automatization\_notebook\_02

2023-12-31

# Чтение данных

В вашем варианте нужно использовать датасет food.

data <- read\_csv("data/raw/food.csv")  
glimpse(data)

## Rows: 7,083  
## Columns: 38  
## $ Category <chr> "Milk", "Milk", "Milk", "Milk", "Milk…  
## $ Description <chr> "Milk, human", "Milk, NFS", "Milk, wh…  
## $ `Nutrient Data Bank Number` <dbl> 11000000, 11100000, 11111000, 1111110…  
## $ `Data.Alpha Carotene` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ `Data.Beta Carotene` <dbl> 7, 4, 7, 7, 7, 1, 0, 3, 1, 3, 1, 2, 1…  
## $ `Data.Beta Cryptoxanthin` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Data.Carbohydrate <dbl> 6.89, 4.87, 4.67, 4.46, 4.67, 5.19, 4…  
## $ Data.Cholesterol <dbl> 14, 8, 12, 14, 12, 5, 2, 8, 5, 8, 5, …  
## $ Data.Choline <dbl> 16.0, 17.9, 17.8, 16.0, 17.8, 17.4, 1…  
## $ Data.Fiber <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ `Data.Lutein and Zeaxanthin` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Data.Lycopene <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Data.Niacin <dbl> 0.177, 0.110, 0.105, 0.043, 0.105, 0.…  
## $ Data.Protein <dbl> 1.03, 3.34, 3.28, 3.10, 3.28, 3.38, 3…  
## $ Data.Retinol <dbl> 60, 58, 31, 28, 31, 58, 137, 83, 58, …  
## $ Data.Riboflavin <dbl> 0.036, 0.137, 0.138, 0.105, 0.138, 0.…  
## $ Data.Selenium <dbl> 1.8, 1.9, 1.9, 2.0, 1.9, 2.1, 2.1, 1.…  
## $ `Data.Sugar Total` <dbl> 6.89, 4.89, 4.81, 4.46, 4.81, 4.96, 4…  
## $ Data.Thiamin <dbl> 0.014, 0.057, 0.056, 0.020, 0.056, 0.…  
## $ Data.Water <dbl> 87.50, 89.04, 88.10, 88.20, 88.10, 89…  
## $ `Data.Fat.Monosaturated Fat` <dbl> 1.658, 0.426, 0.688, 0.999, 0.688, 0.…  
## $ `Data.Fat.Polysaturated Fat` <dbl> 0.497, 0.065, 0.108, 0.128, 0.108, 0.…  
## $ `Data.Fat.Saturated Fat` <dbl> 2.009, 1.164, 1.860, 2.154, 1.860, 0.…  
## $ `Data.Fat.Total Lipid` <dbl> 4.38, 1.99, 3.20, 3.46, 3.20, 0.95, 0…  
## $ `Data.Major Minerals.Calcium` <dbl> 32, 126, 123, 101, 123, 126, 204, 126…  
## $ `Data.Major Minerals.Copper` <dbl> 0.052, 0.001, 0.001, 0.010, 0.001, 0.…  
## $ `Data.Major Minerals.Iron` <dbl> 0.03, 0.00, 0.00, 0.05, 0.00, 0.00, 0…  
## $ `Data.Major Minerals.Magnesium` <dbl> 3, 12, 12, 5, 12, 12, 11, 12, 12, 12,…  
## $ `Data.Major Minerals.Phosphorus` <dbl> 14, 103, 101, 86, 101, 103, 101, 103,…  
## $ `Data.Major Minerals.Potassium` <dbl> 51, 157, 150, 253, 150, 159, 166, 159…  
## $ `Data.Major Minerals.Sodium` <dbl> 17, 39, 38, 3, 38, 39, 52, 39, 39, 39…  
## $ `Data.Major Minerals.Zinc` <dbl> 0.17, 0.42, 0.41, 0.38, 0.41, 0.43, 0…  
## $ `Data.Vitamins.Vitamin A - RAE` <dbl> 61, 59, 32, 29, 32, 58, 137, 83, 58, …  
## $ `Data.Vitamins.Vitamin B12` <dbl> 0.05, 0.56, 0.54, 0.36, 0.54, 0.61, 0…  
## $ `Data.Vitamins.Vitamin B6` <dbl> 0.011, 0.060, 0.061, 0.034, 0.061, 0.…  
## $ `Data.Vitamins.Vitamin C` <dbl> 5.0, 0.1, 0.0, 0.9, 0.0, 0.0, 1.0, 0.…  
## $ `Data.Vitamins.Vitamin E` <dbl> 0.08, 0.03, 0.05, 0.08, 0.05, 0.02, 0…  
## $ `Data.Vitamins.Vitamin K` <dbl> 0.3, 0.2, 0.3, 0.3, 0.3, 0.1, 0.0, 0.…

# Выведите общее описание данных

summary(data)

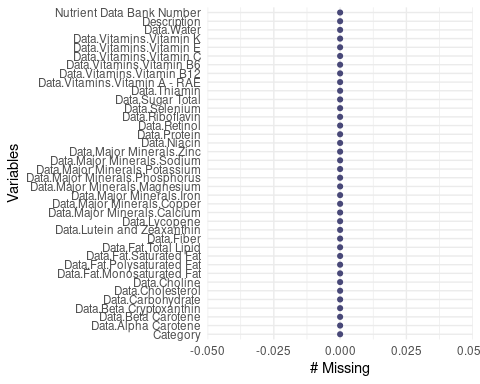
## Category Description Nutrient Data Bank Number  
## Length:7083 Length:7083 Min. :11000000   
## Class :character Class :character 1st Qu.:27150770   
## Mode :character Mode :character Median :53260200   
## Mean :48849646   
## 3rd Qu.:67203450   
## Max. :99998210   
## Data.Alpha Carotene Data.Beta Carotene Data.Beta Cryptoxanthin  
## Min. : 0.00 Min. : 0.0 Min. : 0.000   
## 1st Qu.: 0.00 1st Qu.: 0.0 1st Qu.: 0.000   
## Median : 0.00 Median : 8.0 Median : 0.000   
## Mean : 43.76 Mean : 255.4 Mean : 4.862   
## 3rd Qu.: 1.00 3rd Qu.: 73.0 3rd Qu.: 1.000   
## Max. :4655.00 Max. :14134.0 Max. :1922.000   
## Data.Carbohydrate Data.Cholesterol Data.Choline Data.Fiber   
## Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 5.65 1st Qu.: 0.00 1st Qu.: 10.00 1st Qu.: 0.100   
## Median : 13.30 Median : 8.00 Median : 19.60 Median : 1.000   
## Mean : 20.83 Mean : 34.46 Mean : 34.44 Mean : 1.704   
## 3rd Qu.: 26.20 3rd Qu.: 46.00 3rd Qu.: 44.20 3rd Qu.: 2.100   
## Max. :100.00 Max. :3074.00 Max. :820.20 Max. :46.200   
## Data.Lutein and Zeaxanthin Data.Lycopene Data.Niacin   
## Min. : 0.0 Min. : 0.0 Min. : 0.000   
## 1st Qu.: 0.0 1st Qu.: 0.0 1st Qu.: 0.535   
## Median : 18.0 Median : 0.0 Median : 1.487   
## Mean : 213.4 Mean : 263.6 Mean : 2.647   
## 3rd Qu.: 81.0 3rd Qu.: 0.0 3rd Qu.: 3.400   
## Max. :15643.0 Max. :45902.0 Max. :127.500   
## Data.Protein Data.Retinol Data.Riboflavin Data.Selenium   
## Min. : 0.000 Min. : 0.00 Min. : 0.0000 Min. : 0.00   
## 1st Qu.: 2.220 1st Qu.: 0.00 1st Qu.: 0.0600 1st Qu.: 1.70   
## Median : 6.190 Median : 8.00 Median : 0.1240 Median : 8.10   
## Mean : 8.599 Mean : 49.83 Mean : 0.1888 Mean : 13.09   
## 3rd Qu.:12.130 3rd Qu.: 43.00 3rd Qu.: 0.2200 3rd Qu.: 20.00   
## Max. :78.130 Max. :9349.00 Max. :17.5000 Max. :1917.00   
## Data.Sugar Total Data.Thiamin Data.Water Data.Fat.Monosaturated Fat  
## Min. : 0.000 Min. : 0.0000 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 0.780 1st Qu.: 0.0400 1st Qu.:45.90 1st Qu.: 0.513   
## Median : 2.390 Median : 0.0870 Median :66.59 Median : 1.869   
## Mean : 7.337 Mean : 0.1714 Mean :59.80 Mean : 3.218   
## 3rd Qu.: 7.380 3rd Qu.: 0.1890 3rd Qu.:80.61 3rd Qu.: 4.433   
## Max. :99.800 Max. :23.3750 Max. :99.98 Max. :75.221   
## Data.Fat.Polysaturated Fat Data.Fat.Saturated Fat Data.Fat.Total Lipid  
## Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 0.338 1st Qu.: 0.503 1st Qu.: 2.060   
## Median : 1.036 Median : 1.444 Median : 5.480   
## Mean : 2.174 Mean : 2.795 Mean : 8.958   
## 3rd Qu.: 2.625 3rd Qu.: 3.668 3rd Qu.: 12.650   
## Max. :67.849 Max. :82.500 Max. :100.000   
## Data.Major Minerals.Calcium Data.Major Minerals.Copper  
## Min. : 0.00 Min. : 0.0000   
## 1st Qu.: 14.00 1st Qu.: 0.0500   
## Median : 37.00 Median : 0.0790   
## Mean : 73.47 Mean : 0.1416   
## 3rd Qu.: 92.00 3rd Qu.: 0.1280   
## Max. :1375.00 Max. :14.4660   
## Data.Major Minerals.Iron Data.Major Minerals.Magnesium  
## Min. : 0.000 Min. : 0.00   
## 1st Qu.: 0.460 1st Qu.: 12.00   
## Median : 1.040 Median : 20.00   
## Mean : 1.752 Mean : 27.79   
## 3rd Qu.: 1.840 3rd Qu.: 29.00   
## Max. :64.100 Max. :611.00   
## Data.Major Minerals.Phosphorus Data.Major Minerals.Potassium  
## Min. : 0.0 Min. : 0.0   
## 1st Qu.: 48.0 1st Qu.: 111.0   
## Median : 102.0 Median : 183.0   
## Mean : 133.1 Mean : 217.1   
## 3rd Qu.: 189.0 3rd Qu.: 270.5   
## Max. :1429.0 Max. :6040.0   
## Data.Major Minerals.Sodium Data.Major Minerals.Zinc  
## Min. : 0.0 Min. : 0.000   
## 1st Qu.: 124.0 1st Qu.: 0.360   
## Median : 313.0 Median : 0.680   
## Mean : 340.5 Mean : 1.281   
## 3rd Qu.: 454.0 3rd Qu.: 1.360   
## Max. :7851.0 Max. :98.860   
## Data.Vitamins.Vitamin A - RAE Data.Vitamins.Vitamin B12  
## Min. : 0.00 Min. : 0.0000   
## 1st Qu.: 2.00 1st Qu.: 0.0000   
## Median : 20.00 Median : 0.1800   
## Mean : 73.14 Mean : 0.7052   
## 3rd Qu.: 61.00 3rd Qu.: 0.5500   
## Max. :9363.00 Max. :82.4400   
## Data.Vitamins.Vitamin B6 Data.Vitamins.Vitamin C Data.Vitamins.Vitamin E  
## Min. : 0.0000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 0.0520 1st Qu.: 0.000 1st Qu.: 0.230   
## Median : 0.1100 Median : 0.700 Median : 0.550   
## Mean : 0.1975 Mean : 5.696 Mean : 1.087   
## 3rd Qu.: 0.2030 3rd Qu.: 5.300 3rd Qu.: 1.110   
## Max. :12.0000 Max. :560.000 Max. :149.400   
## Data.Vitamins.Vitamin K  
## Min. : 0.00   
## 1st Qu.: 0.80   
## Median : 3.80   
## Mean : 14.21   
## 3rd Qu.: 9.30   
## Max. :1640.00

# Очистка данных

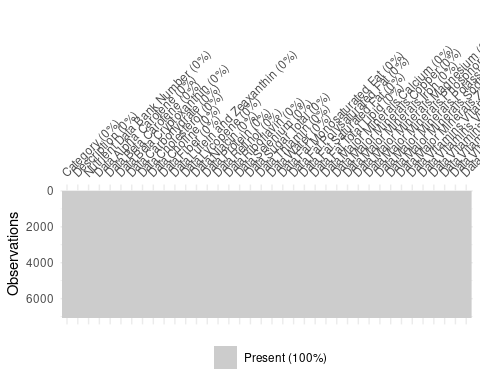
1. Уберите переменные, в которых пропущенных значений больше 20% или уберите субъектов со слишком большим количеством пропущенных значений. Или совместите оба варианта. Напишите обоснование, почему вы выбрали тот или иной вариант:

**Обоснование**: Мне повезло, и в датасете нет пропущенных данных. Также, судя по описательным статистикам и глимпсу, пока не похоже, чтобы наны были зашифрованы иначе. Но вообще я бы, во-первых, подумала о цели анализа - нельзя удалять ключевые переменные, даже если там высокий процент пропущенных значений, бездумно. Я бы также посмотрела на распределение пропущенных значений: если пропущенные значения сконцентрированны в некоторых переменных, логично было бы взять первый подход, если в некоторых измерениях (строках) - второй, ну и третий подход был бы оптимальным, если и переменные, и субъекты имеют значительные проблемы с пропущенными значениями.

library(naniar)  
gg\_miss\_var(data) # график, показывающий долю пропущенных данных для каждой переменной



vis\_miss(data) # хитмап помогает увидеть, где именно в данных находятся пропуски



1. Переименуйте переменные в человекочитаемый вид (что делать с пробелами в названиях?);

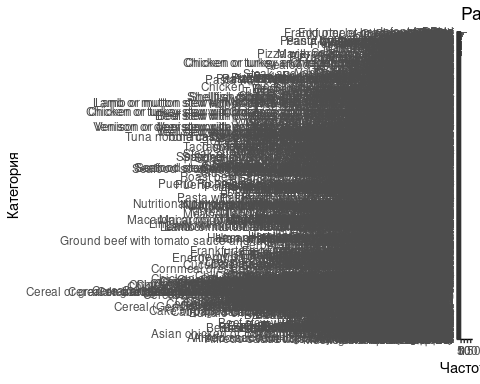
data <- data %>%   
 rename\_with(~ gsub("Data\\.", "", .x), everything()) %>% # Удаление избыточного 'Data.' из названий  
 rename\_with(~ gsub(" - ", "\_", .x), everything()) %>% # Замена ' - ' на '\_'  
 rename\_with(~ gsub(" ", "\_", .x), everything()) # Замена пробела на '\_'   
  
names(data)

## [1] "Category" "Description"   
## [3] "Nutrient\_Data\_Bank\_Number" "Alpha\_Carotene"   
## [5] "Beta\_Carotene" "Beta\_Cryptoxanthin"   
## [7] "Carbohydrate" "Cholesterol"   
## [9] "Choline" "Fiber"   
## [11] "Lutein\_and\_Zeaxanthin" "Lycopene"   
## [13] "Niacin" "Protein"   
## [15] "Retinol" "Riboflavin"   
## [17] "Selenium" "Sugar\_Total"   
## [19] "Thiamin" "Water"   
## [21] "Fat.Monosaturated\_Fat" "Fat.Polysaturated\_Fat"   
## [23] "Fat.Saturated\_Fat" "Fat.Total\_Lipid"   
## [25] "Major\_Minerals.Calcium" "Major\_Minerals.Copper"   
## [27] "Major\_Minerals.Iron" "Major\_Minerals.Magnesium"   
## [29] "Major\_Minerals.Phosphorus" "Major\_Minerals.Potassium"   
## [31] "Major\_Minerals.Sodium" "Major\_Minerals.Zinc"   
## [33] "Vitamins.Vitamin\_A\_RAE" "Vitamins.Vitamin\_B12"   
## [35] "Vitamins.Vitamin\_B6" "Vitamins.Vitamin\_C"   
## [37] "Vitamins.Vitamin\_E" "Vitamins.Vitamin\_K"

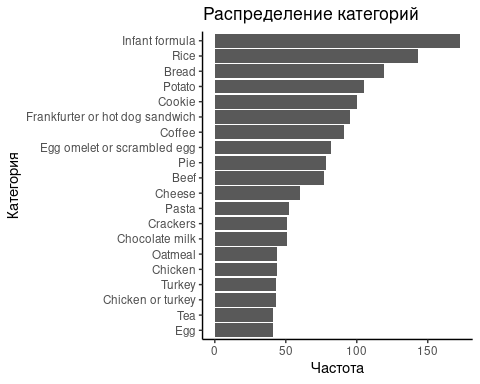
1. В соответствии с описанием данных приведите переменные к нужному типу (numeric или factor);

#Большинство колонок это числовые данные, кроме "Category" и "Description". Помимо этого, есть "Nutrient\_Data\_Bank\_Number" который явно не является числовым (в том смысле, что это идентификатор, который нельзя подвергнуть мат. операциям, например).   
  
data <- data %>%  
 mutate(  
 Category = as.factor(Category),  
 Description = as.factor(Description),  
 Nutrient\_Data\_Bank\_Number = as.factor(Nutrient\_Data\_Bank\_Number)  
 )

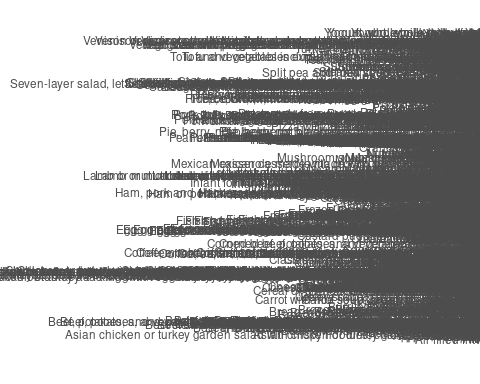
# Визуализируем Category  
data %>%  
 count(Category) %>%   
 arrange(desc(n)) %>%   
 ggplot(aes(x = reorder(Category, n), y = n)) +   
 geom\_col() + coord\_flip() +  
 labs(title = "Распределение категорий", x = "Категория", y = "Частота") +  
 theme\_classic()



# Так как категорий слишком много, давайте посмотрим на топ-20  
data %>%  
 count(Category) %>%   
 arrange(desc(n)) %>%   
 top\_n(20) %>%  
 ggplot(aes(x = reorder(Category, n), y = n)) +   
 geom\_col() + coord\_flip() +  
 labs(title = "Распределение категорий", x = "Категория", y = "Частота") +  
 theme\_classic()



# Описаний наверняка не меньше, чем категорий, так что тоже посмотрим на топ  
data %>%  
 count(Description) %>%   
 arrange(desc(n)) %>%   
 top\_n(20) %>%  
 ggplot(aes(x = reorder(Description, n), y = n)) +   
 geom\_col() + coord\_flip() +  
 labs(title = "Распределение описаний", x = "Описание", y = "Частота") +  
 theme\_classic()



# Судя по всему, в колонке Description все значения уникальны, поэтому давайте посмотрим, верно ли это и для каких колонок, чтобы далее не делать такие ужасные графики  
unique\_cases <- sapply(names(data), function(column\_name) {  
 column <- data[[column\_name]]  
 if (length(unique(column)) == nrow(data)) {  
 return(column\_name) # пишем имя столбца, если все значения уникальны  
 } else {  
 return(NA) # в ином случае нан  
 }  
})  
  
unique\_cases <- unique\_cases[!is.na(unique\_cases)]  
print(unique\_cases) # Видим, что и в случае Description, и в случае Nutrient\_Data\_Bank\_Number все значения уникальны, т.е. из банка нутриентов скорее всего взяли 7083 уникальных значений без дублей и затем их сгруппировали (либо они были уже сгруппированы оригинально)

## Description Nutrient\_Data\_Bank\_Number   
## "Description" "Nutrient\_Data\_Bank\_Number"

1. Отсортируйте данные по углеводам по убыванию;

# Углеводы это судя по всему колонка Carbohydrate  
data <- data %>%   
 arrange(desc(Carbohydrate))  
head(data)

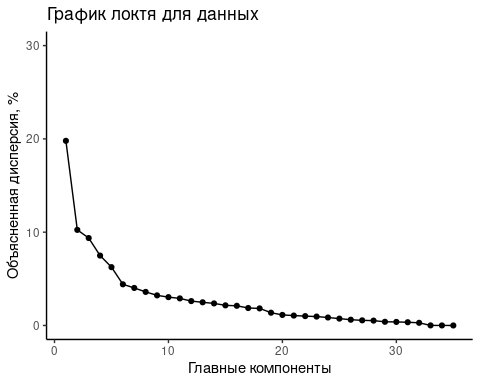
## # A tibble: 6 × 38  
## Category Description Nutrient\_Data\_Bank\_N…¹ Alpha\_Carotene Beta\_Carotene  
## <fct> <fct> <fct> <dbl> <dbl>  
## 1 Sugar substit… Sugar subs… 91108000 0 0  
## 2 Sugar substit… Sugar subs… 91108020 0 0  
## 3 Sugar Sugar, whi… 91101020 0 0  
## 4 Sugar Sugar, NFS 91101000 0 0  
## 5 Sugar Sugar, whi… 91101010 0 0  
## 6 Sugar substit… Sugar subs… 91106010 0 0  
## # ℹ abbreviated name: ¹​Nutrient\_Data\_Bank\_Number  
## # ℹ 33 more variables: Beta\_Cryptoxanthin <dbl>, Carbohydrate <dbl>,  
## # Cholesterol <dbl>, Choline <dbl>, Fiber <dbl>, Lutein\_and\_Zeaxanthin <dbl>,  
## # Lycopene <dbl>, Niacin <dbl>, Protein <dbl>, Retinol <dbl>,  
## # Riboflavin <dbl>, Selenium <dbl>, Sugar\_Total <dbl>, Thiamin <dbl>,  
## # Water <dbl>, Fat.Monosaturated\_Fat <dbl>, Fat.Polysaturated\_Fat <dbl>,  
## # Fat.Saturated\_Fat <dbl>, Fat.Total\_Lipid <dbl>, …

1. Сохраните в файл outliers.csv субъектов, которые являются выбросами (например, по правилу трёх сигм) — это необязательное задание со звёздочкой;

# Так как у меня преимущественно числовые данные, мне кажется, было бы уместно использовать PCA для визуализации и отсечения аутлаеров.   
# Давайте посмотрим на проценты объясненной дисперсии  
  
numeric\_data <- data %>% select\_if(is.numeric)  
pca\_result <- prcomp(numeric\_data, center = TRUE, scale. = TRUE)  
  
summary(pca\_result)

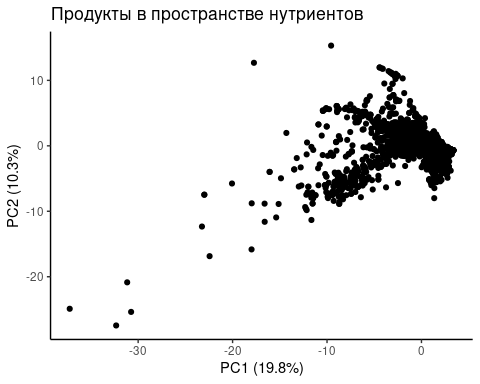
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 2.6320 1.8937 1.81119 1.6191 1.47932 1.24274 1.18629  
## Proportion of Variance 0.1979 0.1025 0.09373 0.0749 0.06253 0.04413 0.04021  
## Cumulative Proportion 0.1979 0.3004 0.39412 0.4690 0.53155 0.57567 0.61588  
## PC8 PC9 PC10 PC11 PC12 PC13 PC14  
## Standard deviation 1.12213 1.06277 1.03137 1.00782 0.95655 0.93240 0.90986  
## Proportion of Variance 0.03598 0.03227 0.03039 0.02902 0.02614 0.02484 0.02365  
## Cumulative Proportion 0.65186 0.68413 0.71452 0.74354 0.76969 0.79452 0.81818  
## PC15 PC16 PC17 PC18 PC19 PC20 PC21  
## Standard deviation 0.86876 0.8594 0.80928 0.8002 0.69189 0.62941 0.60969  
## Proportion of Variance 0.02156 0.0211 0.01871 0.0183 0.01368 0.01132 0.01062  
## Cumulative Proportion 0.83974 0.8608 0.87956 0.8979 0.91153 0.92285 0.93347  
## PC22 PC23 PC24 PC25 PC26 PC27 PC28  
## Standard deviation 0.59260 0.57803 0.54669 0.50767 0.46448 0.4389 0.42628  
## Proportion of Variance 0.01003 0.00955 0.00854 0.00736 0.00616 0.0055 0.00519  
## Cumulative Proportion 0.94350 0.95305 0.96159 0.96895 0.97512 0.9806 0.98581  
## PC29 PC30 PC31 PC32 PC33 PC34 PC35  
## Standard deviation 0.37213 0.3645 0.34860 0.31574 0.05305 0.03594 0.001226  
## Proportion of Variance 0.00396 0.0038 0.00347 0.00285 0.00008 0.00004 0.000000  
## Cumulative Proportion 0.98977 0.9936 0.99703 0.99988 0.99996 1.00000 1.000000

var\_explained = (pca\_result$sdev^2 / sum(pca\_result$sdev^2))\*100  
  
x\_values <- 1:length(var\_explained)  
  
ggplot(data = data.frame(PC = x\_values, Var = var\_explained), aes(x = PC, y = Var)) +   
 geom\_line() +   
 geom\_point() +   
 xlab("Главные компоненты") +   
 ylab("Объясненная дисперсия, %") +  
 ggtitle("График локтя для данных") +  
 ylim(0, 30)+  
 theme\_classic()



#Видим, что PCA достаточно плохо описывает данные, в первых двух компонентах всего 30% объясненной дисперсии, а 75% достигаются лишь на 11 компоненте. Но давайте построим скаттерплот по ним и посмотрим как распределяются образцы в этих координатах

PC1 <- pca\_result$x[,1]  
PC2 <- pca\_result$x[,2]  
ggplot(data = data,   
 aes(x = PC1,   
 y = PC2)) +  
 geom\_point()+   
 xlab("PC1 (19.8%)") +   
 ylab("PC2 (10.3%)")+  
 ggtitle("Продукты в пространстве нутриентов")+  
 theme\_classic()

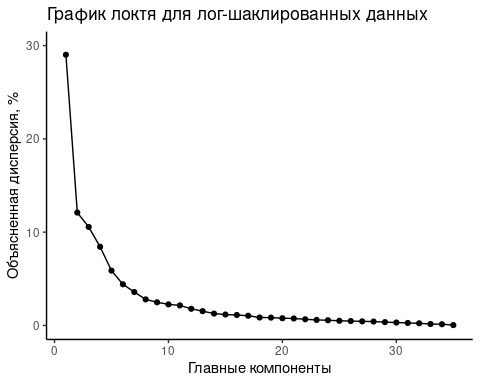


#Давайте попробуем лог-шкалировать данные, т.к. мы видели очень насыщенные нулями переменные с длинными хвостами (очень высокими max значениями), может быть это поможет с PCA

numeric\_data <- data %>% select\_if(is.numeric)  
numeric\_data <- log1p(numeric\_data) #лог-шкалируем, убирая 0 чтобы избежать отрицательных значений  
pca\_result <- prcomp(numeric\_data, center = TRUE, scale. = TRUE)  
  
summary(pca\_result)

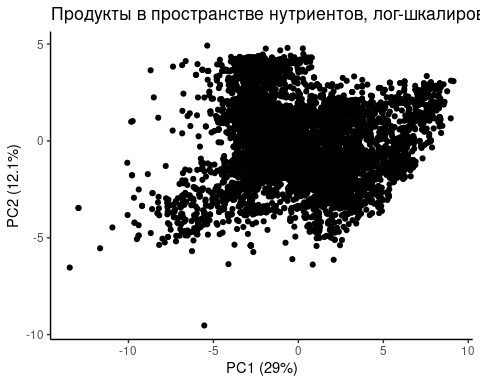
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 3.1871 2.0580 1.9223 1.71822 1.43435 1.24294 1.12047  
## Proportion of Variance 0.2902 0.1210 0.1056 0.08435 0.05878 0.04414 0.03587  
## Cumulative Proportion 0.2902 0.4112 0.5168 0.60116 0.65994 0.70408 0.73995  
## PC8 PC9 PC10 PC11 PC12 PC13 PC14  
## Standard deviation 0.98922 0.93281 0.88989 0.86764 0.79102 0.73268 0.66772  
## Proportion of Variance 0.02796 0.02486 0.02263 0.02151 0.01788 0.01534 0.01274  
## Cumulative Proportion 0.76791 0.79277 0.81539 0.83690 0.85478 0.87012 0.88286  
## PC15 PC16 PC17 PC18 PC19 PC20 PC21  
## Standard deviation 0.64015 0.62717 0.60374 0.54722 0.54247 0.52136 0.5122  
## Proportion of Variance 0.01171 0.01124 0.01041 0.00856 0.00841 0.00777 0.0075  
## Cumulative Proportion 0.89456 0.90580 0.91622 0.92477 0.93318 0.94095 0.9484  
## PC22 PC23 PC24 PC25 PC26 PC27 PC28  
## Standard deviation 0.48185 0.45575 0.44202 0.41921 0.40796 0.39203 0.38422  
## Proportion of Variance 0.00663 0.00593 0.00558 0.00502 0.00476 0.00439 0.00422  
## Cumulative Proportion 0.95508 0.96101 0.96659 0.97161 0.97637 0.98076 0.98498  
## PC29 PC30 PC31 PC32 PC33 PC34 PC35  
## Standard deviation 0.35355 0.33172 0.30774 0.28503 0.23325 0.21566 0.1177  
## Proportion of Variance 0.00357 0.00314 0.00271 0.00232 0.00155 0.00133 0.0004  
## Cumulative Proportion 0.98855 0.99169 0.99440 0.99672 0.99828 0.99960 1.0000

var\_explained = (pca\_result$sdev^2 / sum(pca\_result$sdev^2))\*100  
  
x\_values <- 1:length(var\_explained)  
  
ggplot(data = data.frame(PC = x\_values, Var = var\_explained), aes(x = PC, y = Var)) +   
 geom\_line() +   
 geom\_point() +   
 xlab("Главные компоненты") +   
 ylab("Объясненная дисперсия, %") +  
 ggtitle("График локтя для лог-шаклированных данных") +  
 ylim(0, 30)+  
 theme\_classic()



#Это действительно помогло, теперь первые 2 компоненты объясняют больше дисперсии, а 75% достигается на 9-й компоненте. Это все еще мало, но, думаю, может подойти для отсечения аутлаеров. Давайте также визуализируем скаттер по PC2 и 1.

PC1 <- pca\_result$x[,1]  
PC2 <- pca\_result$x[,2]  
ggplot(data = data,   
 aes(x = PC1,   
 y = PC2)) +  
 geom\_point()+   
 xlab("PC1 (29%)") +   
 ylab("PC2 (12.1%)")+  
 ggtitle("Продукты в пространстве нутриентов, лог-шкалированные")+  
 theme\_classic()

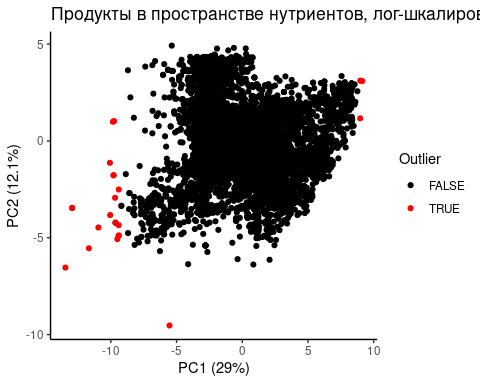


#Видим, что аутлаеров явно стало меньше и нам стало лучше видно "облако". Давайте попробуем итеративный PCA, будем брать первые 2 компоненты, вычленять продукты с median deviation больше 4.5 (подобранная отсечка, т.к. стандартная 6 не включала некоторых явных аутлаеров), пока таких измерений не станет. Все это будем делать в лог-шкалированном пространстве. На мой взгляд все способы определения аутлаеров субъективны, и этот способ нравится мне, хотя некоторым может показать экзотичным, поскольку мне кажется, что выбросы, определяемые таким способом, действительно выбросы(в отличие от isolation forest или некоторых стат тестов); и мне нравится, что не нужно отдавать приоритет каким-либо переменным, за меня это делает PCA (как в случае с отсечением по квантилям, например)

library(stats)  
  
numeric\_data <- data %>% select\_if(is.numeric)  
numeric\_data <- log1p(numeric\_data)  
df <- as.matrix(numeric\_data)  
original\_row\_count <- nrow(df)  
  
outliers\_mask <- rep(FALSE, original\_row\_count)  
  
while(TRUE) {  
 pca\_result <- prcomp(df, center = TRUE, scale. = TRUE)  
 transformed\_data <- pca\_result$x[, 1:2]  
  
 medians <- apply(transformed\_data, 2, median)  
 mads <- apply(transformed\_data, 2, mad)  
  
 scores <- abs((transformed\_data - medians) / mads)  
 outliers\_idx <- apply(scores, 1, function(x) any(x > 4.5))  
  
 if(any(outliers\_idx)) {  
 outliers\_mask[which(outliers\_mask == FALSE)] <- outliers\_idx  
 df <- df[!outliers\_idx, ]  
 } else {  
 break  
 }  
}  
  
outliers\_counts <- table(outliers\_mask)  
  
print(outliers\_counts) #Аутлаеров получилось 23, давайте визуализируем их и сохраним

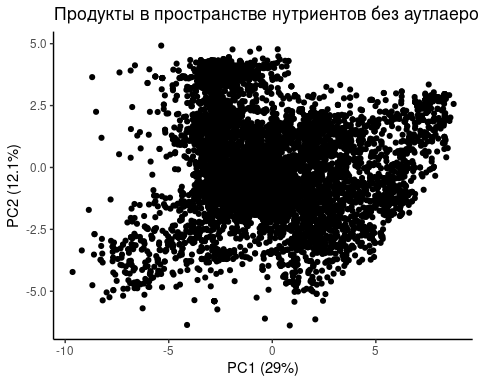
## outliers\_mask  
## FALSE TRUE   
## 7060 23

pca\_result <- prcomp(numeric\_data, center = TRUE, scale. = TRUE)  
  
pca\_data <- data.frame(PC1 = pca\_result$x[, 1], PC2 = pca\_result$x[, 2], Outlier = outliers\_mask)  
  
ggplot(pca\_data, aes(x = PC1, y = PC2, color = Outlier)) +  
 geom\_point() +  
 xlab("PC1 (29%)") +   
 ylab("PC2 (12.1%)") +  
 ggtitle("Продукты в пространстве нутриентов, лог-шкалированные") +  
 theme\_classic() +  
 scale\_color\_manual(values = c("black", "red")) # Аутлаеры красим красным



#Видим выбросы, особенно в левом углу -- достаточно очевидные

ggplot(pca\_data[!pca\_data$Outlier, ], aes(x = PC1, y = PC2)) +  
 geom\_point() +  
 xlab("PC1 (29%)") +   
 ylab("PC2 (12.1%)") +  
 ggtitle("Продукты в пространстве нутриентов без аутлаеров, лог-шкалированные") +  
 theme\_classic()



# График без аутлаеров выглядит действительно гораздо лучше, хотя явно облако группируется на некие батчи, разбираться глубже не буду, но было бы интересно, возможно, жиры, углеводы и белки могут группироваться в разных частях пространства.

outliers\_data <- data[outliers\_mask, ]  
write.csv(outliers\_data, "data/output/outliers.csv", row.names = FALSE)  
# Видим, что в аутлаеры вошла вода, соки, бычья печень, хлопья и бары -- для некоторых не очень понятно, почему

1. Отфильтруйте датасет так, чтобы остались только Rice и Cookie (переменная Category и есть группирующая);
2. Присвойте получившийся датасет переменной “cleaned\_data”.

cleaned\_data <- data %>%   
 filter(Category %in% c("Rice", "Cookie"))  
head(cleaned\_data)

## # A tibble: 6 × 38  
## Category Description Nutrient\_Data\_Bank\_N…¹ Alpha\_Carotene Beta\_Carotene  
## <fct> <fct> <fct> <dbl> <dbl>  
## 1 Cookie Cookie, fortune 53222010 0 0  
## 2 Cookie Cookie, tea, Jap… 53246000 0 0  
## 3 Cookie Cookie, rice, ba… 53803300 0 0  
## 4 Cookie Cookie, marshmal… 53226500 0 0  
## 5 Cookie Cookie, marshmal… 53226600 0 0  
## 6 Cookie Cookie, peanut b… 53234250 0 0  
## # ℹ abbreviated name: ¹​Nutrient\_Data\_Bank\_Number  
## # ℹ 33 more variables: Beta\_Cryptoxanthin <dbl>, Carbohydrate <dbl>,  
## # Cholesterol <dbl>, Choline <dbl>, Fiber <dbl>, Lutein\_and\_Zeaxanthin <dbl>,  
## # Lycopene <dbl>, Niacin <dbl>, Protein <dbl>, Retinol <dbl>,  
## # Riboflavin <dbl>, Selenium <dbl>, Sugar\_Total <dbl>, Thiamin <dbl>,  
## # Water <dbl>, Fat.Monosaturated\_Fat <dbl>, Fat.Polysaturated\_Fat <dbl>,  
## # Fat.Saturated\_Fat <dbl>, Fat.Total\_Lipid <dbl>, …

# Сколько осталось переменных?

#Кол-во колонок не поменялось, как, мне кажется, и должно быть. Их было и осталось 38  
num\_variables\_old <- ncol(data)  
num\_variables\_new <- ncol(cleaned\_data)  
print(num\_variables\_old)

## [1] 38

print(num\_variables\_new)

## [1] 38

# Сколько осталось случаев?

#Кол-во строк сильно уменьшилось, как мы бы и ожидали. Было 7083, стало 243.  
num\_rows\_old <- nrow(data)  
num\_rows\_new <- nrow(cleaned\_data)  
print(num\_rows\_old)

## [1] 7083

print(num\_rows\_new)

## [1] 243

# Есть ли в данных идентичные строки?

#Видим, что дублей нет  
duplicates\_old <- any(duplicated(data))  
duplicates\_new <- any(duplicated(cleaned\_data))  
  
print(duplicates\_old)

## [1] FALSE

print(duplicates\_new)

## [1] FALSE

# Сколько всего переменных с пропущенными значениями в данных и сколько пропущенных точек в каждой такой переменной?

# С помощью naniar выше выяснили, что пропусков, слава богам, нет.

# Описательные статистики

## Количественные переменные

1. Рассчитайте для всех количественных переменных для каждой группы (Category):

1.1) Количество значений;

1.2) Количество пропущенных значений;

1.3) Среднее;

1.4) Медиану;

1.5) Стандартное отклонение;

1.6) 25% квантиль и 75% квантиль;

1.7) Интерквартильный размах;

1.8) Минимум;

1.9) Максимум;

1.10) 95% ДИ для среднего - задание со звёздочкой.

# Для cleaned дата все считается хорошо, а для всех категори для SD и CI есть пропуски, из-за того что, видимо, SD выходит корнем из 0 в группе из 1 сэмпла (среднее и есть этот сэмпл). Думаю, т.к. мы получили cleaned data, преполагается, что работаем дальше с ним, что и буду делать, но на всякий оставлю вычисления и для data  
descriptive\_numeric\_stats <- cleaned\_data %>%   
 group\_by(Category) %>%   
 summarise(across(where(is.numeric), list(  
 count = ~sum(!is.na(.)),  
 missing = ~sum(is.na(.)),  
 mean = ~mean(., na.rm = TRUE),  
 median = ~median(., na.rm = TRUE),  
 sd = ~if (sum(!is.na(.)) > 1) sd(., na.rm = TRUE) else NA\_real\_,  
 quantile\_25 = ~quantile(., 0.25, na.rm = TRUE),  
 quantile\_75 = ~quantile(., 0.75, na.rm = TRUE),  
 IQR = ~IQR(., na.rm = TRUE),  
 min = ~min(., na.rm = TRUE),  
 max = ~max(., na.rm = TRUE),  
 CI95\_lower = ~if (sum(!is.na(.)) > 1) mean(., na.rm = TRUE) - 1.96 \* (sd(., na.rm = TRUE) / sqrt(sum(!is.na(.)))) else NA\_real\_,  
 CI95\_upper = ~if (sum(!is.na(.)) > 1) mean(., na.rm = TRUE) + 1.96 \* (sd(., na.rm = TRUE) / sqrt(sum(!is.na(.)))) else NA\_real\_  
 )))  
  
  
print(descriptive\_numeric\_stats)

## # A tibble: 2 × 421  
## Category Alpha\_Carotene\_count Alpha\_Carotene\_missing Alpha\_Carotene\_mean  
## <fct> <int> <int> <dbl>  
## 1 Cookie 100 0 8.25  
## 2 Rice 143 0 125.   
## # ℹ 417 more variables: Alpha\_Carotene\_median <dbl>, Alpha\_Carotene\_sd <dbl>,  
## # Alpha\_Carotene\_quantile\_25 <dbl>, Alpha\_Carotene\_quantile\_75 <dbl>,  
## # Alpha\_Carotene\_IQR <dbl>, Alpha\_Carotene\_min <dbl>,  
## # Alpha\_Carotene\_max <dbl>, Alpha\_Carotene\_CI95\_lower <dbl>,  
## # Alpha\_Carotene\_CI95\_upper <dbl>, Beta\_Carotene\_count <int>,  
## # Beta\_Carotene\_missing <int>, Beta\_Carotene\_mean <dbl>,  
## # Beta\_Carotene\_median <dbl>, Beta\_Carotene\_sd <dbl>, …

# Для whole data  
  
descriptive\_numeric\_stats\_whole <- data %>%   
 group\_by(Category) %>%   
 summarise(across(where(is.numeric), list(  
 count = ~sum(!is.na(.)),  
 missing = ~sum(is.na(.)),  
 mean = ~mean(., na.rm = TRUE),  
 median = ~median(., na.rm = TRUE),  
 sd = ~if (sum(!is.na(.)) > 1) sd(., na.rm = TRUE) else NA\_real\_,  
 quantile\_25 = ~quantile(., 0.25, na.rm = TRUE),  
 quantile\_75 = ~quantile(., 0.75, na.rm = TRUE),  
 IQR = ~IQR(., na.rm = TRUE),  
 min = ~min(., na.rm = TRUE),  
 max = ~max(., na.rm = TRUE),  
 CI95\_lower = ~if (sum(!is.na(.)) > 1) mean(., na.rm = TRUE) - 1.96 \* (sd(., na.rm = TRUE) / sqrt(sum(!is.na(.)))) else NA\_real\_,  
 CI95\_upper = ~if (sum(!is.na(.)) > 1) mean(., na.rm = TRUE) + 1.96 \* (sd(., na.rm = TRUE) / sqrt(sum(!is.na(.)))) else NA\_real\_  
 )))  
  
  
print(descriptive\_numeric\_stats\_whole)

## # A tibble: 2,429 × 421  
## Category Alpha\_Carotene\_count Alpha\_Carotene\_missing Alpha\_Carotene\_mean  
## <fct> <int> <int> <dbl>  
## 1 100 GRAND Bar 1 0 0  
## 2 3 MUSKETEERS… 1 0 0  
## 3 3 Musketeers… 1 0 0  
## 4 Abalone 3 0 0  
## 5 Adobo 2 0 0  
## 6 Agave liquid… 1 0 0  
## 7 Air filled f… 1 0 0  
## 8 Alcoholic ma… 3 0 0  
## 9 Alexander 1 0 0  
## 10 Alfalfa spro… 1 0 6  
## # ℹ 2,419 more rows  
## # ℹ 417 more variables: Alpha\_Carotene\_median <dbl>, Alpha\_Carotene\_sd <dbl>,  
## # Alpha\_Carotene\_quantile\_25 <dbl>, Alpha\_Carotene\_quantile\_75 <dbl>,  
## # Alpha\_Carotene\_IQR <dbl>, Alpha\_Carotene\_min <dbl>,  
## # Alpha\_Carotene\_max <dbl>, Alpha\_Carotene\_CI95\_lower <dbl>,  
## # Alpha\_Carotene\_CI95\_upper <dbl>, Beta\_Carotene\_count <int>,  
## # Beta\_Carotene\_missing <int>, Beta\_Carotene\_mean <dbl>, …

## Категориальные переменные

1. Рассчитайте для всех категориальных переменных для каждой группы (Category):

1.1) Абсолютное количество;

1.2) Относительное количество внутри группы;

1.3) 95% ДИ для доли внутри группы - задание со звёздочкой.

categorical\_data <- cleaned\_data %>%   
 select(where(~is.factor(.) | is.character(.)))  
  
categorical\_stats <- categorical\_data %>%   
 pivot\_longer(cols = c('Description','Nutrient\_Data\_Bank\_Number'), names\_to = "variable", values\_to = "value") %>%  
 group\_by(Category, variable, value) %>%  
 summarise(  
 count = n(),   
 proportion = n() / sum(n()),   
 .groups = 'drop'  
 ) %>%  
 group\_by(Category, variable) %>%  
 mutate(  
 CI95\_lower = ifelse(proportion > 0, proportion - 1.96 \* sqrt(proportion \* (1 - proportion) / sum(count)), 0),  
 CI95\_upper = ifelse(proportion > 0, proportion + 1.96 \* sqrt(proportion \* (1 - proportion) / sum(count)), 0)# Не уверена, правильно ли считаю тут ДИ  
 )  
  
# Мы уже знаем, что все переменные в категориях кроме нашей группирующей уникальные, но вроде бы код работает, правда проверить это сложно. ДИ скорее всего посчитаны неправильно, т.к. они должны быть равны 0 (их просто должно быть нельзя рассчитать)  
# Не буду делать то же для полного датафрейма, т.к. результат понятен  
print(categorical\_stats)

## # A tibble: 486 × 7  
## # Groups: Category, variable [4]  
## Category variable value count proportion CI95\_lower CI95\_upper  
## <fct> <chr> <fct> <int> <dbl> <dbl> <dbl>  
## 1 Cookie Description Cookie, almond 1 1 1 1  
## 2 Cookie Description Cookie, animal 1 1 1 1  
## 3 Cookie Description Cookie, animal, … 1 1 1 1  
## 4 Cookie Description Cookie, applesau… 1 1 1 1  
## 5 Cookie Description Cookie, baby food 1 1 1 1  
## 6 Cookie Description Cookie, bar, wit… 1 1 1 1  
## 7 Cookie Description Cookie, batter o… 1 1 1 1  
## 8 Cookie Description Cookie, biscotti 1 1 1 1  
## 9 Cookie Description Cookie, brownie,… 1 1 1 1  
## 10 Cookie Description Cookie, brownie,… 1 1 1 1  
## # ℹ 476 more rows

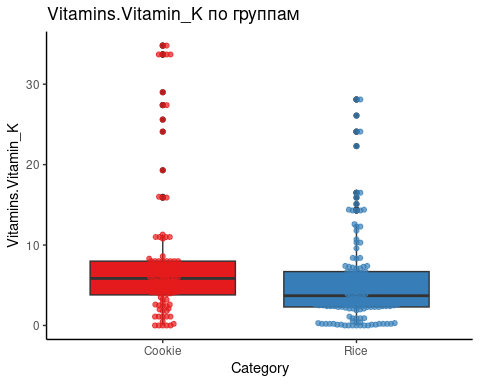
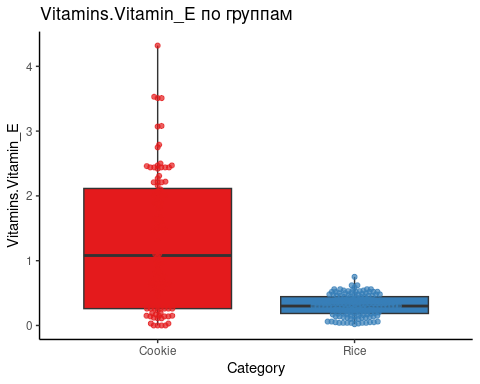
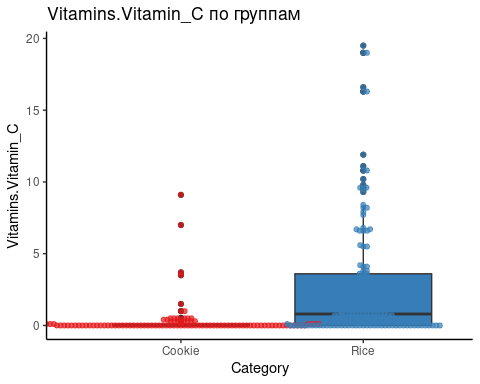
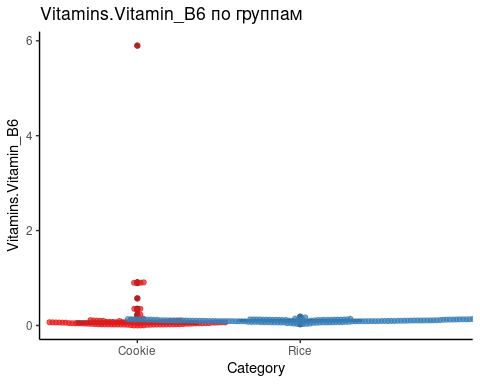
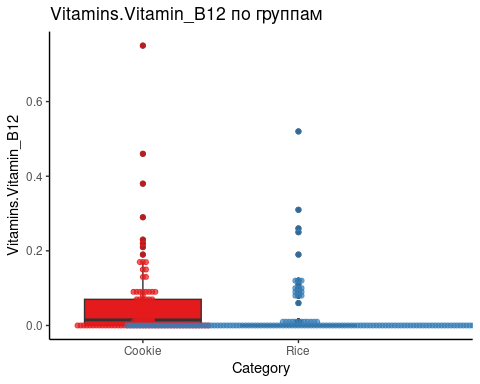
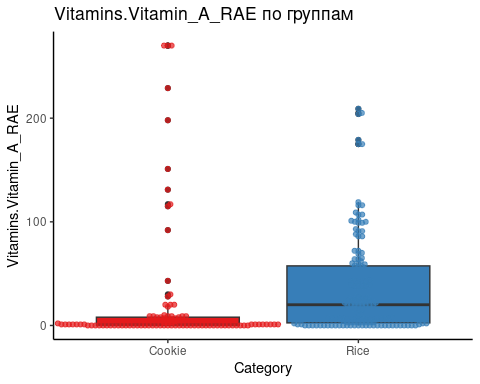
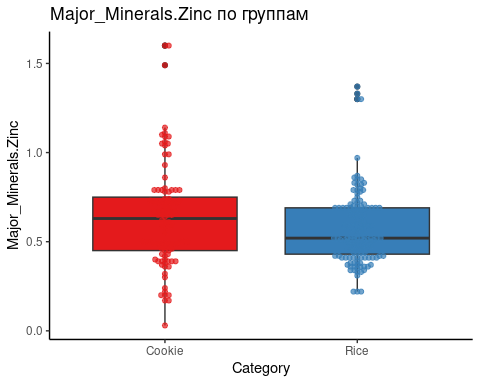
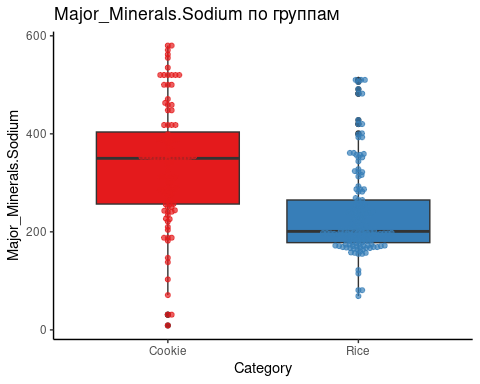
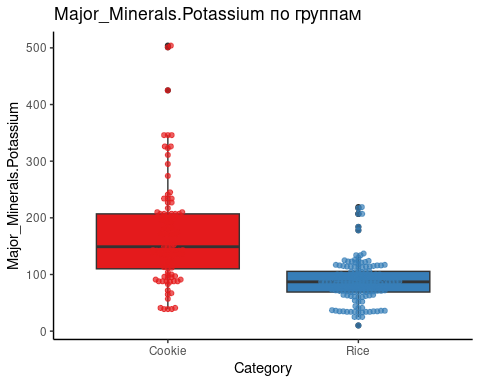
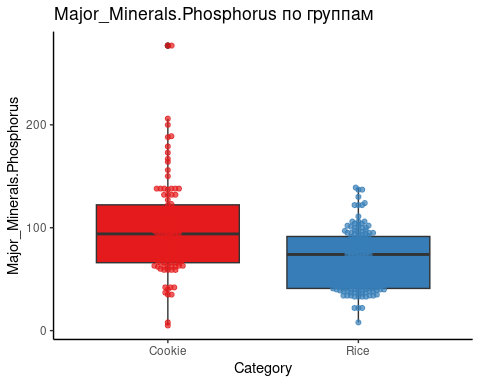
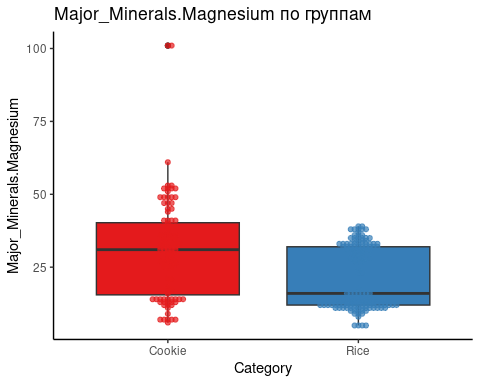
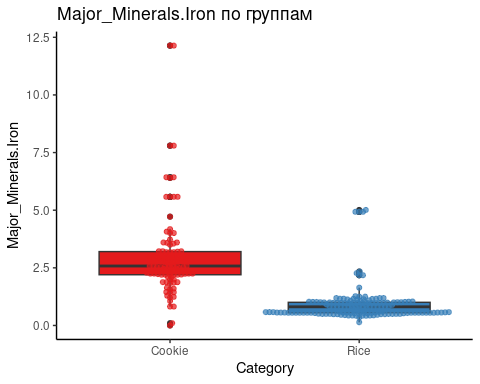
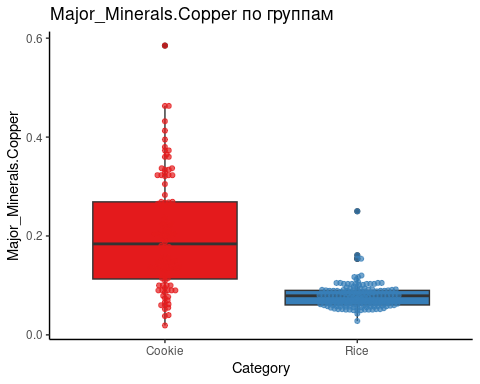
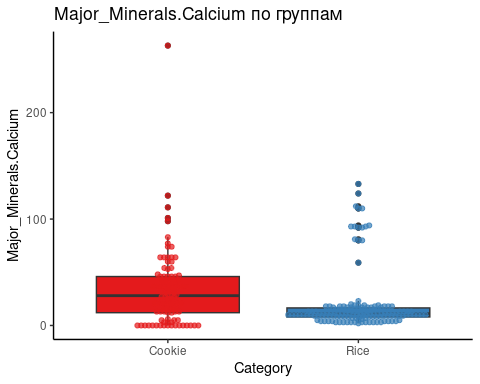
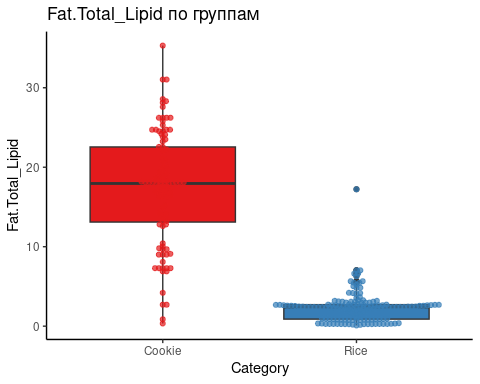
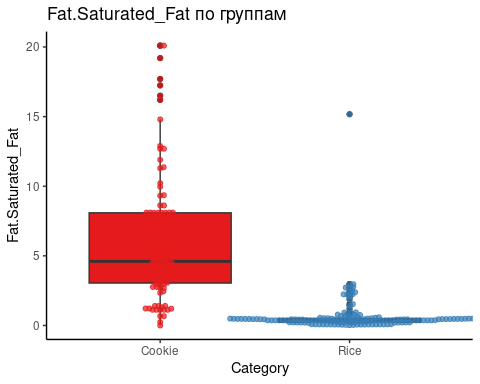
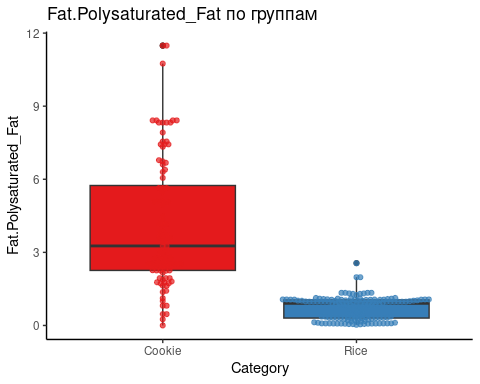
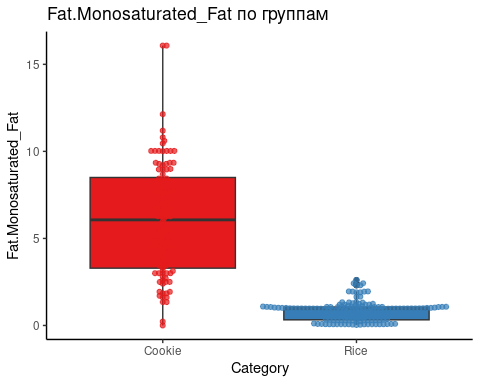
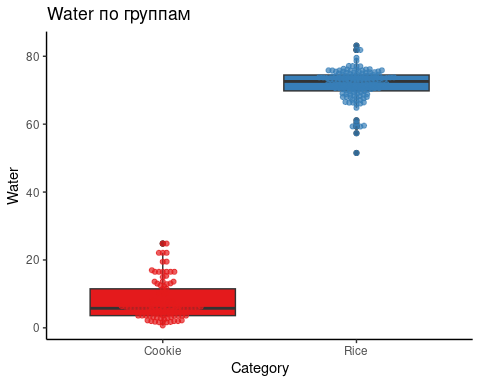
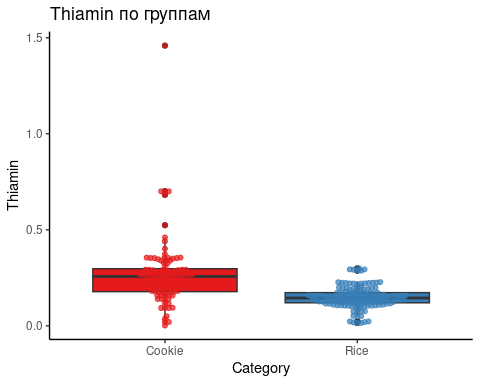
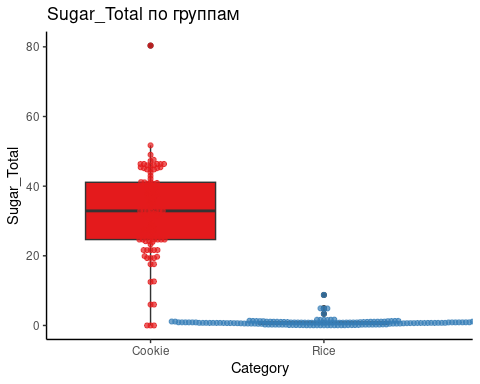
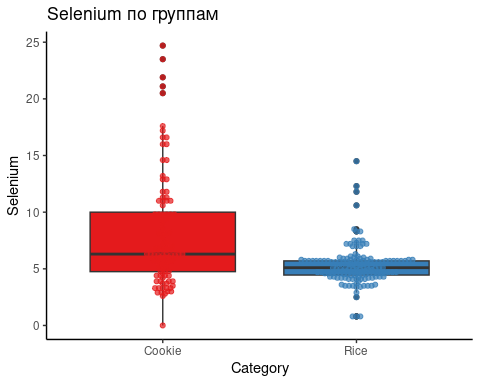
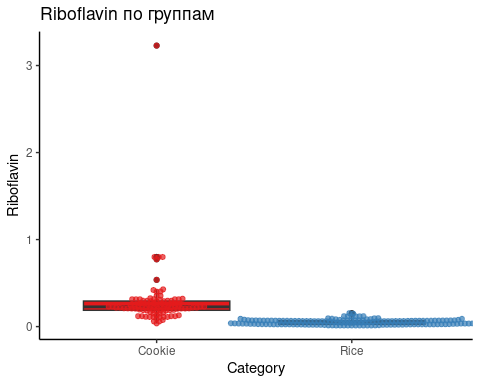
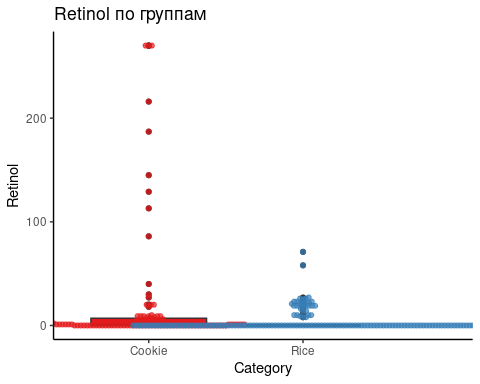
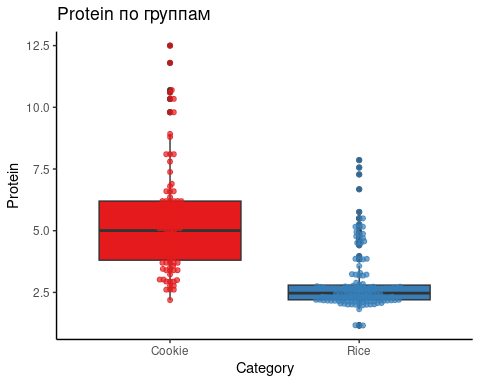
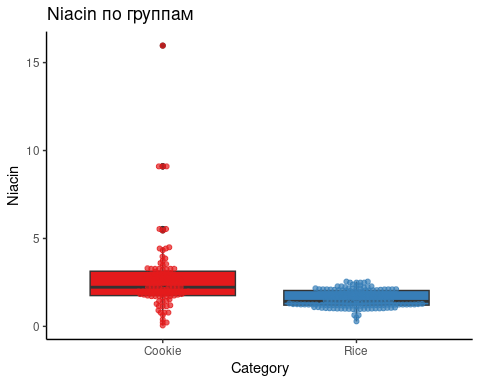
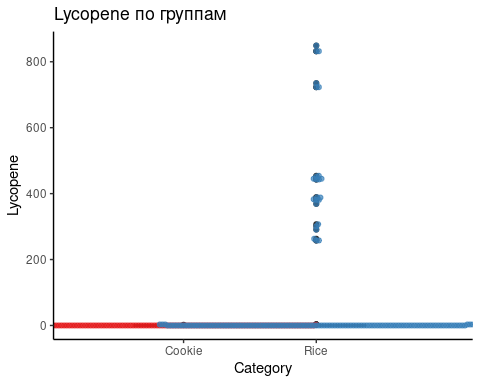
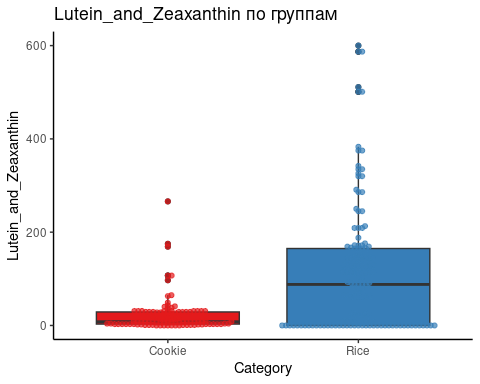
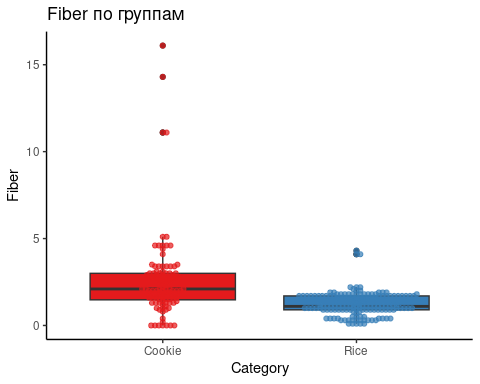
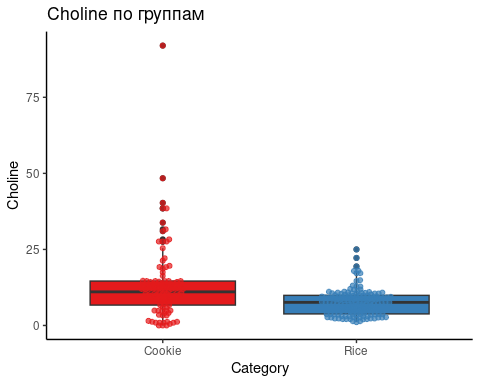
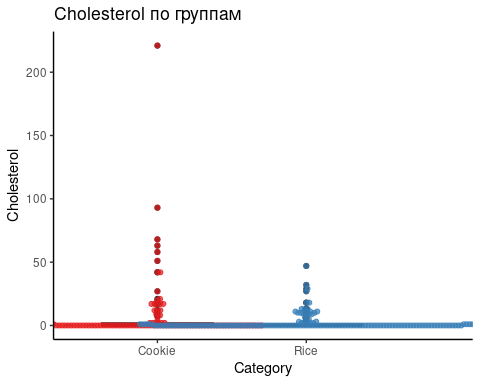
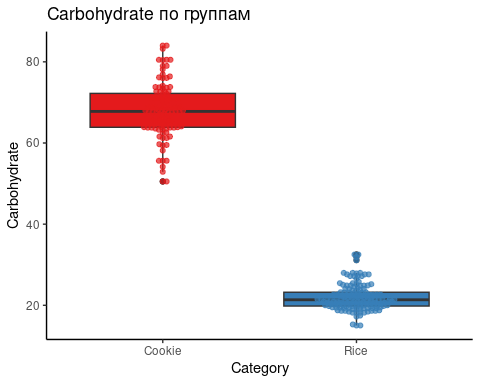
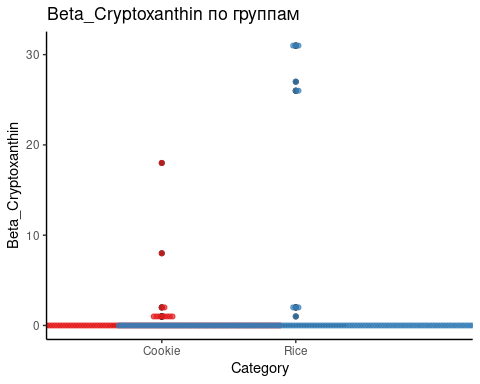
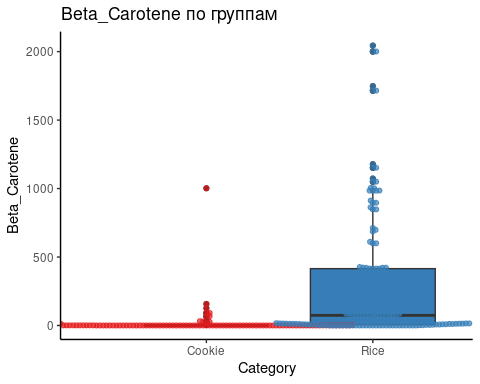
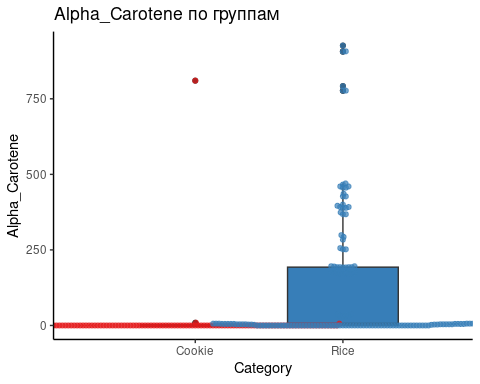
# Визуализация

## Количественные переменные

1. Для каждой количественной переменой сделайте боксплоты по группам. Расположите их либо на отдельных рисунках, либо на одном, но читаемо;
2. Наложите на боксплоты beeplots - задание со звёздочкой.
3. Раскрасьте боксплоты с помощью библиотеки RColorBrewer.

#Думаю, здесь точно нужно использовать cleaned\_data, т.к. иначе категорий будет слишком много  
  
library(RColorBrewer)  
library(ggbeeswarm)  
  
numeric\_data <- cleaned\_data %>% select(where(is.numeric))  
numeric\_vars <- names(numeric\_data)  
  
for (var in numeric\_vars) {  
 plot <- ggplot(cleaned\_data, aes\_string(x = 'Category', y = var, fill = 'Category')) +  
 geom\_boxplot() +  
 geom\_beeswarm(aes(color = Category), size = 1.5, alpha = 0.7) +  
 scale\_fill\_brewer(palette = "Set1") +  
 scale\_color\_brewer(palette = "Set1") +  
 theme\_classic() +  
 ggtitle(paste(var, "по группам")) +  
 theme(legend.position = "none")  
   
 print(plot)  
}

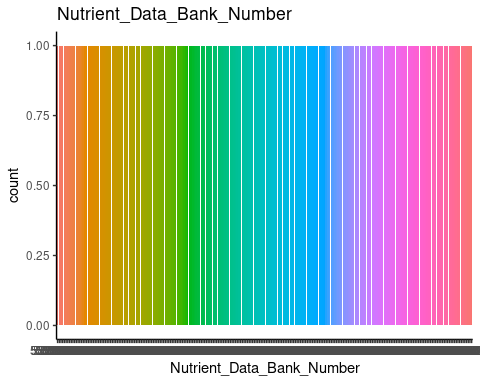
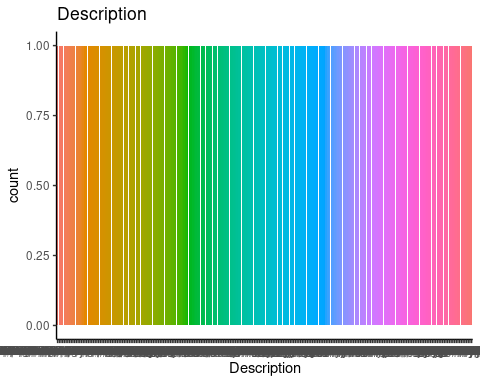
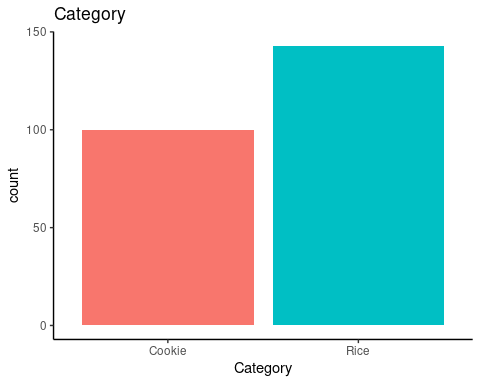
## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



## Категориальные переменные

1. Сделайте подходящие визуализации категориальных переменных. Обоснуйте, почему выбрали именно этот тип.

# Давайте используем столбчатую диаграмму. Она подходит для сравнения групп лучше, чем, например, пай чарт, поскольку в последнем сложно бывает сравнивать площади. Кроме того, в двух группах у нас сплошь уникальные значения, которые будут красиво переливаться.  
  
categorical\_vars <- names(cleaned\_data)[sapply(cleaned\_data, is.factor)]  
for (var in categorical\_vars) {  
 plot <- ggplot(cleaned\_data, aes\_string(x = var, fill = var)) +  
 geom\_bar() +  
 theme\_classic() +  
 ggtitle(var) +  
 theme(legend.position = "none")  
   
 print(plot)  
}



# Статистические оценки

## Проверка на нормальность

1. Оцените каждую переменную на соответствие нормальному распределению с помощью теста Шапиро-Уилка. Какие из переменных являются нормальными и как как вы это поняли?

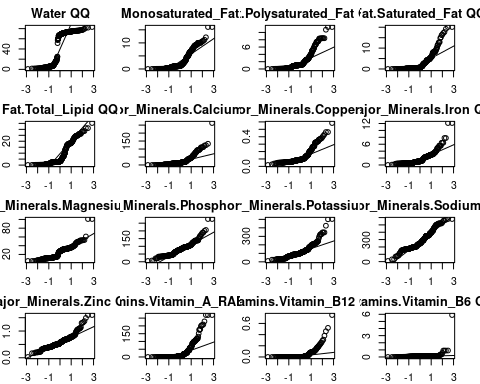
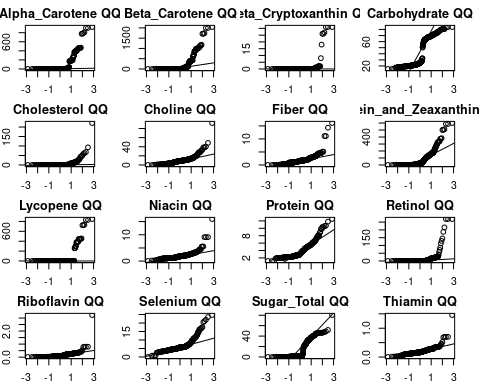
shapiro\_test\_results <- lapply(numeric\_data, function(x) {shapiro.test(x)})  
  
normality\_test\_results <- sapply(shapiro\_test\_results, function(result) {result$p.value >= 0.05})  
print(table(normality\_test\_results))

## normality\_test\_results  
## FALSE   
## 35

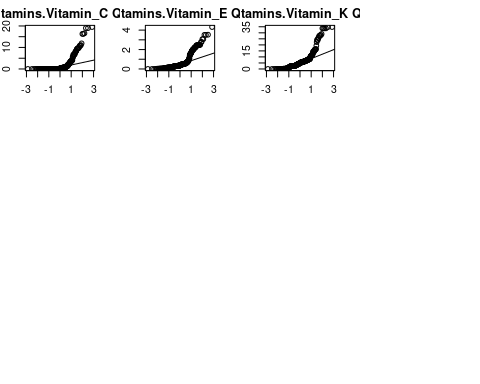
# Все переменные не являются нормальными по этому тесту (p теста больше 0.05). Этот тест очень чувствителен к малейшим отклонениям от нормальности, поэтому например qq плот может быть лучше для оценки нормальности.

1. Постройте для каждой количественной переменной QQ-плот. Отличаются ли выводы от теста Шапиро-Уилка? Какой метод вы бы предпочли и почему?

par(mar = c(2, 2, 2, 2))  
par(mfrow = c(4, 4))  
for (var in numeric\_vars) {  
 qqnorm(numeric\_data[[var]], main = paste(var, "QQ"))  
 qqline(numeric\_data[[var]])  
}



# Выводы отличаются, как минимум для переменных Sodium и Zinс даже для нешкалированных данных распределение явно близко к нормальному, что не отметил тест. Я бы предпочла QQ плоты, поскольку при работе с реальными данными шум это достаточно стандартная вещь, но тесты слишком чувствительны, а на итоговых оценках такие отклонения от нормальности (особенно после того, как мы уберем выбросы) не　должны　сильно сказаться



1. Ниже напишите, какие ещё методы проверки на нормальность вы знаете и какие у них есть ограничения.

Знаю тест Колмогорова-Cмирнова, который сравнивает эмпирическое распределение данных с теоретическим нормальным распределением. Он также очень чувствителен, особенно к выбросам в центральной части распределения, и в этом его недостаток. Также можно использовать другие визуальные методы, например kdeplot (он строит гистограмму, делая ее похожей на график плотности вероятности), что может дать представление о распределении, но это очень субъективный метод без определенных критериев и в этом его большое ограничение.

## Сравнение групп

1. Сравните группы (переменная **Category**) по каждой переменной (как количественной, так и категориальной). Для каждой переменной выберите нужный критерий и кратко обоснуйте его выбор в комментариях.

# 1) Я применю t-test для переменных "Major\_Minerals.Magnesium", "Major\_Minerals.Phosphorus", "Major\_Minerals.Sodium", "Major\_Minerals.Zinc", "Fiber", "Niacin", "Thiamin" после отсечения 0.1 и 0.9 перцентилей, поскольку они прошли "проверку" на нормальность по QQ плотам  
  
# 2) Для остальных численных переменных я применю mann whitney u test, непараметрический аналог t test-а, поскольку для них распределение не нормальное  
  
# 3) Для категориальных данных я бы применила хи-квадрат, но все переменные уникальны, поэтому пересечений не будет, и я не буду проводить эти сравнения, посколько в итоге все равно мы не получим результата  
  
t\_test\_vars <- c("Major\_Minerals.Magnesium", "Major\_Minerals.Phosphorus", "Major\_Minerals.Sodium", "Major\_Minerals.Zinc", "Fiber", "Niacin", "Thiamin")  
mann\_whitney\_vars <- setdiff(names(numeric\_data), t\_test\_vars)  
  
test\_results <- data.frame()  
  
# t-тест  
for (var in t\_test\_vars) {  
 data\_for\_test <- cleaned\_data %>%  
 filter(between(!!sym(var), quantile(!!sym(var), 0.1, na.rm = TRUE), quantile(!!sym(var), 0.9, na.rm = TRUE))) %>%  
 select(Category, !!sym(var))  
   
 test\_result <- broom::tidy(t.test(reformulate("Category", response = var), data = data\_for\_test))  
 test\_result <- test\_result %>% select(p.value) %>% mutate(variable = var, test = "t-test")  
 test\_results <- rbind(test\_results, test\_result)  
}  
  
# Манн-Уитни  
for (var in mann\_whitney\_vars) {  
 data\_for\_test <- cleaned\_data %>%  
 select(Category, !!sym(var))  
   
 test\_result <- broom::tidy(wilcox.test(reformulate("Category", response = var), data = data\_for\_test))  
 test\_result <- test\_result %>% select(p.value) %>% mutate(variable = var, test = "Mann-Whitney U")  
 test\_results <- rbind(test\_results, test\_result)  
}  
  
# Попробуем применить FDR, хотя у нас 2 разных теста, но идеологически эта поправка нам подходит: мы хотим найти интересные результаты, а сет сравнений у нас в целом, один и тот же, хоть и с разными тестами (это все равно немного некорректно, лучше было бы использовать на все сравнения Манн-Уитни, возможно)  
test\_results$p.adj <- p.adjust(test\_results$p.value, method = "fdr")  
  
test\_results <- test\_results %>%   
 arrange(p.adj)  
  
print(test\_results)

## # A tibble: 35 × 4  
## p.value variable test p.adj  
## <dbl> <chr> <chr> <dbl>  
## 1 3.98e-40 Carbohydrate Mann-Whitney U 6.97e-39  
## 2 3.98e-40 Water Mann-Whitney U 6.97e-39  
## 3 8.53e-37 Riboflavin Mann-Whitney U 9.95e-36  
## 4 2.45e-36 Fat.Total\_Lipid Mann-Whitney U 2.14e-35  
## 5 3.34e-36 Fat.Monosaturated\_Fat Mann-Whitney U 2.34e-35  
## 6 1.22e-35 Sugar\_Total Mann-Whitney U 7.14e-35  
## 7 6.86e-34 Fat.Saturated\_Fat Mann-Whitney U 3.43e-33  
## 8 8.68e-33 Fat.Polysaturated\_Fat Mann-Whitney U 3.80e-32  
## 9 3.44e-31 Major\_Minerals.Iron Mann-Whitney U 1.34e-30  
## 10 7.07e-25 Protein Mann-Whitney U 2.48e-24  
## # ℹ 25 more rows

# Видим, что рис и печеньки отличаются практически по всему, что, наверное, достаточно ожидаемо. Визуально по боксплотам это также было так.

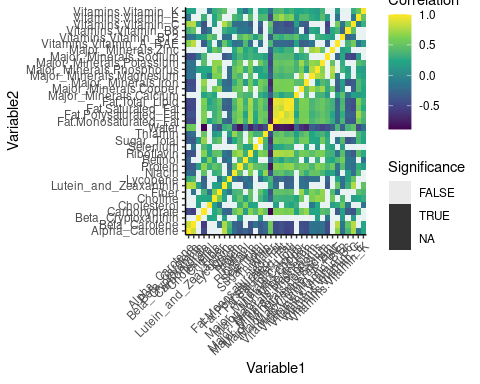
# Далее идут **необязательные** дополнительные задания, которые могут принести вам дополнительные баллы в том числе в случае ошибок в предыдущих

## Корреляционный анализ

1. Создайте корреляционную матрицу с визуализацией и поправкой на множественные сравнения. Объясните, когда лучше использовать корреляционные матрицы и в чём минусы и плюсы корреляционных исследований.

# Буду использовать Спирмана, поскольку он непараметрический, а по распределениям мы уже более менее выяснили  
# В качестве поправки возьму FDR, поскольку мы все еще хотим найти интересные взаимодействия в куче сравнений  
# Когда использовать коррелограммы? Когда нам необходимо выявить потенциально очень похожие признаки, когда необходимо выявить источник мультиколлинеарности, чтобы выявить выбросы, чтобы лучше понять структуру данных  
# Плюсы: они интуитивно понятны и помогают быстро идентифицировать взаимосвязи между переменными  
# Минусы: корреляцию часто путают с причинно-следственной связью, что делать нельзя. Направления у корреляции нет, и взаимосвязь не может никак быть объяснена с помощью лишь коррелограммы.   
  
library(Hmisc)  
library(reshape2)  
  
cor\_results <- rcorr(as.matrix(numeric\_data), type = "spearman")  
cor\_matrix <- cor\_results$r  
p\_matrix <- cor\_results$P  
p\_matrix\_adj <- apply(p\_matrix, c(1, 2), function(x) p.adjust(x, method = "fdr"))  
mask <- p\_matrix\_adj > 0.05  
  
data\_for\_plot <- melt(cor\_matrix)  
names(data\_for\_plot) <- c("Variable1", "Variable2", "Correlation")  
data\_for\_plot$Significance <- p\_matrix\_adj[cbind(match(data\_for\_plot$Variable1, rownames(p\_matrix\_adj)), match(data\_for\_plot$Variable2, colnames(p\_matrix\_adj)))] <= 0.05  
  
ggplot(data\_for\_plot, aes(x = Variable1, y = Variable2, fill = Correlation)) +  
 geom\_tile(aes(alpha = Significance)) +  
 scale\_fill\_viridis\_c() +  
 theme\_classic() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))

## Warning: Using alpha for a discrete variable is not advised.



## Моделирование

1. Постройте регрессионную модель для переменной **Category**. Опишите процесс построения

#Для начала прошкалируем данные и уберем неинформативные переменные и мультиколлинеарность. Категориальные данные неинформативны, мы их удалим. Кроме того, несколько переменных, не отличающиеся по сравнениям, мы тоже удалим (p.adj>0.05)  
# Переменную ответа преобразуем в 0 для категории "Rice" и 1 для "Cookies".   
# Коррелограмма показывает, что жиры и каротины очень хорошо коррелируют между собой, я оставлю Fat.Total\_Lipid и "жирного" кластера и Beta\_Carotene из каротинов. Также вода очень антикоррелирует с кластером жиров, но пока оставлю ее, посчитаю VIF и если он будет высоким, попробую удалить ее также.   
# Все численные переменные сначала лог-шкалирую, уберу найденных аутлаеров, а затем перенесу в область от 0 до 1 для лучшей работы регрессии.   
# Затем посчитаю VIF, отсечкой будет стандартно 10. Буду использовать lm, не лог-регрессию, посколько VIF не зависит от таргетной переменной, а с glm могут быть достаточно странные числа  
library(car)  
library(scales)  
  
insignificant\_vars <- test\_results %>%  
 filter(p.value > 0.05) %>%  
 pull(variable)  
  
model\_df <- data %>% filter(!outliers\_mask)%>%  
 filter(Category %in% c("Rice", "Cookie")) %>%  
 mutate(Category = ifelse(Category == "Rice", 0, 1))%>%  
 select(-all\_of(insignificant\_vars))%>%  
 select(-c(Fat.Saturated\_Fat, Fat.Polysaturated\_Fat, Fat.Monosaturated\_Fat, Alpha\_Carotene, Description,Nutrient\_Data\_Bank\_Number)) %>%  
 mutate(across(where(is.numeric), ~rescale(log1p(.))))  
  
print(head(model\_df))

## # A tibble: 6 × 28  
## Category Beta\_Carotene Carbohydrate Choline Fiber Lutein\_and\_Zeaxanthin  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 0 1 0.429 0.337 0.217  
## 2 1 0 1 0.429 0.337 0.217  
## 3 1 0 0.994 0.685 0 0   
## 4 1 0 0.975 0 0 0   
## 5 1 0 0.975 0 0 0   
## 6 1 0 0.975 0 0 0   
## # ℹ 22 more variables: Lycopene <dbl>, Niacin <dbl>, Protein <dbl>,  
## # Retinol <dbl>, Riboflavin <dbl>, Selenium <dbl>, Sugar\_Total <dbl>,  
## # Thiamin <dbl>, Water <dbl>, Fat.Total\_Lipid <dbl>,  
## # Major\_Minerals.Calcium <dbl>, Major\_Minerals.Copper <dbl>,  
## # Major\_Minerals.Iron <dbl>, Major\_Minerals.Phosphorus <dbl>,  
## # Major\_Minerals.Potassium <dbl>, Major\_Minerals.Sodium <dbl>,  
## # Vitamins.Vitamin\_A\_RAE <dbl>, Vitamins.Vitamin\_B12 <dbl>, …

model\_for\_vif <- lm(Category ~ ., data = model\_df)  
vif\_results <- vif(model\_for\_vif)  
print(sort(vif\_results, decreasing = TRUE))

## Carbohydrate Beta\_Carotene Water   
## 22.950227 21.212660 17.787235   
## Vitamins.Vitamin\_A\_RAE Sugar\_Total Riboflavin   
## 16.278042 12.839048 11.825513   
## Fat.Total\_Lipid Thiamin Niacin   
## 8.768656 7.439261 6.042792   
## Protein Retinol Vitamins.Vitamin\_B6   
## 6.005265 5.925507 5.192080   
## Lutein\_and\_Zeaxanthin Vitamins.Vitamin\_C Major\_Minerals.Potassium   
## 5.132319 5.000045 4.737850   
## Major\_Minerals.Phosphorus Choline Vitamins.Vitamin\_K   
## 4.036458 3.933569 3.767912   
## Major\_Minerals.Copper Major\_Minerals.Iron Vitamins.Vitamin\_E   
## 3.645577 3.596600 3.572148   
## Vitamins.Vitamin\_B12 Fiber Major\_Minerals.Calcium   
## 3.299415 2.749548 2.255477   
## Lycopene Selenium Major\_Minerals.Sodium   
## 2.207319 2.100545 1.692663

# Видим, что Carbohydrate, Beta\_Carotene, Water, Vitamins.Vitamin\_A\_RAE, Sugar\_Total и Riboflavin имеют очень высокий VIF. Я не заметила высокую корреляцию Riboflavin с жирами, а также витамина С c A с каротинами. Попробуем удалить воду и Beta\_Carotene

model\_df <- model\_df%>%  
 select(-c(Water, Beta\_Carotene))  
  
model\_for\_vif <- lm(Category ~ ., data = model\_df)  
vif\_results <- vif(model\_for\_vif)  
print(sort(vif\_results, decreasing = TRUE))

## Carbohydrate Sugar\_Total Riboflavin   
## 16.086140 12.762031 11.811938   
## Fat.Total\_Lipid Thiamin Niacin   
## 8.009160 7.325695 5.554065   
## Protein Vitamins.Vitamin\_B6 Vitamins.Vitamin\_A\_RAE   
## 5.212675 5.175420 4.971612   
## Vitamins.Vitamin\_C Major\_Minerals.Potassium Lutein\_and\_Zeaxanthin   
## 4.928552 4.700019 4.309010   
## Retinol Major\_Minerals.Phosphorus Choline   
## 4.015822 3.967291 3.732781   
## Vitamins.Vitamin\_K Major\_Minerals.Iron Major\_Minerals.Copper   
## 3.725291 3.527235 3.517745   
## Vitamins.Vitamin\_B12 Vitamins.Vitamin\_E Fiber   
## 3.276635 3.240932 2.741046   
## Major\_Minerals.Calcium Lycopene Selenium   
## 2.216055 2.174078 2.083805   
## Major\_Minerals.Sodium   
## 1.588016

# Снизили виф для витамина A, но не для сахаров и рибофлавина. Удалим рибофлавин и Sugar\_Total

model\_df <- model\_df%>%  
 select(-c(Sugar\_Total, Riboflavin))  
  
model\_for\_vif <- lm(Category ~ ., data = model\_df)  
vif\_results <- vif(model\_for\_vif)  
print(sort(vif\_results, decreasing = TRUE))

## Fat.Total\_Lipid Thiamin Carbohydrate   
## 7.857589 6.833448 5.563111   
## Vitamins.Vitamin\_C Vitamins.Vitamin\_A\_RAE Niacin   
## 4.867508 4.836090 4.504796   
## Protein Major\_Minerals.Potassium Lutein\_and\_Zeaxanthin   
## 4.442156 4.400177 4.229658   
## Retinol Major\_Minerals.Phosphorus Vitamins.Vitamin\_K   
## 3.898346 3.872576 3.682435   
## Choline Major\_Minerals.Iron Major\_Minerals.Copper   
## 3.641800 3.453162 3.330789   
## Vitamins.Vitamin\_B6 Vitamins.Vitamin\_E Vitamins.Vitamin\_B12   
## 3.193627 3.161672 2.663687   
## Fiber Major\_Minerals.Calcium Lycopene   
## 2.511672 2.161138 2.145027   
## Selenium Major\_Minerals.Sodium   
## 2.012992 1.542634

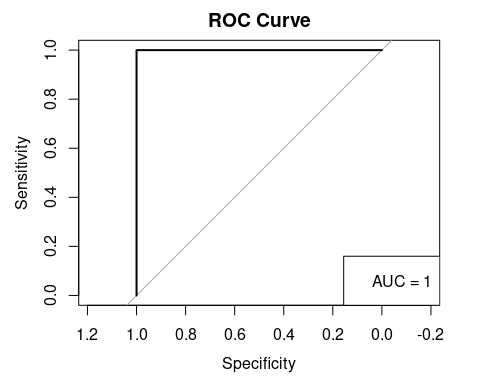
# Отлично, теперь мы готовы к регрессии. Сделаем сплит, построим логрегрессию и посчитаем метрики.

library(caret)  
library(pROC)  
set.seed(42)  
model\_df$Category = as.factor(model\_df$Category)  
split <- createDataPartition(model\_df$Category, p = 0.6, list = FALSE)  
train\_data <- model\_df[split, ]  
test\_data <- model\_df[-split, ]  
  
model <- glm(Category ~ ., data = train\_data, family = binomial(link = "logit"))

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

predictions <- predict(model, test\_data, type = "response")  
  
roc\_curve <- roc(test\_data$Category, predictions)  
auc\_value <- auc(roc\_curve)  
plot(roc\_curve, main = "ROC Curve")  
legend("bottomright", paste("AUC =", round(auc\_value, 2)))



print(roc\_curve)

##   
## Call:  
## roc.default(response = test\_data$Category, predictor = predictions)  
##   
## Data: predictions in 57 controls (test\_data$Category 0) < 40 cases (test\_data$Category 1).  
## Area under the curve: 1

#Идеальная разбивка для риса и печенек. Warning появляется из-за того, что предиктор идеально разделяет группы

control <- trainControl(method = "cv", number = 10)  
cv\_model <- train(Category ~ ., data = model\_df, method = "glm", family = "binomial", trControl = control)

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

print(cv\_model)

## Generalized Linear Model   
##   
## 243 samples  
## 23 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 219, 219, 218, 219, 219, 219, ...   
## Resampling results:  
##   
## Accuracy Kappa  
## 1 1

#То же можно сказать для кросс-валидации