#####################Программирование на языке R. Занятие 1#####################

######################## Занятие 3. Типы объектов в R#######################

#### Язык R поддерживает 6 базовых (atomic) типов данных: integer, complex,

#### numeric, character, logical, raw (сырые байты).

#### Переменные и константы, в которых хранятся эти данные, являются объектами

#### определённого класса.

#### Вектор - основной класс объектов. Для создания векторов используется функция с()

#### (от англ. «combine»). Вектор представляет из себя объект, в котором хранятся данные ОДНОГО типа

## Вектор из трёх чисел

v1 <- c(1, 5, 10)

## Вектор из последовательности чисел от одного до семи

v2 <- c(1:7)

## Вектор логических значений

v3 <- c(TRUE, FALSE)

## Для создания пустого вектора можного воспользоваться следующим кодом

vecemp <- c()

## Для создания вектора заданной длины и заданного типа (например, для накопления данных)

## можно использовать функцию vector().

vec <- vector(mode = "character", length = 5)

## Как вы уже знаете из занятия 2, чтобы обратиться к отдельному элементу вектора, вы можете

## использовать его порядковый номер в квадратных скобках. Также можно вызвать несколько

## элементов вектора, указав их номера (с помощью другого вектора).

v1[2]

v2[3:5]

v1[c(1,3)]

## Вы можете изменять сразу несколько элементов вектора, используя диапазоны.

## Однако помните, что элементы вектора должны относиться к одному типу данных.

## Выполните строки ниже и поймите разницу между результатами, полученными для vec и vecemp

vec

vec[1:3] <- v2[3:5]

vec

vecemp

vecemp[1:3] <- v2[3:5]

vecemp

## Если вы попробуете совместить вместе данные character и logical, последние будут

## конвертированы в character.

vec[4:5] <- v3

vec

is.logical(vec)

is.numeric(vec)

is.character(vec)

## Если вы попробуете совместить вместе данные numeric и logical, последние будут

## конвертированы в numeric: TRUE превратится в 1, FALSE – в 0.

vecemp[4:5] <- v3

vecemp

is.logical(vecemp)

is.numeric(vecemp)

is.character(vecemp)

#### Создание вектора, содержащего последовательный ряд чисел с заданным шагом, производится при помощи

#### функции seq().

## Разберитесь, что делают аргументы данной функции

seq(from = 5, to = 7, by = 0.2)

#### Создание вектора из повторяющихся элементов

rep("Cow", 33)

## Проверить длину вектора можно, используя функцию length()

length(rep("Cow", 33))

cow <- rep("Cow", 33)

length(cow)

#### Случайное извлечение элементов вектора осуществляется с помощью функции sample.

#### Аргумент size указывает, сколько элементов нужно извлечь, аргумент replace = TRUE

#### позволяет извлекать один и тот же элемент вектора несколько раз.

sample(x = LETTERS, size = 12, replace = FALSE)

sample(x = LETTERS, size = 12, replace = TRUE)

## Для каждого из элементов вектора LETTERS можно задать вероятность, с которой он будет

## извлекаться функцией sample(). В этом случае задаётся аргумент prob, который является

## вектором такой же длины, как и LETTERS. Первый элемент probs задаёт вероятность для первого

## элемента LETTERS, второй – для второго и т.д. Все элементы вектора prob ОБЯЗАТЕЛЬНО должны

## давать в сумме 1 (т.е. вероятность в 100 %). Код ниже позволяет создать вектор из шести

## первых букв латинского алфавита.

sample(x = LETTERS, size = 12, replace = TRUE, prob = c(0.5, 0.15, 0.15, 0.1, 0.05, 0.05, rep(0, (length(LETTERS) - 6))))

#### УПРАЖНЕНИЕ 1

#### Создайте вектор, содержащий 3 повтора слова "удав", 5 повторов слова "мартышка" и 38 повторов слова

#### "попугай".

#### ПОДСКАЗКА: Используйте функции rep(), c(), а также функции для редактирования строк из прошлого задания.

#### КОНЕЦ УПРАЖНЕНИЯ

#### Следующий класс объектов - Матрица.

#### Матрица - данные представлены в виде прямоугольника. Стороны прямоугольника могут иметь

#### неравную длину. Аргумент byrow определяет порядок заполнения - по строкам или столбцам.

## Сравните результаты выплнения кода ниже, чтобы осознать разницу

matrix(data = c(1:6), nrow = 2, ncol = 3, byrow = TRUE)

matrix(data = c(1:6), nrow = 2, ncol = 3, byrow = FALSE)

## Создадим переменную в которую запишем матрицу

m <- matrix(data = c(1:6), nrow = 2, ncol = 3, byrow = FALSE)

## Чтобы увидеть внутреннюю структуру объекта, используйте функцию str()

str(m)

#### УПРАЖНЕНИЕ 2

#### Запишите в качестве матрицы вектор с удавами, мартышками и попугаями. Обратите внимание

#### на предупреждения (если они у вас появятся). Как нужно записывать данные в матрицу, чтобы их не было?

#### КОНЕЦ УПРАЖНЕНИЯ

#### Следующий класс объектов - Таблица.

#### Таблица (data frame). Обычно её строки соответствуют наблюдениям, колонки - параметрам

#### наблюдения. Каждая колонка – вектор. В таблице могут быть колонки РАЗНОГО типа.

#### Давайте посмотрим, как это выглядит на примере данных, собранных в 19 веке британским учёным

#### Фрэнсисом Гальтоном, который изучал взаимосвязь между ростом родителей и их детей.

install.packages("HistData")

library(HistData)

df <- GaltonFamilies

? GaltonFamilies

df

str(df)

### Что такое фактор? Это объект, который используется для разбивки данных на категории.

c("Mother", "Father", "Child")

as.factor(c("Mother", "Father", "Child"))

### Для создания фактора с заданными уровнями и их порядком используется функция factor()

factor(c("Mother", "Father", "Child"), levels = c("Mother", "Father", "Child"))

## Создание таблицы

df2 <- data.frame(first = c(1:10), this\_is\_second = c(rep(c(TRUE, FALSE), 5)), and\_third = rep(c(NA, NA, 2, 4, 5), 2))

?NA

str(df2)

# Чтобы вывести таблицу в отдельном окне, используйте функцию View()

View(df2)

##УПРАЖНЕНИЕ 3.

## 1) Создадим новую таблицу на основе таблицы GaltonFamilies. Давайте добавим

## в неё информацию о весе родителей, а также о том, имеют ли они

## наследственные заболевания. Масса тела родителей в данном случае будет случайной

## величиной из диапазона 45 и 90 кг, а наличие заболеваний - случайным фактором

## Healthy или Diseased. Создайте вектора нужной длины при помощи функции sample().

## Запишите в переменную dfr новую таблицу, в которую будут добавлены колонки Parents\_Weight с весом

## и колонка Health - для указания статуса здоровья.

#### КОНЕЦ УПРАЖНЕНИЯ

### Список - позволяет комбинировать объекты различных типов. Создаётся при помощи функции list(),

### аналогично data.frame()

## УПРАЖНЕНИЕ 4.

## Создайте лист при помощи функции list() по аналогии с функцией data.frame(). Включите в его состав

## вектор с животными, матрицу и таблицу из прошлых заданий. Изучите струкруру этого объекта

## при помощи str()

#### КОНЕЦ УПРАЖНЕНИЯ

## Атрибуты - метаданные, описывающие структуру объекта. Используйте names(), чтобы добавлить имена

## к вектору

names(v1)

names(v1) <- c("FirstName", "LastName")

names(v1) #обратите внимание на NA

##УПРАЖНЕНИЕ 5

## Изучите colnames() и rownames() для таблицы dfr, измените имена колонок с помощью этих функций

## КОНЕЦ УПРАЖНЕНИЯ