Язык R и среда разработки RStudio.

## **Задание 1. Первый сеанс RStudio.**

1. Запустите RStudio Cloud (<https://rstudio.cloud>). Пройдите регистрацию.
2. В своем рабочем пространстве создайте новый проект PracticeTask1.
3. Создайте простой набор данных (в терминологии R — вектор), состоящий из чисел 1, 2 и 4, и присвойте ему имя х. Выведите на экран фразу «Это мой первый сеанс работы с R».

x <- c(1, 2, 4)  
x

## [1] 1 2 4

print("Это мой первый сеанс работы с R")

## [1] "Это мой первый сеанс работы с R"

1. Выполните команду, которая присваивает q значение (1,2,4,1,2,4,8), используя ранее определенный вектор x.

q <- c(x, x, 8)  
q

## [1] 1 2 4 1 2 4 8

1. Выведите на экран значение 3-го элемента вектора x.

x[3]

## [1] 4

1. Выведите на экран элементы вектора x со 2 по 3

x[2:3]

## [1] 2 4

1. Выведите на экран математическое ожидание и среднеквадратическое отклонение для элементов вектора x.

mean(x)

## [1] 2.333333

sd(x)

## [1] 1.527525

1. Выведите на экран перечень внутренних наборов данных, предназначенных для демонстраций.

data()

1. Вычислите математическое ожидание и среднеквадратическое отклонение для набора данных Nile (этот набор содержит данные о течении Нила).

mean(Nile)

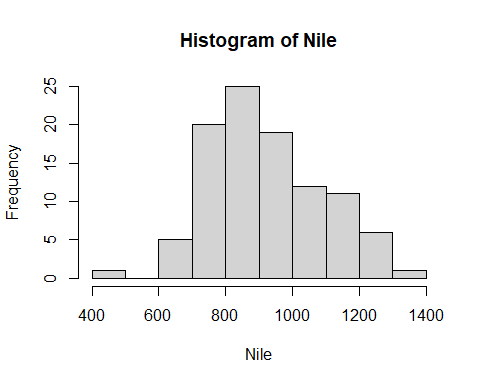
## [1] 919.35

sd(Nile)

## [1] 169.2275

1. Выведите гистограмму этих данных:

hist(Nile)



1. Сохраните проект.
2. Ответьте на вопросы:
3. Как сформировать вектор?  
   Вектор формируется с помощью функции c()
4. Что означает операция с (concatenate)?  
   Операция c (concatenate) объединяет элементы в один вектор
5. Как вывести на экран значение переменной величины?  
   Для вывода значения переменной используется функция print()
6. Нужно ли указывать тип переменной величины?  
   Нет, в R тип переменной определяется автоматически.
7. Как вывести на экран перечень демонстрационных наборов данных?  
   Для вывода перечня демонстрационных наборов данных используется функция data()
8. Как рассчитать среднеквадратическое отклонение?  
   Среднеквадратическое отклонение рассчитывается с помощью функции sd()
9. Как рассчитать математическое ожидание?  
   Математическое ожидание рассчитывается с помощью функции mean()
10. Как построить гистограмму?  
    Гистограмма строится с помощью функции hist()

## **Задание 2 Работа с функциями в языке R.**

1. Запустите RStudio Cloud. Создайте новый проект PracticeTask2 в своем рабочем пространстве.
2. Определите функцию с именем oddcount(), предназначенную для подсчета нечетных чисел в целочисленном векторе:

# Функция подсчитывает количество нечетных целых чисел в x   
oddcount <- function(x) {  
 k <- 0 # в начале вычислений количество нечетных чисел равно 0  
 for (n in x) {  
 if (n %% 2 == 1) {# %% - вычисление остатка  
 k <- k+1   
 }  
 }  
 return(k)  
}

1. Выполните команду Code – Source File…
2. В режиме интерпретатора вычислите количество нечетных чисел в сформированных в предыдущей работе наборов данных x и q с использованием созданной функции:

print(oddcount(x))

## [1] 1

print(oddcount(q))

## [1] 2

1. Попытайтесь вывести в интерпретаторе значения переменных k и n. Что Вы получили?

Error:  
! объект 'k' не найден

1. Определите функцию sum\_2, складывающую два числа, одно из которых имеет значение по умолчанию, равное 5 Протестируйте работу функции. Сохраните файл в своей папке под именем lab2\_2.

sum\_2 <- function(a, b = 5)  
{  
 return(a + b)  
}  
  
print(sum\_2(3))

## [1] 8

print(sum\_2(3, 8))

## [1] 11

1. Напишите функцию fx, определенную на всей числовой прямой Протестируйте работу функции.

fx <- function(x) ifelse(x <= -2, 1-(x+2)^2, ifelse(x <= 2, -x/2, (x-2)^2 + 1))  
  
print(fx(4.5))

## [1] 7.25

print(fx(-4.5))

## [1] -5.25

print(fx(1))

## [1] -0.5

1. Сохраните проект.
2. Ответьте на вопросы:
3. Как определить функцию?  
   Функция определяется с помощью ключевого слова function, например: my\_function <- function(arg1, arg2) {}
4. Как описать тело функции?  
   Тело функции описывается в фигурных скобках {}, где размещается код, который будет выполняться.
5. Как записывается оператор if в языке R?  
   if записывается как: if (условие) { действия }
6. Как записывается оператор for в языке R?  
   for записывается как: for (i in 1:10) { действия }
7. Какие переменные являются глобальными?  
   Глобальными являются переменные, определенные вне функций и доступные во всем скрипте.
8. Какие переменные являются локальными по отношению к функции?  
   Локальными являются переменные, определенные внутри функции и доступные только в ее теле.
9. Как определить значения по умолчанию для аргументов функции?  
   Значения по умолчанию для аргументов функции определяются с помощью знака = в определении функции, например: function(arg1 = значение)
10. В каком порядке должны располагаться аргументы функции, не имеющие значений по умолчанию, и аргументы, имеющие значения по умолчанию?  
    Аргументы без значений по умолчанию должны располагаться перед аргументами с значениями по умолчанию.

## **Задание 3. Структуры данных в R.**

1. Запустите RStudio Cloud. В своем рабочем пространстве создайте новый проект PracticeTask3.
2. Создайте вектор а1 из вектора a1 получите вектор а1, добавив в середину вектора число 161.

a <- c(34, 5, 73, 12)  
a1 <- c(a[1:(length(a)/2)], 161, a[(length(a)/2):length(a)])  
a1

## [1] 34 5 161 5 73 12

1. Создайте фактор month1, содержащий названия весенних месяцев года, при этом должен быть задан полный перечень месяцев и установлен порядок следования месяцев.

month1 <- factor(c("Март", "Апрель", "Май"), levels = c("Март", "Апрель", "Май"), ordered = T)  
month1

## [1] Март Апрель Май   
## Levels: Март < Апрель < Май

1. Сформируйте список emp, содержащий сведения о сотруднике: имя (name – имеет символьные значения); зарплата (salary – вещественное число), принадлежность к профсоюзу (union – логическое значение). Внесите данные об одном сотруднике: Irina, 40800.00, F. Выведите зарплату сотрудника. Измените зарплату сотрудника, задав значение 50000.00.

emp <- list(name = "Irina",  
 salary = 40800.00,  
 union = F)  
emp

## $name  
## [1] "Irina"  
##   
## $salary  
## [1] 40800  
##   
## $union  
## [1] FALSE

1. Создайте фрейм department, описав сотрудников отдела атрибутами: name; salary; union. Введите данные по 2-м сотрудникам. Выведите имена всех сотрудников. Выведите зарплату второго сотрудника.

department <- data.frame(name = "Sergey",  
 salary = 42000,  
 union = T)  
department <- rbind(department, emp)  
  
print(department)

## name salary union  
## 1 Sergey 42000 TRUE  
## 2 Irina 40800 FALSE

print(department$name)

## [1] "Sergey" "Irina"

print(department[2, 2])

## [1] 40800

1. Создайте матрицу m размером 3х2.

m <- matrix(1:6, 3, 2)

1. Сохраните проект.

## **Задание 4. Управление данными в R.**

1. Откройте файл PracticeTask3, созданный как результат выполнение предыдущего задания, сохраните объекты с именами subject1, department, month1 в файл mydata.RData.

save(subject1, department, month1, file = "mydata.RData")

1. Создайте новый файл с именем lab2.R и загрузите в него объекты из файла mydata\_lab1.RData, выведите объекты subject1, department, month1.

rm(list = ls())  
load("mydata.RData")

1. Сохраните текущий сеанс, воспользовавшись командой save.image().

save.image()

1. Используя функцию rm() удалите объекты month1 и subject1, проверьте список оставшихся объектов, используя функцию ls(). Удалите все объекты. Используя файл .RData, восстановите объекты.

rm(month1, subject1)  
ls()

## [1] "department"

rm(list = ls())  
ls()

## character(0)

load("mydata.RData")

1. Откройте файл R, созданный в ходе выполнения предыдущего задания. Запишите в файл pt\_data.csv фрейм данных pt\_data, определив параметр row.names=FALSE. Вернитесь в файл R текущей работы. Прочитайте содержимое файла pt\_data.csv.

pt\_data <- read.csv("pt\_data.csv", stringsAsFactors = F)  
  
write.csv(pt\_data, file = "pt\_data.csv", row.names = F)

1. Загрузите файл usedcars.csv на сервер, если Вы используете среду RStudio Cloud, используя кнопку.
2. Загрузите данные их файла usedcars.csv во фрейм usedcars.

usedcars <- read.csv("usedcars.csv")

1. Отобразите структуру usedcars, используя команду str().

str(usedcars)

## 'data.frame': 150 obs. of 6 variables:  
## $ year : int 2011 2011 2011 2011 2012 2010 2011 2010 2011 2010 ...  
## $ model : chr "SEL" "SEL" "SEL" "SEL" ...  
## $ price : int 21992 20995 19995 17809 17500 17495 17000 16995 16995 16995 ...  
## $ mileage : int 7413 10926 7351 11613 8367 25125 27393 21026 32655 36116 ...  
## $ color : chr "Yellow" "Gray" "Silver" "Gray" ...  
## $ transmission: chr "AUTO" "AUTO" "AUTO" "AUTO" ...

1. Используя функцию summary() запросите статистику по всем числовым переменным usedcars.

summary(usedcars)

## year model price mileage   
## Min. :2000 Length:150 Min. : 3800 Min. : 4867   
## 1st Qu.:2008 Class :character 1st Qu.:10995 1st Qu.: 27200   
## Median :2009 Mode :character Median :13592 Median : 36385   
## Mean :2009 Mean :12962 Mean : 44261   
## 3rd Qu.:2010 3rd Qu.:14904 3rd Qu.: 55125   
## Max. :2012 Max. :21992 Max. :151479   
## color transmission   
## Length:150 Length:150   
## Class :character Class :character   
## Mode :character Mode :character   
##   
##   
##

1. Используя функцию mean() посчитайте средние значения для всех числовых переменных usedcars.

sapply(usedcars[, c(3, 4)], mean, rm.na = T)

## price mileage   
## 12961.93 44260.65

1. Используя функцию median() посчитайте медианы для всех числовых переменных usedcars.

sapply(usedcars[, c(3, 4)], median, rm.na = T)

## price mileage   
## 13591.5 36385.0

1. Изучите пятичисловую сводку для переменных price и mileage.

summary(usedcars$price)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3800 10995 13592 12962 14904 21992

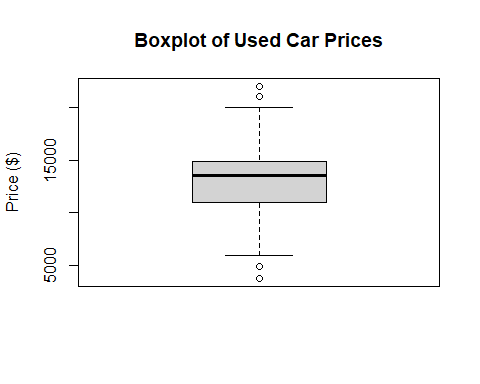
summary(usedcars$mileage)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 4867 27200 36385 44261 55125 151479

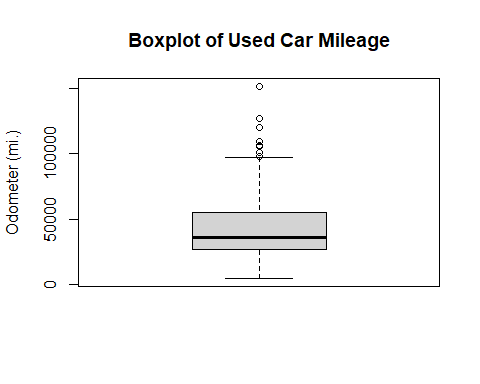
**Задание 5. Визуализация данных в R.**

1. Рассмотрим диаграмму размаха для данных о цене и пробеге подержанных автомобилей. Чтобы получить диаграмму размаха для переменной, воспользуемся функцией boxplot().

**boxplot**(usedcars**$**price, main = "Boxplot of Used Car Prices", ylab = "Price ($)")

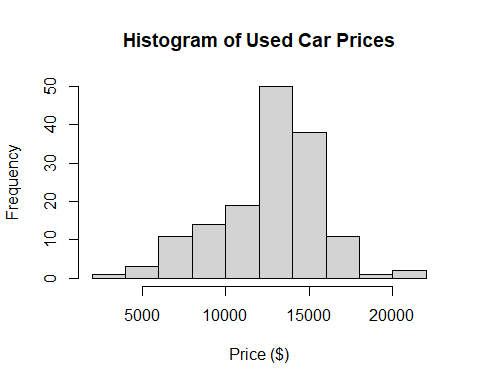


**boxplot**(usedcars**$**mileage, main = "Boxplot of Used Car Mileage", ylab = "Odometer (mi.)")

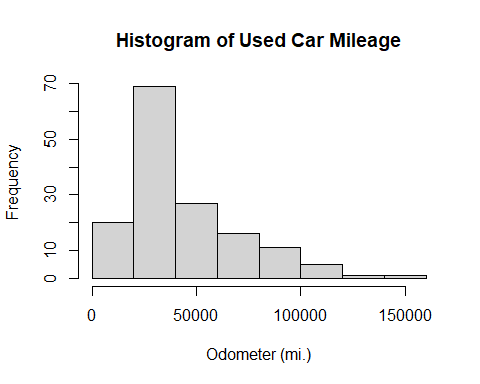


1. Для того чтобы построить гистограмму для данных о цене и пробеге подержанных автомобилей, можно воспользоваться функцией hist().

**hist**(usedcars**$**price, main = "Histogram of Used Car Prices", xlab = "Price ($)")



**hist**(usedcars**$**mileage, main = "Histogram of Used Car Mileage", xlab = "Odometer (mi.)")



1. В R для вычисления дисперсии и стандартного отклонения используются функции var() и sd().

**print**(**var**(usedcars**$**price))

## [1] 9749892

**print**(**sd**(usedcars**$**price))

## [1] 3122.482

**print**(**var**(usedcars**$**mileage))

## [1] 728033954

**print**(**sd**(usedcars**$**mileage))

## [1] 26982.1

1. Чтобы построить таблицу частотности для данных о подержанном автомобиле, можно воспользоваться функцией table().

**print**(**table**(usedcars**$**year))

##   
## 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012   
## 3 1 1 1 3 2 6 11 14 42 49 16 1

**print**(**table**(usedcars**$**model))

##   
## SE SEL SES   
## 78 23 49

**print**(**table**(usedcars**$**color))

##   
## Black Blue Gold Gray Green Red Silver White Yellow   
## 35 17 1 16 5 25 32 16 3

1. R также позволяет вычислить пропорции таблицы напрямую, с помощью команды prop.table() для таблицы, созданной функцией table().

model\_table <- **table**(usedcars**$**model)  
**prop.table**(model\_table)

##   
## SE SEL SES   
## 0.5200000 0.1533333 0.3266667

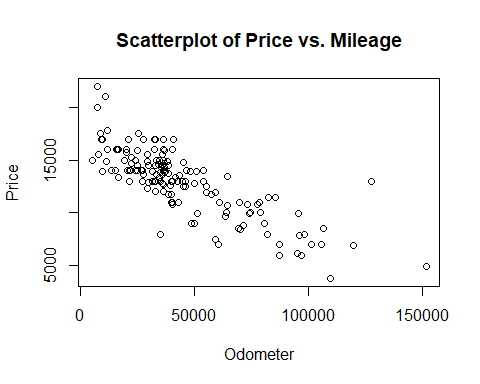
1. Результаты prop.table() можно объединять с другими функциями R для представления вывода в другой форме. Предположим, что мы хотим отобразить результаты в процентах с одним десятичным знаком.

color\_table <- **table**(usedcars**$**color)  
color\_pct <- **prop.table**(color\_table) **\*** 100  
**round**(color\_pct, digits = 1)

##   
## Black Blue Gold Gray Green Red Silver White Yellow   
## 23.3 11.3 0.7 10.7 3.3 16.7 21.3 10.7 2.0

1. Чтобы ответить на вопрос о соотношении цены и пробега, построим диаграмму разброса. Для этого воспользуемся функцией plot() с параметрами main, xlab и ylab.

**plot**(x = usedcars**$**mileage, y = usedcars**$**price,  
 main = "Scatterplot of Price vs. Mileage",   
 xlab = "Odometer", ylab = "Price")



1. Прежде чем приступить к анализу, упростим наш проект, сократив количество уровней в переменной color. Исследуя результат выполнения функции table() для нашей новой переменной, можно увидеть, что примерно две трети автомобилей имеют консервативные цвета, а одна треть — нет.

usedcars**$**conservative <- usedcars**$**color **%in%** **c**("Black", "Gray", "Silver", "White")  
**table**(usedcars**$**conservative)

##   
## FALSE TRUE   
## 51 99

1. Теперь посмотрим на кросс-таблицу, чтобы увидеть, как количество автомобилей консервативных цветов зависит от модели.

**library**("gmodels")  
**CrossTable**(x = usedcars**$**model, y = usedcars**$**conservative)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | Chi-square contribution |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 150   
##   
##   
## | usedcars$conservative   
## usedcars$model | FALSE | TRUE | Row Total |   
## ---------------|-----------|-----------|-----------|  
## SE | 27 | 51 | 78 |   
## | 0.009 | 0.004 | |   
## | 0.346 | 0.654 | 0.520 |   
## | 0.529 | 0.515 | |   
## | 0.180 | 0.340 | |   
## ---------------|-----------|-----------|-----------|  
## SEL | 7 | 16 | 23 |   
## | 0.086 | 0.044 | |   
## | 0.304 | 0.696 | 0.153 |   
## | 0.137 | 0.162 | |   
## | 0.047 | 0.107 | |   
## ---------------|-----------|-----------|-----------|  
## SES | 17 | 32 | 49 |   
## | 0.007 | 0.004 | |   
## | 0.347 | 0.653 | 0.327 |   
## | 0.333 | 0.323 | |   
## | 0.113 | 0.213 | |   
## ---------------|-----------|-----------|-----------|  
## Column Total | 51 | 99 | 150 |   
## | 0.340 | 0.660 | |   
## ---------------|-----------|-----------|-----------|  
##   
##

Отчет о выполнении индивидуального задания

### **Разработка предиктивной модели для определения вероятности заболевания диабетом**

Целью является прогнозировать вероятность наличия диабета и распределение по ней наблюдаемых объектов по двум классам: 0 - отсутствие и 1 - наличие соответствующего заболевания.

Это может быть полезно медицинским работникам при выявлении пациентов, которые могут быть подвержены риску развития диабета, и при разработке индивидуальных планов лечения.

### **Описание используемой модели**

Для предсказания использована одно из семейств моделей GLM. Существует несколько семейств моделей GLM в зависимости от состава переменной отклика. Для работы выбрана биномиальное семейство. Выбор основан на том, что Биномиальное семейство используется для бинарных переменных отклика (т.е. двух категорий) и предполагает биномиальное распределение. Иными словами, в работе используется логистическая регрессия, которая будет определять зависимую бинарную переменную наличия заболевания от независимых переменных. Независимыми переменными будут служить все остальные переменные в train\_data в качестве предикторов. Это может включать различные факторы, такие как возраст, индекс массы тела, уровень глюкозы и т.д.

## **Описание набора данных**

Набор данных взят с сайта kaggle. Ссылка для скачивания: <https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/data>

Набор данных для прогнозирования диабета представляет собой набор медицинских и демографических данных о пациентах, а также их диабетическом статусе (положительном или отрицательном). Эти данные включают такие характеристики, как возраст, пол, индекс массы тела (ИМТ), наличие гипертонии, наличие заболевания сердца, история курения, уровень HbA1c и глюкозы в крови. Этот набор данных используется для построения модели машинного обучения и прогнозирования диабета у пациентов на основе их истории болезни и демографической информации.

# **Подробный разбор кода**

# **Подключение необходимых пакетов**

# libraries  
library("caret")  
library("ggplot2")  
library("ROCR")  
set.seed(42)

## **Чтение и предобработка данных**

1. Считывание данных из файла.

Загрузим данные из файла CSV с именем “diabetes\_prediction\_dataset.csv” и сохраним их в переменной my\_df. Также посмотрим структуру данных.

my\_df <- read.csv("diabetes\_prediction\_dataset.csv")  
str(my\_df)

## 'data.frame': 100000 obs. of 9 variables:  
## $ gender : chr "Female" "Female" "Male" "Female" ...  
## $ age : num 80 54 28 36 76 20 44 79 42 32 ...  
## $ hypertension : int 0 0 0 0 1 0 0 0 0 0 ...  
## $ heart\_disease : int 1 0 0 0 1 0 0 0 0 0 ...  
## $ smoking\_history : chr "never" "No Info" "never" "current" ...  
## $ bmi : num 25.2 27.3 27.3 23.4 20.1 ...  
## $ HbA1c\_level : num 6.6 6.6 5.7 5 4.8 6.6 6.5 5.7 4.8 5 ...  
## $ blood\_glucose\_level: int 140 80 158 155 155 85 200 85 145 100 ...  
## $ diabetes : int 0 0 0 0 0 0 1 0 0 0 ...

1. Преобразование столбцов в факторы.

Некоторые столбцы измерены не в количественных шкалах, их стоит перевести в фактор, что позволит R правильно обрабатывать категориальные переменные.

my\_df[, c(1, 3, 4, 5, 9)] <- lapply(my\_df[, c(1, 3, 4, 5, 9)], as.factor)  
str(my\_df)

## 'data.frame': 100000 obs. of 9 variables:  
## $ gender : Factor w/ 3 levels "Female","Male",..: 1 1 2 1 2 1 1 1 2 1 ...  
## $ age : num 80 54 28 36 76 20 44 79 42 32 ...  
## $ hypertension : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...  
## $ heart\_disease : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 1 1 ...  
## $ smoking\_history : Factor w/ 6 levels "current","ever",..: 4 5 4 1 1 4 4 5 4 4 ...  
## $ bmi : num 25.2 27.3 27.3 23.4 20.1 ...  
## $ HbA1c\_level : num 6.6 6.6 5.7 5 4.8 6.6 6.5 5.7 4.8 5 ...  
## $ blood\_glucose\_level: int 140 80 158 155 155 85 200 85 145 100 ...  
## $ diabetes : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...

1. Удаление дубликатов и проверка на наличие пропущенных значений.

# удалим повторяющиеся записи  
my\_df <- unique(my\_df)  
# проверим на пропущенные значения  
any(is.na(my\_df))

## [1] FALSE

Пропущенных значений нет, теперь можно посмотреть краткую сводку о данных и перейти к обработке данных.

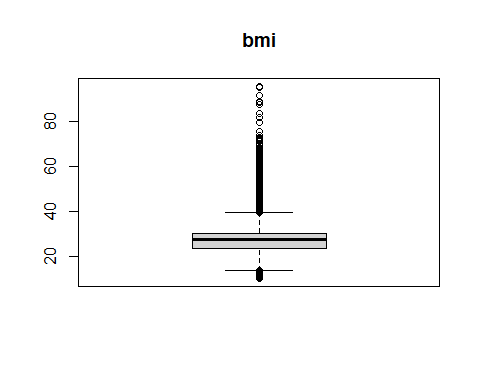
summary(my\_df)

## gender age hypertension heart\_disease smoking\_history   
## Female:56161 Min. : 0.08 0:88685 0:92223 current : 9197   
## Male :39967 1st Qu.:24.00 1: 7461 1: 3923 ever : 3998   
## Other : 18 Median :43.00 former : 9299   
## Mean :41.79 never :34398   
## 3rd Qu.:59.00 No Info :32887   
## Max. :80.00 not current: 6367   
## bmi HbA1c\_level blood\_glucose\_level diabetes   
## Min. :10.01 Min. :3.500 Min. : 80.0 0:87664   
## 1st Qu.:23.40 1st Qu.:4.800 1st Qu.:100.0 1: 8482   
## Median :27.32 Median :5.800 Median :140.0   
## Mean :27.32 Mean :5.533 Mean :138.2   
## 3rd Qu.:29.86 3rd Qu.:6.200 3rd Qu.:159.0   
## Max. :95.69 Max. :9.000 Max. :300.0

1. Удаление выбросов методом межквартильного размаха.

Есть подозрения на наличие выбросов в переменной bmi. Это можно проверить с помощью графика boxplot (ящик с усами)

boxplot(my\_df$bmi, main = "bmi")



Как видно, существует множество значений, выходящих за границу крайних квартилей. Удалим наблюдения с ними для лучшей работы моделей

# удаляем выбросы  
q\_1 <- quantile(my\_df$bmi, 0.25)  
q\_3 <- quantile(my\_df$bmi, 0.75)  
iqr <- q\_3 - q\_1  
lower\_bound <- q\_1 - 1.5 \* iqr  
upper\_bound <- q\_3 + 1.5 \* iqr  
my\_df <- my\_df[my\_df$bmi >= lower\_bound & my\_df$bmi <= upper\_bound, ]

1. Стандартизация данных.

my\_df[, -c(1, 3, 4, 5, 9)] <- sapply(my\_df[, -c(1, 3, 4, 5, 9)], scale)  
head(my\_df)

## gender age hypertension heart\_disease smoking\_history bmi  
## 1 Female 1.6912937 0 1 never -0.2332038  
## 2 Female 0.5442608 0 0 No Info 0.1737743  
## 3 Male -0.6027722 0 0 never 0.1737743  
## 4 Female -0.2498389 0 0 current -0.5656648  
## 5 Male 1.5148271 1 1 current -1.1981050  
## 6 Female -0.9557054 0 0 never 0.1737743  
## HbA1c\_level blood\_glucose\_level diabetes  
## 1 1.0133688 0.05707638 0  
## 2 1.0133688 -1.43018178 0  
## 3 0.1687918 0.50325382 0  
## 4 -0.4881014 0.42889092 0  
## 5 -0.6757852 0.42889092 0  
## 6 1.0133688 -1.30624360 0

1. Разделение данных на обучающую и тестовую выборки.

В этом шаге данные разделяются на обучающую (70%) и тестовую (30%) выборки. Функция createDataPartition используется для случайного выбора индексов для обучающей выборки, а оставшиеся данные используются для тестовой выборки.

# split data  
train\_index <- createDataPartition(my\_df$diabetes, p = 0.7, list = FALSE)  
train\_data <- my\_df[train\_index, ]  
test\_data <- my\_df[-train\_index, ]

## **Обучение модели и предсказание модели.**

# train model on split data  
model <- glm(diabetes ~ ., data = train\_data, family = "binomial")  
  
predictions <- predict(model, newdata = test\_data[, c(-9)], type = "response")

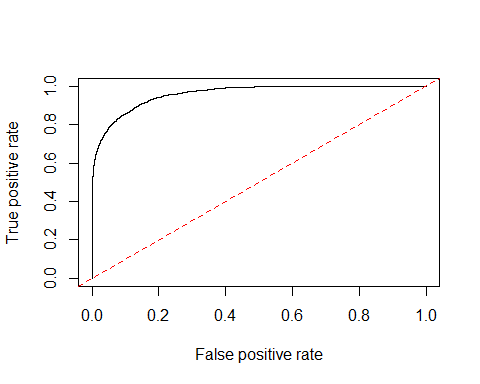
Выполнив код, мы получили вероятности того, относится ли наблюдаемый объект к классу 1 (имеется диабет). Установив пороговое значение, мы сможем определять сам класс.

## **Определение порогового значения**

Теперь остро встает вопрос определения порогового значения для отнесения объекта к классу. Для этого в своих воспользуемся визуализацией метрик модели

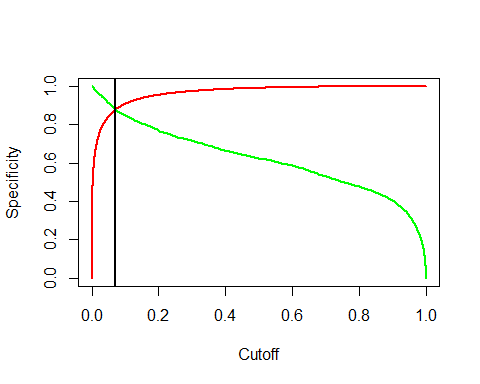
Для начала отобразим ROC Curve, чтобы убедиться в хорошей точности модели и отсутствии перекошенности.

#####  
pred\_fit <- prediction(as.numeric(predictions), as.numeric(test\_data$diabetes))  
perf\_fit <- performance(pred\_fit, "tpr", "fpr")  
plot(perf\_fit)  
# Add a diagonal reference line  
abline(a = 0, b = 1, col = "red", lty = 2)



Теперь отобразим две метрики Specificity и Sensitivity. Их пересечение позволит определить оптимальную границу.

#####  
perf3 <- performance(pred\_fit, x.measure = "cutoff", measure = "spec")  
perf4 <- performance(pred\_fit, x.measure = "cutoff", measure = "sens")  
  
plot(perf3, col = "red", lwd = 2)  
plot(add = T, perf4, col = "green", lwd = 2)  
# Отобразим линию пересечения на координате 0.07  
abline(v = 0.07, lwd = 2)



Так мы определили пороговое значение, и теперь можем поменять предсказанные вероятности на определенный класс.

threshold <- 0.07

# **Определим классы и вектор с метками, которые говорят о правильности определения класса.**

predicted\_classes <- ifelse(predictions > threshold, 1, 0)  
  
correct <- ifelse(predicted\_classes == test\_data$diabetes, 1, 0)

# **Используем различные метрики точности модели.**

predicted\_classes <- factor(predicted\_classes)  
test\_data$diabetes <- factor(test\_data$diabetes)  
  
confusion\_matrix <- confusionMatrix(predicted\_classes, test\_data$diabetes)  
print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 21980 252  
## 1 3093 1912  
##   
## Accuracy : 0.8772   
## 95% CI : (0.8732, 0.8811)  
## No Information Rate : 0.9205   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4752   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8766   
## Specificity : 0.8835   
## Pos Pred Value : 0.9887   
## Neg Pred Value : 0.3820   
## Prevalence : 0.9205   
## Detection Rate : 0.8070   
## Detection Prevalence : 0.8162   
## Balanced Accuracy : 0.8801   
##   
## 'Positive' Class : 0   
##

#####  
  
accuracy <- mean(correct)  
f1\_score <- confusion\_matrix$byClass["F1"]

Итак, мы получили Confusion matrix (матрицу ошибок) с различными метриками и две основные: accuracy = 0.8771891 и f1 score = 0.9292887.

# **Сохранение модели.**

Сохраним модель вместе с пороговым значением в model.RData.

save(model, threshold, file = "model.RData")

# **Заключение**

Таким образом, …