_date="date",
                end_date="date",
                nutrient_value="double"
                ),
            sep="^"
        ),
    MoistNFatAdjust = list(
            title="Moisture & Fat Adjustments", # to account for changes
during cooking
            column_types=c(
                food_code="integer",
                start_date="date",
                end_date="date",
                moisture_change="double",
                fat_change="double",
                type_of_fat="integer"   # SR code or food code
                ),
            sep="^"
        ),
    NutDesc = list(
            title="Nutrient Descriptions",
            column_types=c(
                nutrient_code="integer",
```

```
                nutrient_description="text",
                tagname="text",
                unit="text",
                decimals="integer"  # decimal places
                ),
            sep="^"
        ),
    SubcodeDesc = list(
            title="Subcode Descriptions",
            column_types=c(
                subcode="integer",  # key; 0=use default gram weights
                start_date="date",
                end_date="date",
                subcode_description="text"
                ),
            sep="^"
        )
)

# flat file to a data frame: called by fndds2sqlite for each table
assign_data_frame <- function(tbl_name){
    tbl <- read.table(
        file.path(data_dir, paste0(tbl_name, ".txt")),
        sep="^",
        quote="~",
        stringsAsFactors=FALSE)
    # drop last (empty) column
    tbl <- tbl[1:(length(tbl)-1)]
    # gets names of columns from tbl_name element of fndds_tables list of
list
    names(tbl) <- names(fndds_tables[[tbl_name]][["column_types"]])
    # assigns the data frame tbl to global variable named by string contents
of tbl_name
    assign(tbl_name, tbl, envir = .GlobalEnv)
}

# flat file to database
#fndds2sqlite <- function(data_dir, table_details, sqlite_filename){

    #library("RSQLite")
  #open database named by sqlite_filename, create empty if doesn't exist
    #con <- dbConnect(SQLite(), sqlite_filename)

    #for (tbl_name in names(table_details)){
    #   file_name <- paste0(tbl_name, ".txt") # paste0 has empty string as
separator
    #   assign_data_frame(tbl_name)
    #   print(file_name)
    #   tbl <- get(tbl_name)
```

```
    #   print(tbl_name)
        # function exits with error message next line if database already
exists
    #   dbWriteTable(con, tbl_name, tbl, row.names = FALSE)
    #}

    #dbDisconnect(con)# seems to auto save the updated database
#}

#First time run creates the database from flat files and saves to database
file
# first run also creates dataframes for each table
#If you run whwn database already exists you get:
# Error: Table AddFoodDesc exists in database, and both overwrite and append
are FALSE
# and you get only one data frame, but can go on, no harm done
#fndds2sqlite("FNDDS_2011", fndds_tables, "fndds.sqlite")

library(DBI)

## Warning: package 'DBI' was built under R version 3.1.3

# Creates 3 of the data frames (useful if database already exists so data
frames not created)
for (tbl in c("FNDDSNutVal", "MainFoodDesc", "NutDesc"))
    assign_data_frame(tbl)

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.1.3

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)

## Warning: package 'tidyr' was built under R version 3.1.3

# Make a simplified selection of foods, store in data frame.
# TO DO: have MainFoodDesc be a tbl sourced from SQLite.
get_selected_foods <- function(){
    # Pull out all "Not Further Specified" foods as a wide selection of
reasonably generic items.
  # NB: grepl returns boolean for each string in vector: TRUE if a string
```

```
contains the pattern, otherwise FALSE
    generics <- MainFoodDesc %>%
        filter( grepl(", NFS", main_food_description )) %>%
        filter(!grepl("infant formula", main_food_description, ignore.case =
TRUE ) )

    # Raw fruits
    # Berries are covered by "Berries, raw, NFS" and "Berries, frozen, NFS"
    # NB: grepl can search for pattern specified by regular expressions:
http://www.rexegg.com/regex-quickstart.html
    # NB: with regular expressions '^' matches empty string at beginning of
line
    #  food codes for fruits begin with 6
    fruits <- MainFoodDesc %>%
        filter( grepl("^6", food_code) ) %>%
        filter( grepl("^([^,\\(]+), raw$", main_food_description) ) %>%
        filter( !grepl("berries", main_food_description) )

    # Raw vegetables
    # Potatoes are covered by "White potato, NFS", "Sweet potato, NFS", etc.
    # NB: food codes for vegetables begin with 7
    vegetables <- MainFoodDesc %>%
        filter( grepl("^7", food_code) ) %>%
        filter(!grepl("potato", main_food_description)) %>%
        filter( grepl(", raw$", main_food_description))

    # 4="legumes, nuts, and seeds"
    # NB: food codes for legumes, nuts, and seeds begin with 4
    nuts_and_seeds <- MainFoodDesc %>%
        filter( grepl("^4", food_code) ) %>%
        mutate( firstWord = strsplit(main_food_description, " ")[[1]][1] )

    # Selected alcoholic beverages
    # All alcoholic beverages: grepl("^93", food_code))
    # "Cocktail, NFS" already gives us "Cocktail"
    # NB: food codes for alcoholic beverages begin with 93
    alcoholic_beverages <- MainFoodDesc %>%
        filter( main_food_description %in% c("Beer", "Wine, table, red",
"Wine, table, white",
            "Whiskey", "Gin", "Rum", "Vodka") )

    # Collect them all into one table
    rbind(generics, fruits, vegetables, alcoholic_beverages) %>%
        select( food_code, main_food_description, fortification_id )  %>%
        filter( nchar(main_food_description) < 20 ) %>%
      # gsub(pattern, replacement, string_for_Pattern_matching)
      # replace pattern with empty string (remove pattern)
        mutate( main_food_description = gsub("(, NFS|, raw)", "",
main_food_description) )
```

```
}

foods <- get_selected_foods()   # 163 items


library(sqldf)

## Warning: package 'sqldf' was built under R version 3.1.3

## Loading required package: gsubfn

## Warning: package 'gsubfn' was built under R version 3.1.3

## Loading required package: proto

## Warning: package 'proto' was built under R version 3.1.3

## Loading required package: RSQLite

## Warning: package 'RSQLite' was built under R version 3.1.3

long_food_nutrients <- sqldf("SELECT f.food_code, nd.nutrient_description,
nv.nutrient_value
  FROM foods f
    INNER JOIN FNDDSNutVal nv ON f.food_code = nv.food_code
    INNER JOIN NutDesc nd ON nv.nutrient_code = nd.nutrient_code")

## Loading required package: tcltk

nutrient_food_df <- spread(long_food_nutrients, food_code, nutrient_value,
fill=0)

food_nutrient_mat <- t(as.matrix(nutrient_food_df[-1]))
#food_nutrient_mat1<-t(as.matrix(nutrient_food_df))
colnames(food_nutrient_mat) <- nutrient_food_df$nutrient_description
```

Once this is done, the following code creates the category vector which is appended to the food matrix.

```
food_code<-rownames(food_nutrient_mat)
food_code<-as.integer(food_code)
category={}

for (i in 1:length(food_code))
{
  #print(i)
  if(food_code[i]<20000000)
    category[i]="dairy"
  else if(food_code[i]<30000000)
    category[i]="meat/fish"
  else if(food_code[i]<40000000)
    category[i]="eggs"
```

```
  else if(food_code[i]<50000000)
    category[i]="legumes,nuts,seeds"
  else if(food_code[i]<60000000)
    category[i]="grains"
  else if(food_code[i]<70000000)
    category[i]="fruits"
  else if(food_code[i]<80000000)
    category[i]="vegetables"
  else if(food_code[i]<90000000)
    category[i]="fats"
  else if (food_code[i]<93000000)
    category[i]="sugars"
  else if (food_code[i]<94000000)
    category[i]="alcohol"
  else if(food_code[i]>=94000000)
    category[i]="protien powder"
}


food_nutrient_mat<-cbind(food_nutrient_mat,category)

food_nutrient_df<-as.data.frame(food_nutrient_mat,stringsAsFactors=FALSE)
```

The number of observations in each row can be dispayed using the following code snippet. The output is shown below.

```
as.data.frame(table(food_nutrient_df$category))
```

```
##                  Var1 Freq
## 1             alcohol    8
## 2               dairy    4
## 3                fats    4
## 4              fruits   37
## 5              grains   28
## 6  legumes,nuts,seeds    6
## 7           meat/fish   16
## 8      protien powder    1
## 9              sugars    6
## 10         vegetables   53
```

As we now have the data frame, we will proceed with some more data wrangling:

```
#Converting all columns to integers

food_nutrient_int<-mapply(food_nutrient_df[-66],FUN = as.double)

food_nutrient_int<-as.data.frame(food_nutrient_int)

food_nutrient_int$Category<-category
```

```
#SUMMARY OF THE DATA

summary(food_nutrient_int)

##       10:0             12:0             14:0             16:0        
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0200  
##  Median :0.0000   Median :0.0000   Median :0.0020   Median : 0.0590  
##  Mean   :0.0328   Mean   :0.0448   Mean   :0.1213   Mean   : 0.9205  
##  3rd Qu.:0.0000   3rd Qu.:0.0020   3rd Qu.:0.0180   3rd Qu.: 0.6540  
##  Max.   :2.5290   Max.   :2.5870   Max.   :7.4360   Max.   :21.6970  
##       16:1              18:0             18:1              18:2        
##  Min.   :0.00000   Min.   :0.0000   Min.   : 0.0000   Min.   : 0.0000  
##  1st Qu.:0.00000   1st Qu.:0.0020   1st Qu.: 0.0155   1st Qu.: 0.0355  
##  Median :0.00200   Median :0.0080   Median : 0.0750   Median : 0.1030  
##  Mean   :0.05771   Mean   :0.4203   Mean   : 2.3506   Mean   : 1.3496  
##  3rd Qu.:0.01700   3rd Qu.:0.1815   3rd Qu.: 1.0295   3rd Qu.: 0.4890  
##  Max.   :0.96100   Max.   :9.9990   Max.   :39.6590   Max.   :37.5720  
##       18:3             18:4              20:1              20:4        
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.00000   Min.   :0.000000  
##  1st Qu.:0.0050   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.000000  
##  Median :0.0200   Median :0.000000   Median :0.00000   Median :0.000000  
##  Mean   :0.1259   Mean   :0.000135   Mean   :0.02306   Mean   :0.004074  
##  3rd Qu.:0.0680   3rd Qu.:0.000000   3rd Qu.:0.00900   3rd Qu.:0.000000  
##  Max.   :5.7060   Max.   :0.009000   Max.   :0.65600   Max.   :0.134000  
##     20:5 n-3             22:1            22:5 n-3         
##  Min.   :0.0000000   Min.   :0.00000   Min.   :0.000000  
##  1st Qu.:0.0000000   1st Qu.:0.00000   1st Qu.:0.000000  
##  Median :0.0000000   Median :0.00000   Median :0.000000  
##  Mean   :0.0005583   Mean   :0.00238   Mean   :0.000638  
##  3rd Qu.:0.0000000   3rd Qu.:0.00000   3rd Qu.:0.000000  
##  Max.   :0.0340000   Max.   :0.09100   Max.   :0.021000  
##     22:6 n-3              4:0               6:0               8:0         
##  Min.   :0.0000000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.0000000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.0000000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.0004601   Mean   :0.03851   Mean   :0.02349   Mean   :0.01617  
##  3rd Qu.:0.0000000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :0.0270000   Max.   :3.22600   Max.   :2.00700   Max.   :1.19000  
##     Alcohol          Caffeine          Calcium         Carbohydrate    
##  Min.   : 0.000   Min.   : 0.0000   Min.   :  0.00   Min.   : 0.00  
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:  9.00   1st Qu.: 4.91  
##  Median : 0.000   Median : 0.0000   Median : 22.00   Median : 9.58  
##  Mean   : 1.056   Mean   : 0.1595   Mean   : 54.43   Mean   :19.65  
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 51.00   3rd Qu.:19.95  
##  Max.   :37.900   Max.   :11.0000   Max.   :950.00   Max.   :99.98  
##  Carotene, alpha   Carotene, beta     Cholesterol      Choline, total  
##  Min.   :   0.00   Min.   :    0.0   Min.   :  0.000   Min.   :  0.00  
##  1st Qu.:   0.00   1st Qu.:    0.0   1st Qu.:  0.000   1st Qu.:  6.05  
##  Median :   0.00   Median :   29.0   Median :  0.000   Median :  9.80
```

```
##  Mean   :  54.98   Mean   :  523.9   Mean   :  8.853   Mean   : 18.66
##  3rd Qu.:   4.00   3rd Qu.:  251.5   3rd Qu.:  0.000   3rd Qu.: 22.00
##  Max.   :3477.00   Max.   :10980.0   Max.   :215.000   Max.   :224.00
##      Copper       Cryptoxanthin, beta     Energy
##  Min.   :0.0000   Min.   :   0.00     Min.   : 11.0
##  1st Qu.:0.0420   1st Qu.:   0.00     1st Qu.: 34.0
##  Median :0.0730   Median :   0.00     Median : 68.0
##  Mean   :0.1313   Mean   :  27.87     Mean   :152.8
##  3rd Qu.:0.1500   3rd Qu.:   1.50     3rd Qu.:238.0
##  Max.   :2.2200   Max.   :1447.00     Max.   :886.0
##  Fatty acids, total monounsaturated Fatty acids, total polyunsaturated
##  Min.   : 0.0000                    Min.   : 0.0000
##  1st Qu.: 0.0185                    1st Qu.: 0.0555
##  Median : 0.0860                    Median : 0.1440
##  Mean   : 2.4454                    Mean   : 1.4888
##  3rd Qu.: 1.0440                    3rd Qu.: 0.5410
##  Max.   :40.4390                    Max.   :43.2770
##  Fatty acids, total saturated Fiber, total dietary  Folate, DFE
##  Min.   : 0.000               Min.   : 0.000       Min.   :   0.00
##  1st Qu.: 0.026               1st Qu.: 0.600       1st Qu.:   6.00
##  Median : 0.071               Median : 1.700       Median :  18.00
##  Mean   : 1.676               Mean   : 2.221       Mean   :  81.44
##  3rd Qu.: 1.123               3rd Qu.: 2.950       3rd Qu.:  57.00
##  Max.   :51.368               Max.   :11.800       Max.   :1256.00
##   Folate, food    Folate, total      Folic acid          Iron
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 0.000
##  1st Qu.:  5.00   1st Qu.:  6.00   1st Qu.:  0.00   1st Qu.: 0.250
##  Median : 14.00   Median : 18.00   Median :  0.00   Median : 0.580
##  Mean   : 23.77   Mean   : 57.69   Mean   : 33.93   Mean   : 2.367
##  3rd Qu.: 28.50   3rd Qu.: 50.50   3rd Qu.:  0.00   3rd Qu.: 1.765
##  Max.   :194.00   Max.   :741.00   Max.   :737.00   Max.   :33.300
##  Lutein + zeaxanthin    Lycopene        Magnesium          Niacin
##  Min.   :    0.0     Min.   :   0.0   Min.   :  0.00   Min.   : 0.000
##  1st Qu.:    0.0     1st Qu.:   0.0   1st Qu.: 10.00   1st Qu.: 0.252
##  Median :   19.0     Median :   0.0   Median : 15.00   Median : 0.640
##  Mean   :  550.4     Mean   : 227.4   Mean   : 28.42   Mean   : 2.297
##  3rd Qu.:  129.5     3rd Qu.:   0.0   3rd Qu.: 27.00   3rd Qu.: 1.629
##  Max.   :12500.0     Max.   :6312.0   Max.   :279.00   Max.   :28.967
##    Phosphorus        Potassium        Protein          Retinol
##  Min.   :   0.00   Min.   :  0.0   Min.   : 0.000   Min.   :   0.00
##  1st Qu.:  18.50   1st Qu.:115.5   1st Qu.: 0.815   1st Qu.:   0.00
##  Median :  41.00   Median :191.0   Median : 1.800   Median :   0.00
##  Mean   :  88.28   Mean   :231.9   Mean   : 4.750   Mean   :  60.64
##  3rd Qu.:  89.00   3rd Qu.:314.0   3rd Qu.: 4.920   3rd Qu.:   0.00
##  Max.   :1321.00   Max.   :762.0   Max.   :78.130   Max.   :1250.00
##    Riboflavin       Selenium          Sodium       Sugars, total
##  Min.   :0.000   Min.   :  0.000   Min.   :   0.0   Min.   : 0.000
##  1st Qu.:0.027   1st Qu.:  0.400   1st Qu.:   4.0   1st Qu.: 0.890
##  Median :0.055   Median :  0.900   Median :  38.0   Median : 3.940
##  Mean   :0.198   Mean   :  5.558   Mean   : 204.0   Mean   : 8.040
```

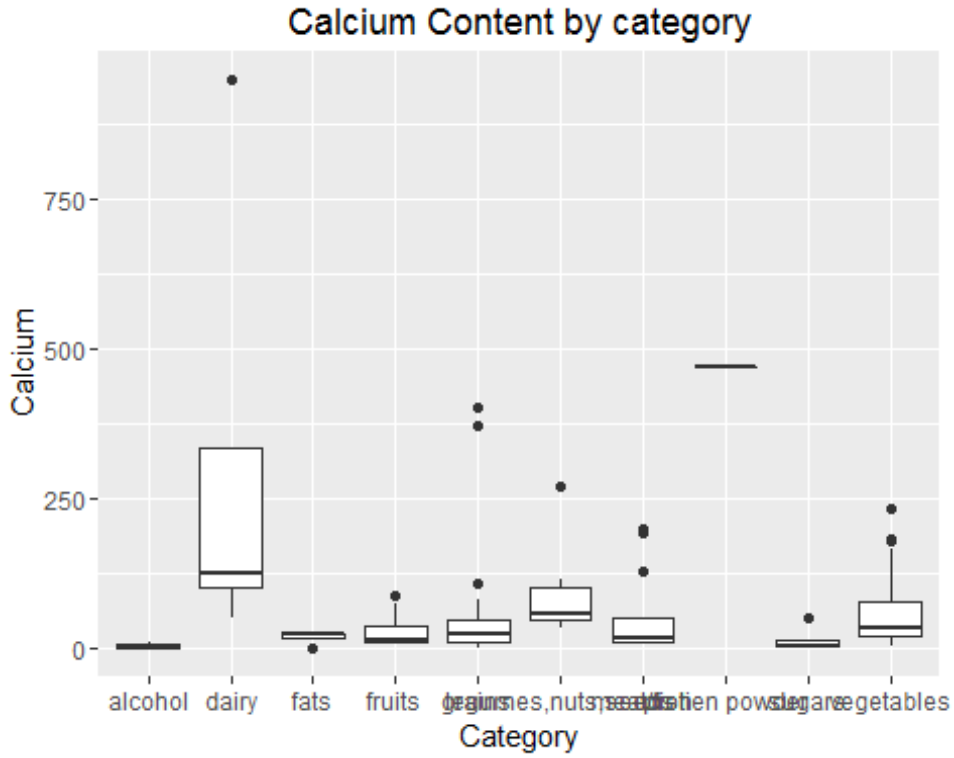
```
##  3rd Qu.:0.151   3rd Qu.:  5.600   3rd Qu.: 331.5   3rd Qu.: 9.205
##  Max.   :2.827   Max.   :111.400   Max.   :1737.0   Max.   :99.800
##   Theobromine         Thiamin          Total Fat       Vitamin A, RAE
##  Min.   : 0.0000   Min.   :0.0000   Min.   :  0.000   Min.   :   0.0
##  1st Qu.: 0.0000   1st Qu.:0.0280   1st Qu.:  0.175   1st Qu.:   0.0
##  Median : 0.0000   Median :0.0520   Median :  0.420   Median :  10.0
##  Mean   : 0.9386   Mean   :0.1622   Mean   :  6.022   Mean   : 107.7
##  3rd Qu.: 0.0000   3rd Qu.:0.1175   3rd Qu.:  3.485   3rd Qu.:  55.0
##  Max.   :83.0000   Max.   :2.0500   Max.   :100.000   Max.   :1250.0
##   Vitamin B-12    Vitamin B-12, added  Vitamin B-6       Vitamin C
##  Min.   :0.0000   Min.   :0.0000      Min.   :0.0000   Min.   :  0.00
##  1st Qu.:0.0000   1st Qu.:0.0000      1st Qu.:0.0435   1st Qu.:  0.10
##  Median :0.0000   Median :0.0000      Median :0.0900   Median :  5.90
##  Mean   :0.5129   Mean   :0.3361      Mean   :0.2546   Mean   : 18.35
##  3rd Qu.:0.0650   3rd Qu.:0.0000      3rd Qu.:0.1945   3rd Qu.: 25.95
##  Max.   :7.1000   Max.   :7.1000      Max.   :2.4910   Max.   :228.30
##  Vitamin D (D2 + D3) Vitamin E (alpha-tocopherol) Vitamin E, added
##  Min.   :0.0000      Min.   : 0.000               Min.   : 0.0000
##  1st Qu.:0.0000      1st Qu.: 0.100               1st Qu.: 0.0000
##  Median :0.0000      Median : 0.300               Median : 0.0000
##  Mean   :0.2804      Mean   : 1.064               Mean   : 0.1154
##  3rd Qu.:0.0000      3rd Qu.: 0.785               3rd Qu.: 0.0000
##  Max.   :5.0000      Max.   :23.900               Max.   :11.1800
##  Vitamin K (phylloquinone)     Water            Zinc
##  Min.   :   0.0            Min.   : 0.00   Min.   : 0.000
##  1st Qu.:   0.3            1st Qu.:58.71   1st Qu.: 0.105
##  Median :   2.6            Median :83.07   Median : 0.250
##  Mean   :  46.9            Mean   :67.25   Mean   : 1.116
##  3rd Qu.:  13.2            3rd Qu.:89.70   3rd Qu.: 0.795
##  Max.   :1640.0            Max.   :96.73   Max.   :16.730
##    Category
##  Length:163
##  Class :character
##  Mode  :character
##
##
##
```

```
Data<-food_nutrient_int

#omit first 20 nnumeric columns
Data<-Data[,c(20:66)]

Names_Data<-colnames(Data)

#replace all special characters with underscore
Names_Data<-gsub('([[:punct:]])|\\s+','_',Names_Data)

colnames(Data)<-Names_Data
```

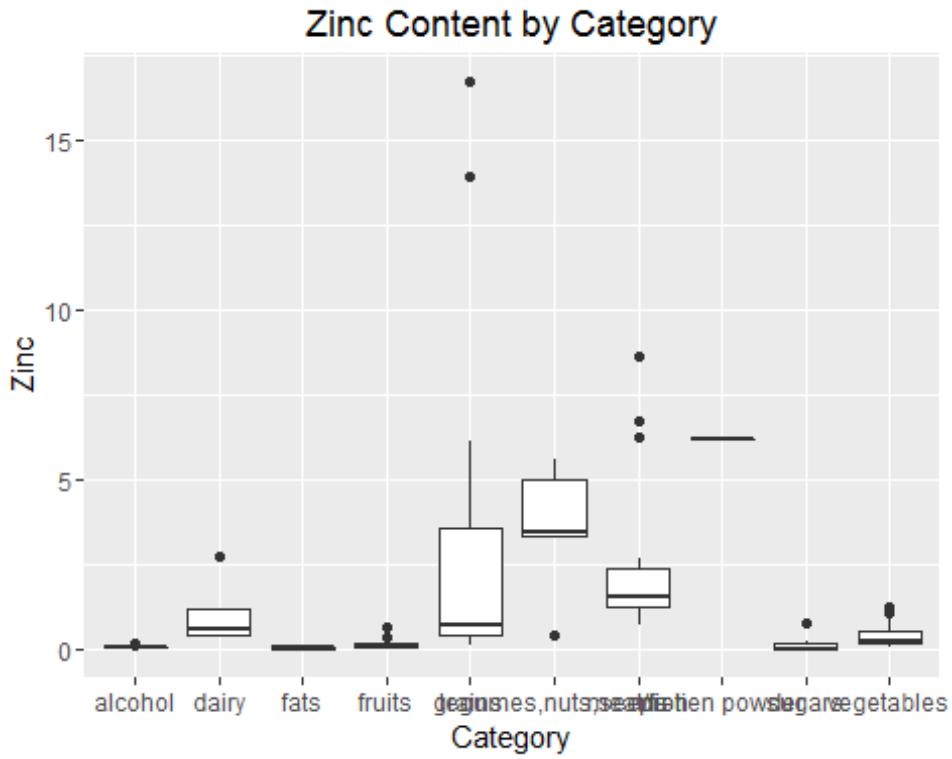
```
#Add a column food_code to data
Data$food_code<-rownames(food_nutrient_mat)
```

Visualizations

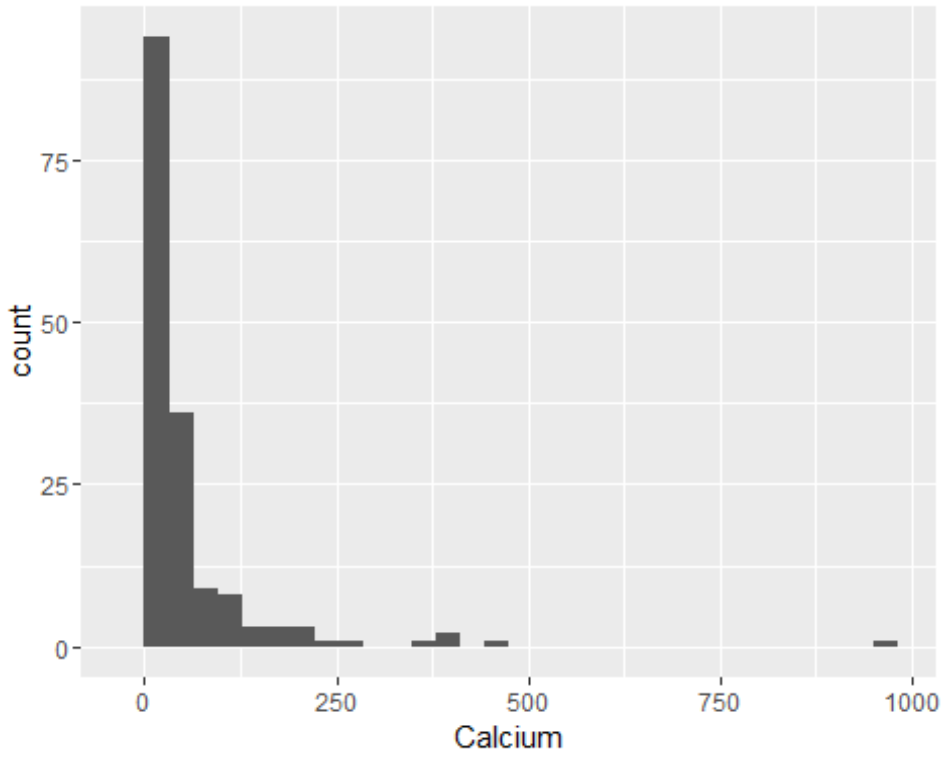
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.1.3

g=ggplot(Data)
#1
g+geom_boxplot(aes(x=Category,y=Calcium))+ggtitle("Calcium Content by
category")
```



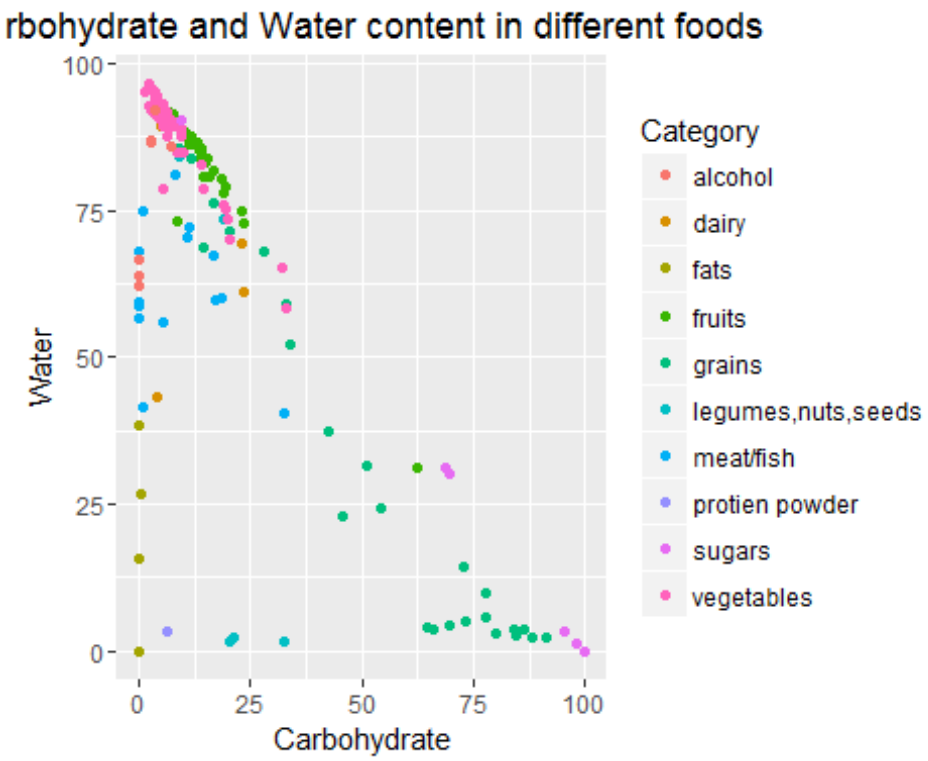

```
#Dairy products are high and aclohols and sugars least in calcium content.

#2
g+geom_boxplot(aes(x=Category,y=Zinc))+ggtitle("Zinc Content by Category")
```



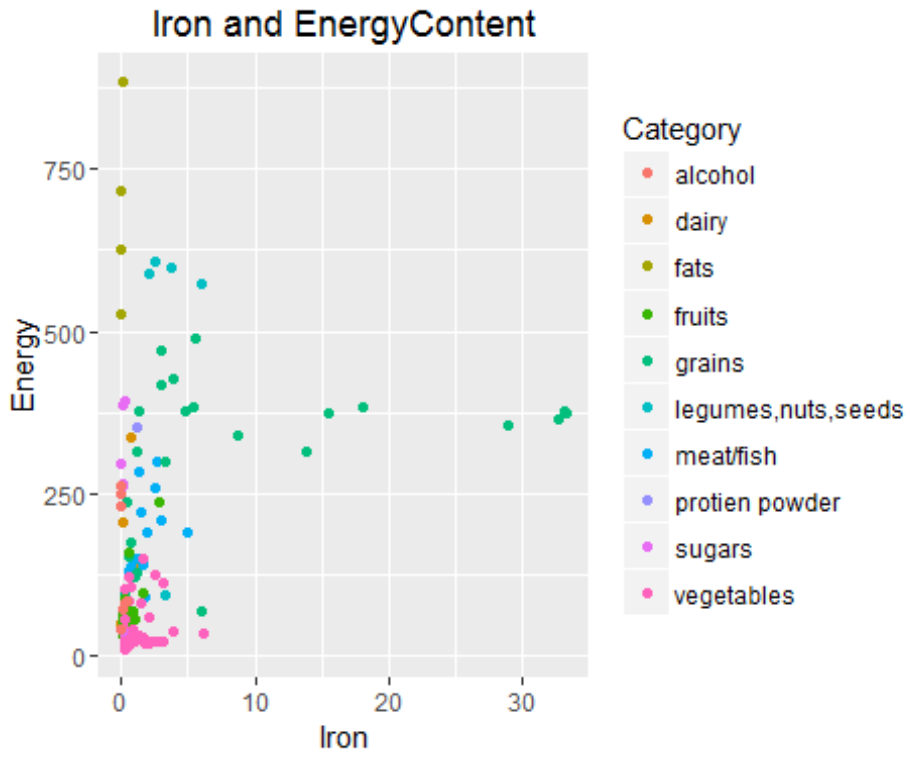

```
#Grains high in zinc content followed by meat/fish. Least in alcohol and
fats.
#3
g+geom_histogram(aes(x=Calcium))

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



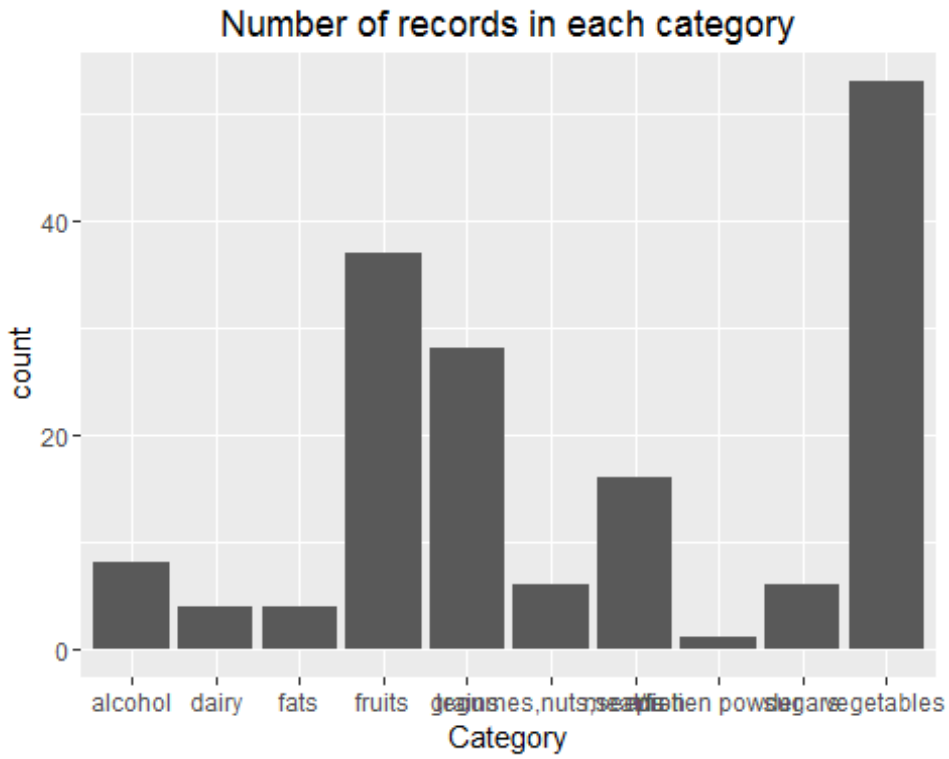

```
#Most of the foods range in Calcium content of 0 to 100. Very few beyond
that.
#4
g+geom_point(aes(x=Carbohydrate,y=Water,color= Category,Shape=Category))
+ggtitle("Carbohydrate and Water content in different foods")
```



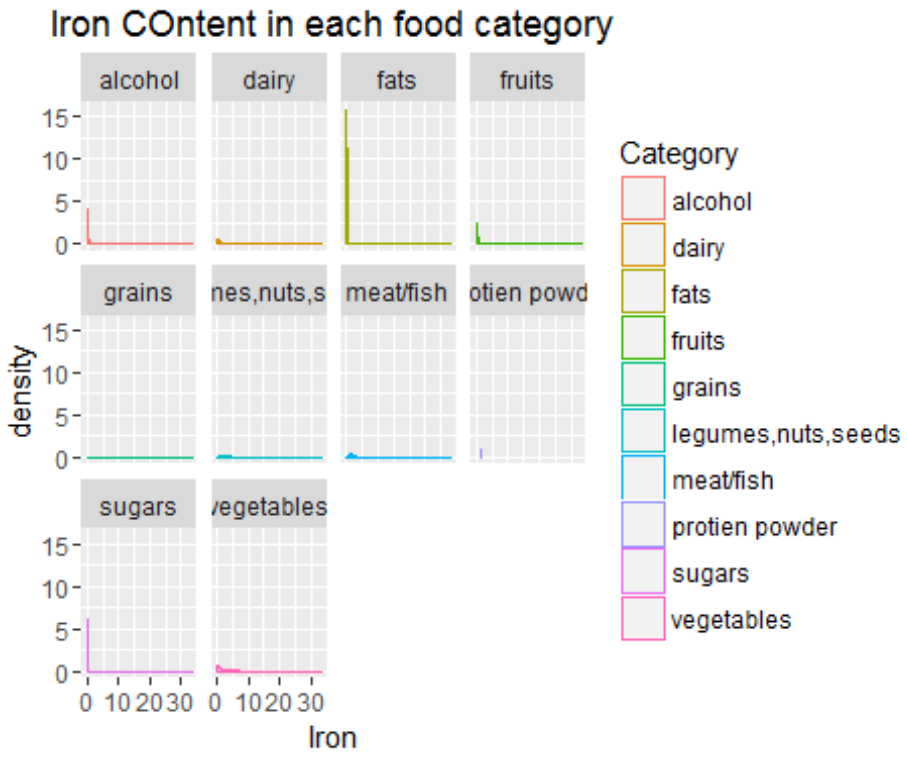

```
#Veggies have high water and low carb content and alcohol vice-versa.

#5

g+geom_point((aes(x=Iron ,y= Energy, color=Category)))+ggtitle("Iron and
EnergyContent")+
  xlab("Iron")+ylab("Energy")
```



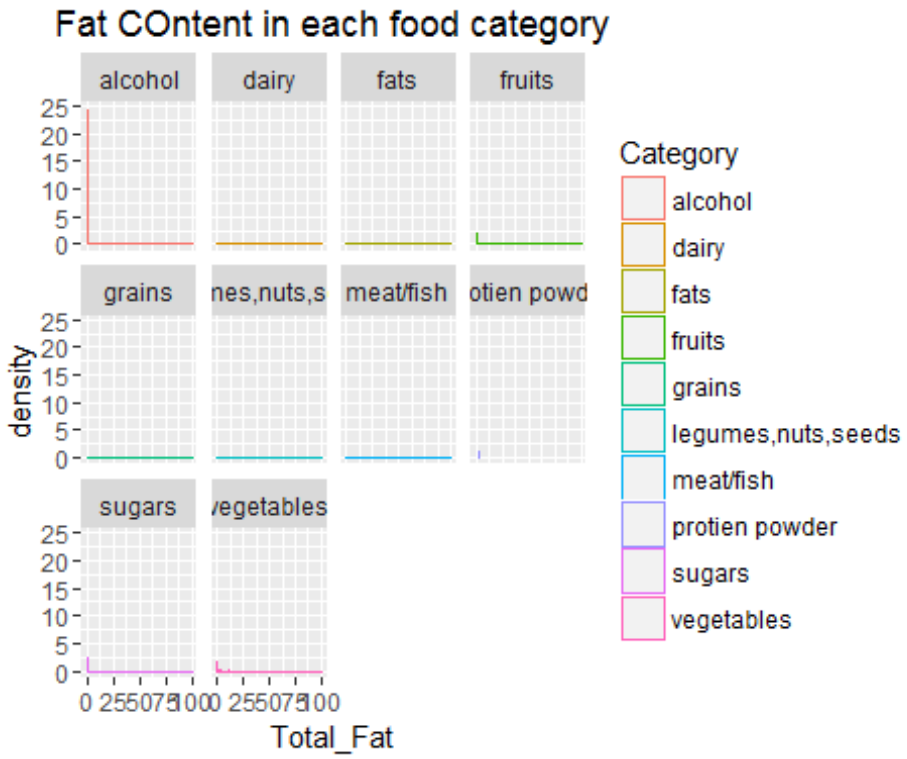

```
#Alcohols and Vegetables low on Iron content, but grains very high, with mid
range energy levels.

#6
g+geom_bar(aes(x=Category,Color="Blue"))+ggtitle("Number of records in each
category")
```



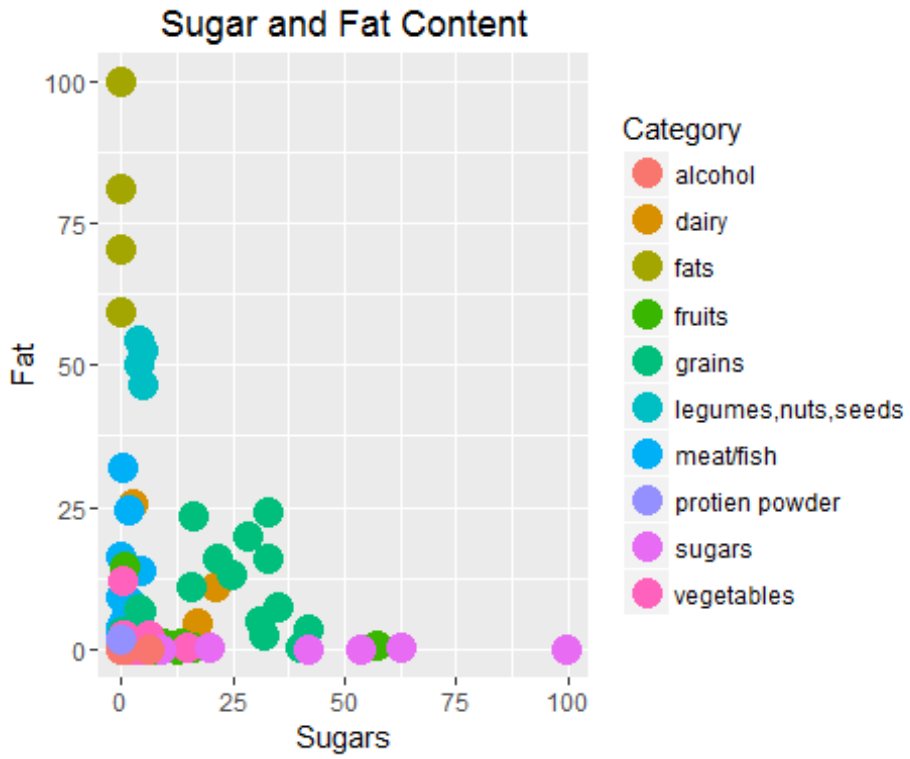

```
#Already accomplished using table, however now representing it visually,

#7

ggplot(Data,aes(x=Iron))+geom_density(aes(color=Category))+facet_wrap(~Catego
ry)+ggtitle("Iron COntent in each food category")
```



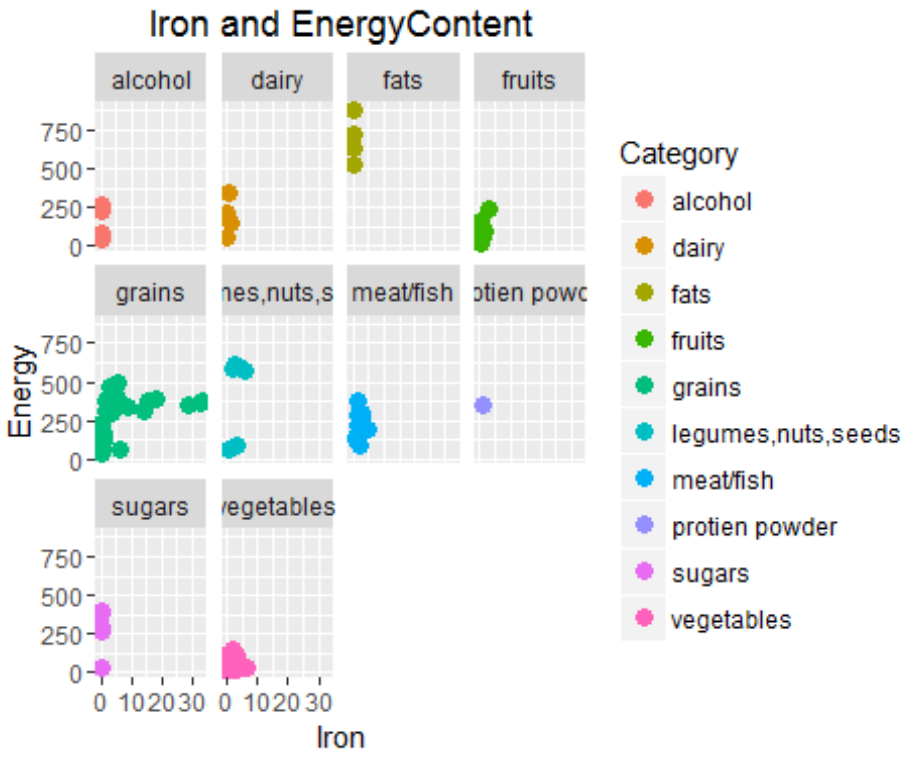

```
#nuts,meat/fish, vegetables have decent Iron content. Almost none for
Alcohol, fats.

#8
ggplot(Data,aes(x=Total_Fat))+geom_density(aes(color=Category))+facet_wrap(~C
ategory)+ggtitle("Fat COntent in each food category")
```



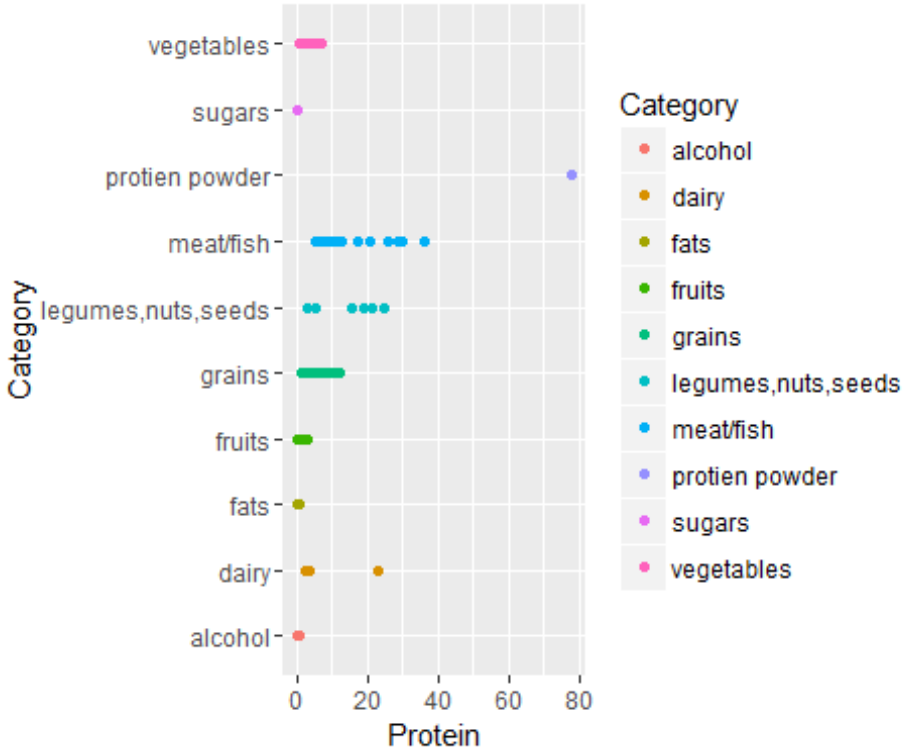

```
#Alcohol seems to contain a lot of fats.

#9
g+geom_point((aes(x=Sugars__total ,y= Total_Fat,
color=Category)),size=5)+ggtitle("Sugar and Fat Content")+
  xlab("Sugars")+ylab("Fat")
```



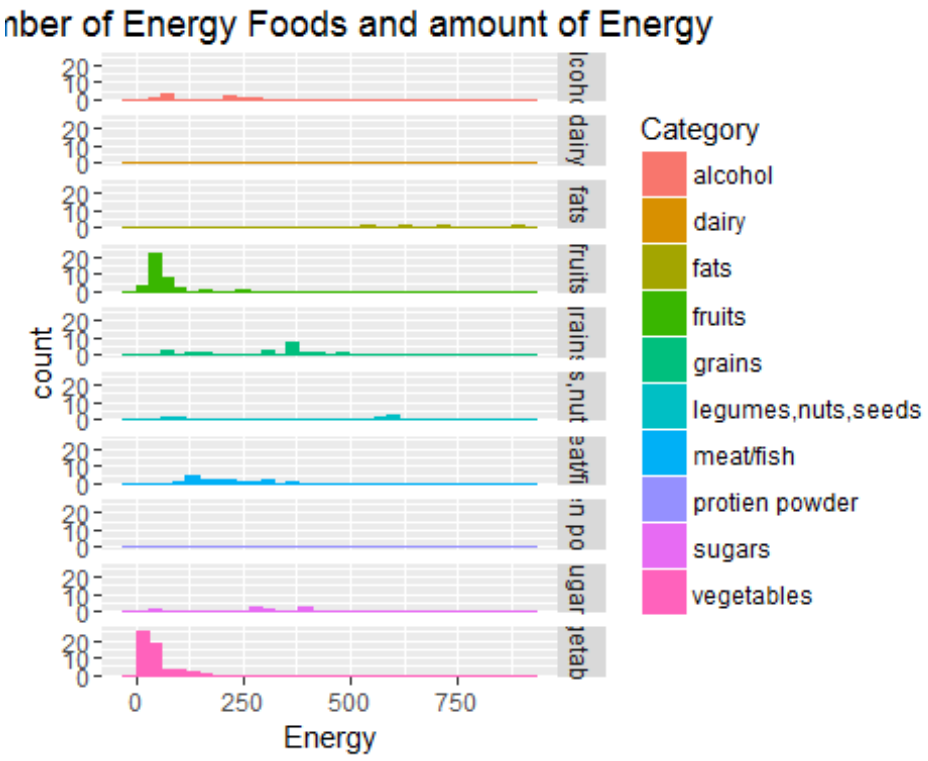

```
#Vegetables least sugar and Fat content.


#10
g+geom_point((aes(x=Iron ,y= Energy, color=Category)),size=3)+ggtitle("Iron
and EnergyContent")+
  xlab("Iron")+ylab("Energy")+facet_wrap(~Category)
```



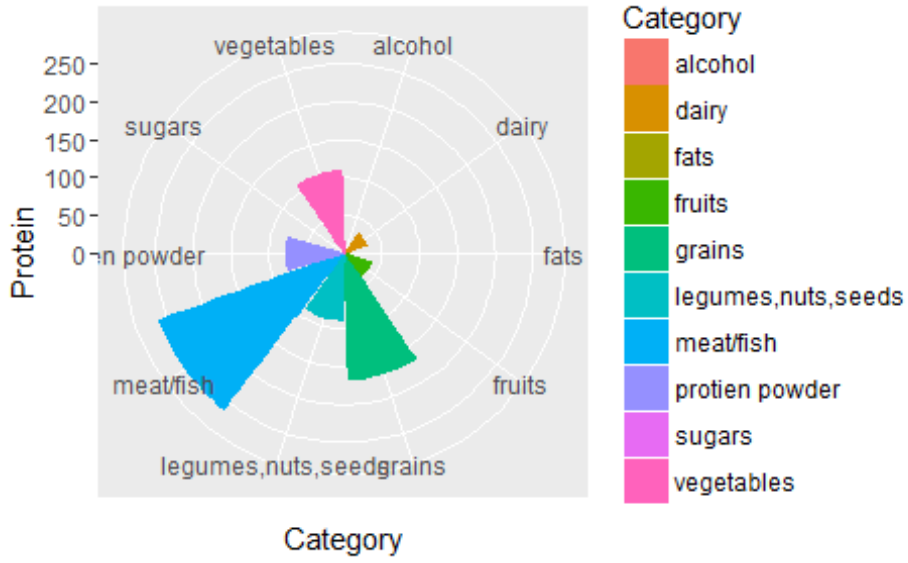

```
#Another way of visualising.

#11
g+geom_point(aes(x=Protein,y=Category,color=Category))
```



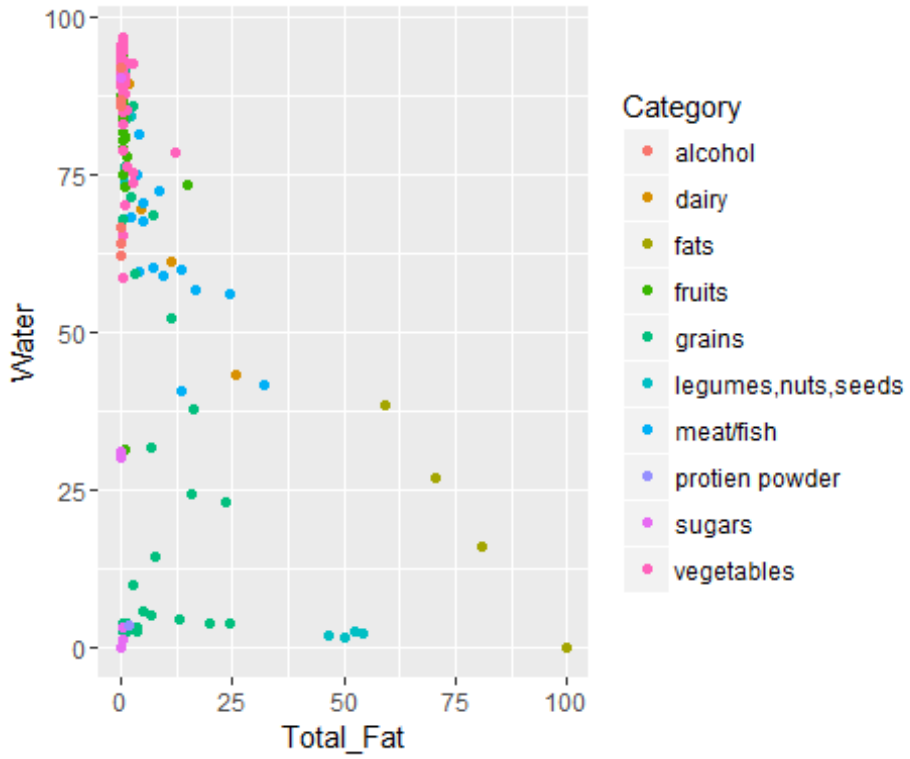

```
#Protein powder has highest protein, and alocohols the least.
#12
ggplot(Data,aes(Energy))+

geom_histogram(aes(color=Category,fill=Category))+facet_grid(Category~.)+ggti
tle("Number of Energy Foods and amount of Energy")

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




```
#Energy values for each category, and number of such foods

#13
ggplot(Data,aes(x=Category,y=Protein))+geom_bar(stat="identity",aes(fill=Cate
gory,color=Category))+coord_polar(theta = "x", direction=1 )
```




```
#Protein content in each category
#No Protein in alcohol, fats, high protein in grains, meat/fish.

#14
ggplot(Data,aes(x=Total_Fat,y=Water))+geom_point(aes(fill=Category,color=Cate
gory))
```




```
#Grains are mid range fat and low on water. (Of course!)
```

Visualisations help us understand the data, in terms of how each categories differ, the ranges of nutrients etc. Based on this primitive understanding, we can gain deeper sense and inisghtful understanding using the SQL queries.

SQL QUERIES

SQL Queries to subset Data, used for analysis later

```
Fruits_df<-sqldf('select * from Data WHERE Category ="fruits"')
Grains_df<-sqldf('select * from Data WHERE Category = "grains"')
Meat_df<-sqldf('select * from Data WHERE Category="meat/fish"')
Veggie_df<-sqldf('select * from Data WHERE Category="vegetables"')
Dairy_df<-sqldf('select * from Data WHERE Category="dairy"')

Data_Nut<-sqldf('select
Carbohydrate,Calcium,Cholesterol,Protein,Sugars__total as
Sugars,Total_Fat,Category from Data')
```

** Following SQL Queries were written to understand and explore the data set.**

```
#1

sqldf('select Category, count(*) as count from Data GROUP BY Category')

##              Category count
## 1             alcohol     8
```

```
## 2               dairy     4
## 3                fats     4
## 4              fruits    37
## 5              grains    28
## 6  legumes,nuts,seeds     6
## 7           meat/fish    16
## 8      protien powder     1
## 9              sugars     6
## 10         vegetables    53
```

```
#Gives the number of records in each category
#2
head(sqldf("SELECT f.food_code,
f.main_food_description,nd.nutrient_description, nv.nutrient_value
  FROM foods f
  INNER JOIN FNDDSNutVal nv ON f.food_code = nv.food_code
      INNER JOIN NutDesc nd ON nv.nutrient_code = nd.nutrient_code
      ORDER BY f.main_food_description"))
```

```
##   food_code main_food_description nutrient_description nutrient_value
## 1  42100100               Almonds              Protein          20.96
## 2  42100100               Almonds            Total Fat          52.54
## 3  42100100               Almonds          Carbohydrate          21.01
## 4  42100100               Almonds               Energy         598.00
## 5  42100100               Almonds              Alcohol           0.00
## 6  42100100               Almonds                Water           2.41
```

```
#returns first six rows of food code, food description, nutrient description
and value for each food.

#3
head(sqldf("SELECT AVG(nv.end_date-nv.start_date) as
Avg_duration,f.main_food_description
      FROM foods f
      INNER JOIN FNDDSNutVal nv on f.food_code=nv.food_code
      GROUP BY f.main_food_description"))
```

```
##   Avg_duration main_food_description
## 1           11               Almonds
## 2           11                 Apple
## 3           11               Apricot
## 4           11             Asparagus
## 5           11               Avocado
## 6           11           Baked beans
```

```
#returns average life of food. First 6 rows

#4
sqldf("SELECT AVG(Water),AVG(Total_Fat),Category from Data
      GROUP BY Category
      ORDER BY Category")
```

```
##    AVG(Water) AVG(Total_Fat)           Category
## 1    76.30750      0.0062500            alcohol
## 2    65.72000     10.7975000              dairy
## 3    20.25250     77.7300000               fats
## 4    83.58297      0.7083784             fruits
## 5    33.18821      6.7828571             grains
## 6    27.62500     34.4600000 legumes,nuts,seeds
## 7    61.44375     10.3487500          meat/fish
## 8     3.44000      1.5600000     protien powder
## 9    26.01500      0.1283333              sugars
## 10   88.23679      0.6924528         vegetables
```

```
#Average Water and Fat in each category

#5
sqldf("SELECT
AVG(Calcium),AVG(Copper),AVG(Magnesium),AVG(Iron),AVG(Phosphorus),AVG(Potassi
um),AVG(Sodium),AVG(Zinc)
      FROM Data
      GROUP BY Category
      ORDER BY Category")
```

```
##    AVG(Calcium) AVG(Copper) AVG(Magnesium) AVG(Iron) AVG(Phosphorus)
## 1       3.00000  0.01412500       3.625000 0.1275000       10.000000
## 2     312.00000  0.04025000      18.000000 0.5100000      213.750000
## 3      16.75000  0.00025000       1.500000 0.0300000       15.000000
## 4      23.40541  0.07605405      14.513514 0.4032432       23.405405
## 5      65.64286  0.17692857      39.000000 9.2978571      145.821429
## 6      95.33333  0.90366667     166.666667 3.0800000      324.833333
## 7      53.62500  0.10862500      19.687500 1.8225000      165.562500
## 8     469.00000  0.04900000     195.000000 1.1300000     1321.000000
## 9      12.33333  0.01916667       2.666667 0.1033333        4.833333
## 10     53.88679  0.11388679      25.849057 1.0935849       47.132075
##    AVG(Potassium) AVG(Sodium) AVG(Zinc)
## 1        30.75000    2.750000 0.0575000
## 2       168.00000  273.000000 1.0500000
## 3        20.25000  499.250000 0.0325000
## 4       209.51351    4.081081 0.1305405
## 5       193.32143  420.035714 2.9385714
## 6       509.33333  439.833333 3.6200000
## 7       262.18750  662.625000 2.5537500
## 8       500.00000  156.000000 6.1800000
## 9        19.16667   24.666667 0.1766667
## 10      297.49057   88.377358 0.3813208
```

```
#Average Minerals in each category
#6
sqldf("SELECT MIN(Energy),MAX(Energy),AVG(Energy), Category from Data
      GROUP BY Category
      ORDER BY Category")
```

```
##    MIN(Energy) MAX(Energy) AVG(Energy)           Category
## 1           43         263   157.25000            alcohol
## 2           50         338   184.25000              dairy
## 3          528         886   689.00000               fats
## 4           21         239    59.86486             fruits
## 5           34         490   284.78571             grains
## 6           68         609   421.83333 legumes,nuts,seeds
## 7           90         378   203.87500          meat/fish
## 8          352         352   352.00000     protien powder
## 9           37         394   273.66667              sugars
## 10          11         149    40.98113         vegetables
```

```
#Energy metrics of foods in each category

#7
head(sqldf("SELECT  nd.nutrient_description nutrient,
fnv.nutrient_value nutrient_value
FROM Data fng
INNER JOIN FNDDSNutVal fnv ON fnv.food_code = fng.food_code
INNER JOIN NutDesc nd ON fnv.nutrient_code = nd.nutrient_code
WHERE category IN ('fruits','grains', 'vegetables' )"))
```

```
##       nutrient nutrient_value
## 1      Protein           8.00
## 2    Total Fat           6.70
## 3 Carbohydrate          51.21
## 4       Energy         301.00
## 5      Alcohol           0.00
## 6        Water          31.58
```

```
#Description of every nutrient and the value in categories fruits, grains and
vegetables.

#8
head(sqldf("SELECT DISTINCT(nutrient_description)
FROM NutDesc"));
```

```
##   nutrient_description
## 1              Protein
## 2            Total Fat
## 3          Carbohydrate
## 4               Energy
## 5              Alcohol
## 6                Water
```

```
#returns the different nutrients

#9
sqldf("SELECT MIN(Protein),MAX(Protein),AVG(Protein), Category from Data
      GROUP BY Category
      ORDER BY Category")
```

```
##    MIN(Protein) MAX(Protein) AVG(Protein)           Category
## 1          0.00         0.46     0.085000            alcohol
## 2          2.09        22.72     7.897500              dairy
## 3          0.00         0.85     0.472500               fats
## 4          0.11         2.80     1.002973             fruits
## 5          0.92        12.09     5.982500             grains
## 6          3.08        24.55    14.698333 legumes,nuts,seeds
## 7          5.23        36.08    16.244375          meat/fish
## 8         78.13        78.13    78.130000     protien powder
## 9          0.00         0.07     0.015000             sugars
## 10         0.59         6.84     2.058868         vegetables
```

```
#Protien levels by category

#10
sqldf("SELECT MIN(Carbohydrate),MAX(Carbohydrate),AVG(Carbohydrate), Category
from Data
      GROUP BY Category
      ORDER BY Category")
```

```
##    MIN(Carbohydrate) MAX(Carbohydrate) AVG(Carbohydrate)
## 1               0.00              7.25          2.013750
## 2               3.82             23.60         13.820000
## 3               0.00              0.46          0.155000
## 4               4.54             62.50         14.153784
## 5               5.80             91.30         52.279286
## 6               8.89             32.69         20.471667
## 7               0.00             32.67          9.768750
## 8               6.25              6.25          6.250000
## 9               9.56             99.98         73.493333
## 10              1.29             33.06          7.864151
##              Category
## 1             alcohol
## 2               dairy
## 3                fats
## 4              fruits
## 5              grains
## 6  legumes,nuts,seeds
## 7           meat/fish
## 8      protien powder
## 9               sugars
## 10         vegetables
```

```
#Carbs by category

#11
sqldf("SELECT
AVG(Carbohydrate),AVG(Total_Fat),AVG(Protein),AVG(Energy),Category
  from Data
    GROUP BY Category
      ORDER BY Category")
```

```
##    AVG(Carbohydrate) AVG(Total_Fat) AVG(Protein) AVG(Energy)
## 1           2.013750      0.0062500     0.085000   157.25000
## 2          13.820000     10.7975000     7.897500   184.25000
## 3           0.155000     77.7300000     0.472500   689.00000
## 4          14.153784      0.7083784     1.002973    59.86486
## 5          52.279286      6.7828571     5.982500   284.78571
## 6          20.471667     34.4600000    14.698333   421.83333
## 7           9.768750     10.3487500    16.244375   203.87500
## 8           6.250000      1.5600000    78.130000   352.00000
## 9          73.493333      0.1283333     0.015000   273.66667
## 10          7.864151      0.6924528     2.058868    40.98113
##              Category
## 1             alcohol
## 2               dairy
## 3                fats
## 4              fruits
## 5              grains
## 6  legumes,nuts,seeds
## 7           meat/fish
## 8      protien powder
## 9              sugars
## 10         vegetables
```

```
#average nutritonal levels of each category

#12
Fatty_Data<-cbind(Data$Category,Data[,c(grep("Fatty",colnames(Data)))])
colnames(Fatty_Data)=c("Category","Mono","Poly","Saturated")
sqldf("select AVG(Mono),AVG(Poly),AVG(Saturated),Category
      from Fatty_Data
      GROUP BY Category
      ORDER BY Category")
```

```
##     AVG(Mono)   AVG(Poly) AVG(Saturated)           Category
## 1   0.0003750  0.00112500      0.0005000            alcohol
## 2   3.3515000  0.36850000      6.3035000              dairy
## 3  25.8405000 21.14275000     26.5865000               fats
## 4   0.3227838  0.14345946      0.1084595             fruits
## 5   2.5535000  2.20717857      1.5891429             grains
## 6  19.3080000  8.54266667      5.0283333 legumes,nuts,seeds
## 7   4.2552500  1.70062500      3.4866250          meat/fish
## 8   0.1580000  0.29900000      0.7810000     protien powder
## 9   0.0240000  0.04816667      0.0140000             sugars
## 10  0.2671132  0.19726415      0.1198113         vegetables
```

```
#Fatty acid metrics of each category

#13

head(sqldf("select d.Category,nv.nutrient_value,nd.nutrient_description
```

```
      from Data d
      INNER JOIN FNDDSNutVal nv ON d.food_code=nv.food_code
      INNER JOIN NutDesc nd ON nv.nutrient_code=nd.nutrient_code"))
```

```
##   Category nutrient_value nutrient_description
## 1    dairy           3.28              Protein
## 2    dairy           1.91            Total Fat
## 3    dairy           4.85        Carbohydrate
## 4    dairy          50.00               Energy
## 5    dairy           0.00              Alcohol
## 6    dairy          89.25                Water
```

```
#Nutrient values for each category

#14

sqldf('SELECT MIN(Total_Fat),MAX(Total_Fat),AVG(Total_Fat),Category from Data
      GROUp BY Category')
```

```
##    MIN(Total_Fat) MAX(Total_Fat) AVG(Total_Fat)           Category
## 1            0.00           0.05      0.0062500            alcohol
## 2            1.91          25.68     10.7975000              dairy
## 3           59.35         100.00     77.7300000               fats
## 4            0.06          14.66      0.7083784             fruits
## 5            0.28          24.28      6.7828571             grains
## 6            0.93          54.39     34.4600000 legumes,nuts,seeds
## 7            2.06          31.81     10.3487500          meat/fish
## 8            1.56           1.56      1.5600000     protien powder
## 9            0.00           0.37      0.1283333             sugars
## 10           0.09          12.07      0.6924528         vegetables
```

```
#fat metrics for each category

#15

sqldf('SELECT SUM(Total_Fat),Category from Data
      GROUP BY Category')
```

```
##    SUM(Total_Fat)           Category
## 1            0.05            alcohol
## 2           43.19              dairy
## 3          310.92               fats
## 4           26.21             fruits
## 5          189.92             grains
## 6          206.76 legumes,nuts,seeds
## 7          165.58          meat/fish
## 8            1.56     protien powder
## 9            0.77             sugars
## 10          36.70         vegetables
```

```
#total fat in each category

#16
sqldf('SELECT DISTINCT Category from Data
       WHERE Energy>500')
```

```
##             Category
## 1 legumes,nuts,seeds
## 2               fats
```

```
#high energy foods in which category

#17
sqldf('SELECT count(food_code) as no_of_foods,category from Data
      WHERE Energy>500
      GROUP BY Category')
```

```
##   no_of_foods           Category
## 1           4               fats
## 2           4 legumes,nuts,seeds
```

```
#number of high energy foods

#18
sqldf('SELECT DISTINCT Category from Data
      WHERE Carbohydrate>50 AND Total_Fat<50')
```

```
##   Category
## 1   grains
## 2   fruits
## 3   sugars
```

```
#foods with more carbs and less fat

#19
sqldf('SELECT count(food_code) as no_of_foods,category from Data
      WHERE Calcium>100
      GROUP BY Category')
```

```
##   no_of_foods           Category
## 1           3              dairy
## 2           4             grains
## 3           2 legumes,nuts,seeds
## 4           4          meat/fish
## 5           1     protien powder
## 6           9         vegetables
```

```
#number of foods high in calcium

#20
sqldf('SELECT count(food_code), Category from Data
```

```
      WHERE Carbohydrate>50 AND Total_Fat<50
      GROUP BY Category')
```

```
##   count(food_code) Category
## 1                1   fruits
## 2               16   grains
## 3                5   sugars
```

```
#number of high carb low fat foods.
#21
sqldf('SELECT f.main_food_description,D.Category
      FROM foods f
      INNER JOIN Data D ON D.food_code=f.food_code
      WHERE D.Energy>500')
```

```
##   main_food_description           Category
## 1               Almonds legumes,nuts,seeds
## 2           Cashew nuts legumes,nuts,seeds
## 3            Mixed nuts legumes,nuts,seeds
## 4               Peanuts legumes,nuts,seeds
## 5             Table fat               fats
## 6                Butter               fats
## 7             Margarine               fats
## 8         Vegetable oil               fats
```

```
#Gives the super energy foods.
```

The SQL Queries help us understand quite a bit about the data. Not only did we get to see the ranges of nutritional content among various categories, but we could also see which category/foods are high in nutritional values. With some insight into the nutrional values, we can now see how similar/different they are across categories using a t-test ### **T-TESTS**

```
#1
t.test(x = Fruits_df$Protein,y = Grains_df$Protein)
```

```
##
##  Welch Two Sample t-test
##
## data:  Fruits_df$Protein and Grains_df$Protein
## t = -7.8589, df = 28.416, p-value = 1.322e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.276577 -3.682477
## sample estimates:
## mean of x mean of y
##  1.002973  5.982500
```

```
#Reject Null Hypotheses, accept alternate hypotheses
#Protient content different in both categories


#2
```

```r
t.test(x= Fruits_df$Calcium,y= Grains_df$Calcium)
```

```
##
##  Welch Two Sample t-test
##
## data:  Fruits_df$Calcium and Grains_df$Calcium
## t = -1.8831, df = 28.233, p-value = 0.07003
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -88.166499   3.691595
## sample estimates:
## mean of x mean of y
##  23.40541  65.64286
```

```r
#Accept null hypotheses; similar calcium content

#3
t.test(x=Meat_df$Carbohydrate,y=Fruits_df$Carbohydrate,conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  Meat_df$Carbohydrate and Fruits_df$Carbohydrate
## t = -1.3803, df = 24.154, p-value = 0.1801
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -10.939676   2.169609
## sample estimates:
## mean of x mean of y
##   9.76875  14.15378
```

```r
#Accept null hypotheses, similar carbs.

#4
t.test(Veggie_df$Protein,Meat_df$Protein)
```

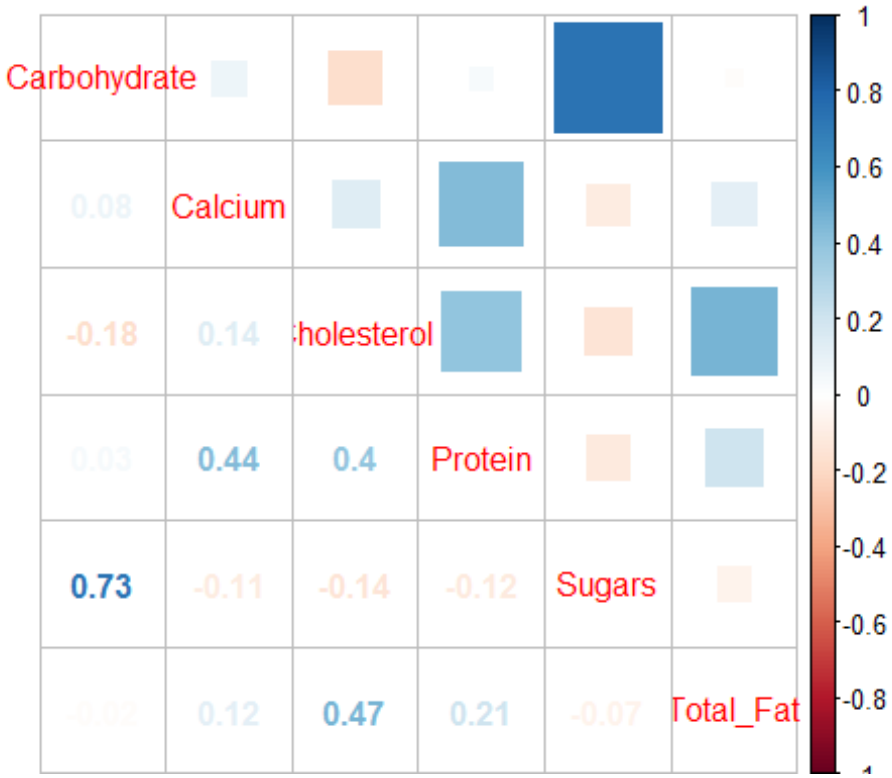
```
##
##  Welch Two Sample t-test
##
## data:  Veggie_df$Protein and Meat_df$Protein
## t = -6.073, df = 15.193, p-value = 2.018e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -19.158698  -9.212316
## sample estimates:
## mean of x mean of y
##  2.058868 16.244375
```

```r
#reject null hypotheses, dissimilar protien content
```

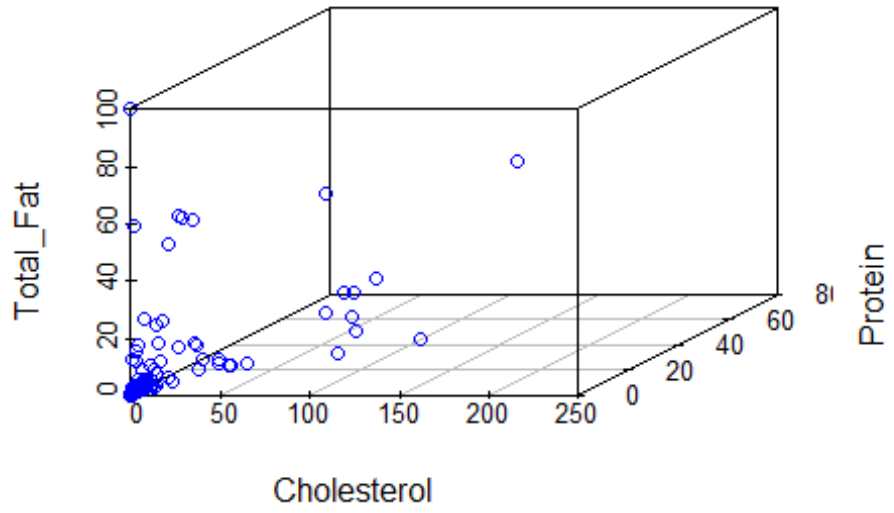
```
#5
t.test(Meat_df$Protein,Grains_df$Protein)

##
##  Welch Two Sample t-test
##
## data:  Meat_df$Protein and Grains_df$Protein
## t = 4.2564, df = 17.193, p-value = 0.0005206
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.179659 15.344091
## sample estimates:
## mean of x mean of y
##  16.24437   5.98250

#reject null hypotheses, dissimilar protein content.
```

The T-Tests on samples reveal that some categories of food have similar nutrient contents while others don't. This gives us an insight into which foods might be the "go-to" foods for specific nutrients. ### ** CORRELATIONS**

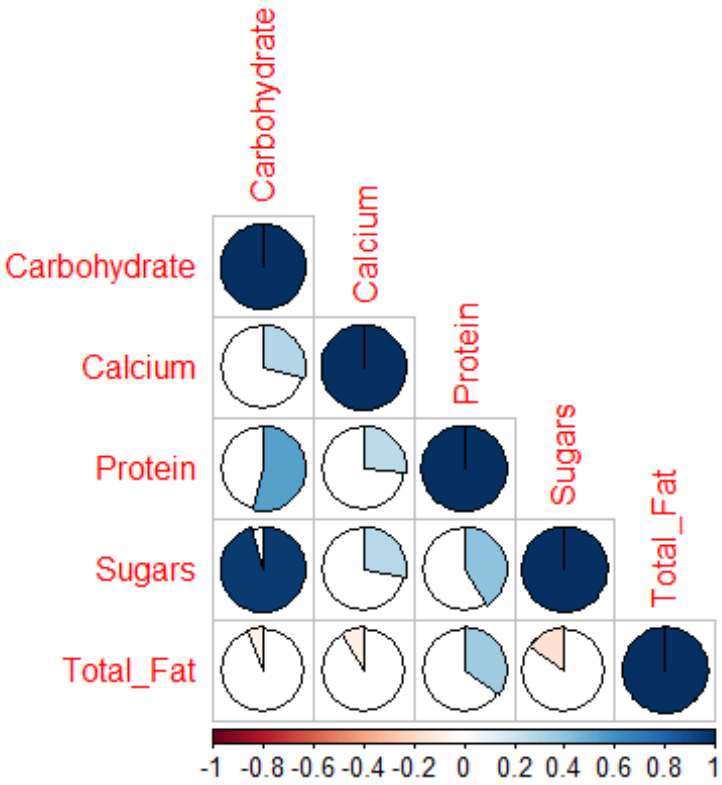
```
#install.packages("corrplot")
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.1.3

C<-as.matrix(cor(Data_Nut[-7]))
cor(Data_Nut[-7])

##              Carbohydrate     Calcium Cholesterol     Protein      Sugars
## Carbohydrate   1.00000000  0.07551887  -0.1751682  0.03093308  0.73150389
## Calcium        0.07551887  1.00000000   0.1363651  0.43590083 -0.10550005
## Cholesterol   -0.17516819  0.13636506   1.0000000  0.39697152 -0.14326592
## Protein        0.03093308  0.43590083   0.3969715  1.00000000 -0.11729602
## Sugars         0.73150389 -0.10550005  -0.1432659 -0.11729602  1.00000000
## Total_Fat     -0.01843762  0.11784949   0.4651718  0.20722571 -0.06717576
##                Total_Fat
## Carbohydrate -0.01843762
## Calcium       0.11784949
## Cholesterol   0.46517184
## Protein       0.20722571
## Sugars       -0.06717576
## Total_Fat     1.00000000

corrplot.mixed(C,upper = "square")
```



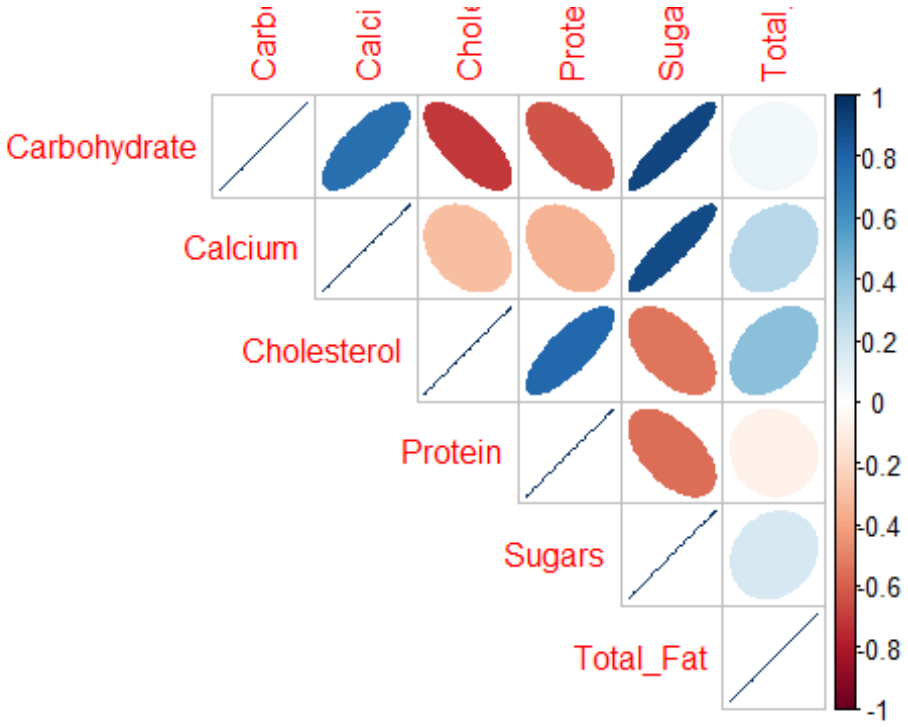

```
#Some correlation between Cholesterol, protein and fat

#install.packages("scatterplot3d")
library(scatterplot3d)

## Warning: package 'scatterplot3d' was built under R version 3.1.3

attach(Data_Nut)

scatterplot3d(x = Cholesterol,y = Protein,z = Total_Fat,color = "blue")
```



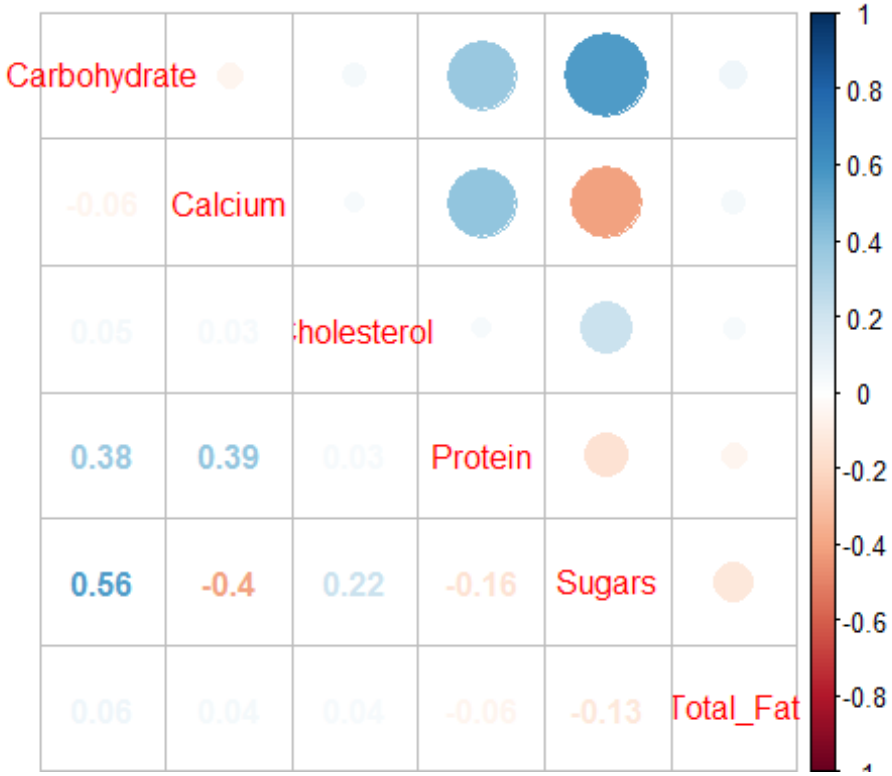

```r
#Examining Correlations between the different food categories

Fruit_Nut<-subset(Data_Nut,Category%in%c("fruits"))
Fruit_Nut<-Fruit_Nut[,!colnames(Fruit_Nut)%in%("Cholesterol")]

Veg_Nut<-subset(Data_Nut,Category%in%c("vegetables"))

Meat_Nut<-subset(Data_Nut,Category%in%c("meat/fish"))

F<-cor(Fruit_Nut[-6])
corrplot(as.matrix(F),method = "pie",type = "lower")
```



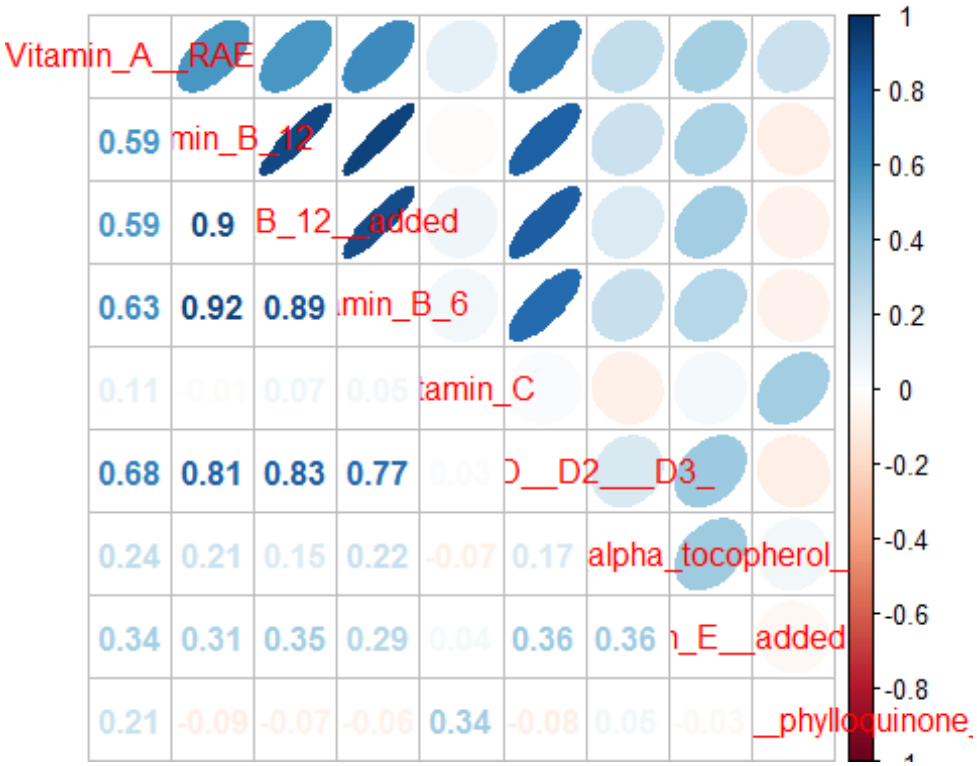

```
#Correlation between Protien and sugar, sugar and total fat.


M<-cor(Meat_Nut[-7])
corrplot(as.matrix(M),method = "ellipse",type="upper")
```




```
#strong correlation between calcium and sugar, carbs and sugar.
#negative correlations between cholesterol sugar, protein-sugar.

V<-cor(Veg_Nut[-7])
corrplot.mixed(as.matrix(V),lower = "number",upper = "circle")
```




```r
#positive correaltion carbs and sugars, negative correlation between Calcium and sugars.

Vit_Data<-Data[,c(grep("Vitamin",colnames(Data)))]

V<-as.matrix(cor(Vit_Data))

corrplot.mixed(V,upper="ellipse")
```




```
#Extremely high correlations between Vitamin B of all types and Between
Vitamin B of all types and Vitamin D.
```

RESULTS FROM MODELS

SVM

THE SVM MOdel has primarily been used for classification and prediction. Since the number of records are less, we have used the replace function to mimic a large dataset. The results of the Model are as below:

YTEST	**YPRED**	
	Fruits	**Vegetables**
FRUITS	13	1
VEGETABLES	7	20

The model yielded an accuracy of 0.81

Logistic Regression

Since We had close to 47 variables, we need to reduce the number of variables. This has been achieved using correlation matrix, where we have removed those variables which have correlation of more than 0.7.

RUnning the step function gives a formula, running the logistic model gives the following results:

Pseudo R^2 for logistic regression	R^2
Hosmer and Lemeshow R^2	0.848
Cox and Snell R^2	0.683
Nagelkerke R^2	0.92

Thus we can state that the model does a good job explaining the variance.

The Chi-square of the model is around 103.3667

The chi-square probability is 0

(Fit in the Mode over null is significant)

Result of ANOVA:

```
                Step Df  Deviance Resid. Df   Resid. Dev       AIC
1                   NA         NA        89 1.219069e+02 123.90687
2          + Sodium -1 44.974144        88 7.693273e+01  80.93273
3            + Zinc -1 19.002361        87 5.793037e+01  63.93037
4       + Total_Fat -1 29.359920        86 2.857045e+01  36.57045
5  + Choline__total -1  7.824349        85 2.074610e+01  30.74610
6   + Folate__total -1  3.768076        84 1.697802e+01  28.97802
7 + Carotene__alpha -1  4.375761        83 1.260226e+01  26.60226

8    + Carbohydrate -1 12.602262        82 8.964383e-08  16.00000
```

Gives the different AIC scores as we keep adding new features to the model. We chose the model that gave an AIC of 16.0