**Практическая работа №3**

# Создание и обработка фреймов

Фрейм (или таблица данных) представляет собой двухмерную структуру, в которой строки имеют одинаковую длину, а значения разных столбцов могут принадлежать различным типам – быть числами допустимых форматов, тестовыми или логическими величинами.

Одним из способов создания фрейма является присвоение источника, в качестве которого может быть текстовый документ, файлы табличных процессоров, баз данных, других специальных программ, либо функции, используемые в таких случаях, – read.table(), read.xlsx(), read.csv(), а также приведены примеры их использования.

Другим способом создания фреймов является применение функции data.frame (), обеспечивающей формирование таблицы данных из указанных векторов и матриц, записываемых в требуемом порядке следования:
data.frame(x1,x2,…,xn, row.names = NULL, check.rows = FALSE,
check.names = TRUE, fix.empty.names = TRUE,
stringsAsFactors = default.stringsAsFactors())

Рассмотрим назначение основных параметров функции. Параметры xl, x2, xn указывают на имена соответствующих векторов и матриц создаваемого фрейма. Данные параметры могут записываться как в форме позиционных, так и ключевых параметров (в виде Xl=xl, X2=x2, Xn=xn, где XI, X2, …,Xn – имена столбцов в создаваемом фрейме).

Параметр row.names представляет собой строковый или целочисленный вектор, определяющий имена строк в создаваемом фрейме. По умолчанию (при значении NULL) строки нумеруются числами.

Параметр check.rows при значении TRUE определяет необходимость проверки строи создаваемого фрейма на согласованность длины и имен.

Параметр check.names при значении TRUE предписывает проверять имена переменных (столбцов) во фрейме на синтаксическую допустимость и отсутствие дублирования. При необходимости имена корректируются.

Для обращения к элементам фрейма следует указывать либо имя фрейма, за которым в квадратных скобках приводить индексы элемента

фрейма как плоской структуры данных, либо записывать составное имя, связывая имя фрейма и имя столбца (вектора, матрицы) знаком $.

Примеры представлены на рисунках 20-22.

```
# Создание фреймов
?data.frame
fio<-c("Антонов А.в.","Борисов С.Р.","Волошин И.В.","Федоров А.Е.")
fio
country<-c("Россия","Белоруссия","Индия","Китай")
country
capital<-c("Москва","Минск","Дели","Пекин")
capital
popul<-c(146390000,214600000,1339180000,1401296000)
popul
fr<-data.frame(FIO=fio,Coutry=country,
               Capital=capital,
               Population=popul,
               check.rows = T,
               check.names = T,
               fix.empty.names = T)
fr
```

Рисунок 1 Пример создания фрейма

```
# Операции над фреймами
fr[,1]
fr[,2]
fr[3,]
fr[4,4]
fr[1,]
fr[2,]
fr$Population[1]
sum_popul<-fr$Population[3]+fr$Population[4]
sum_popul
sum_population<-sum(fr$Population)
sum_population
```

Рисунок 2 Операции над элементами фрейма

```
# Создание фрейма с пустыми значениями вектора
fam<-c(NA,NA,NA,NA)
fam
fr1<-data.frame(Family=fam,Coutry=country,
                Capital=capital,
                Population=popul,
                check.rows = F)

fr1
fr1$Family[2]<-"Петрова О.Л."# добавление элемента в пустой список
fr1
```

Рисунок 3 Создание и обработка частично пустого фрейма

При обработке числовых данных фреймов полезной оказывается функция tranform():

transform(x, tagl=valuel, tag2=value2,…, tag n=value n)

Параметр x – объект, подлежащий трансформации. Параметры tagl, tag2, tag n определяют новые расчетные переменные в фрейме.

В примере, приведенном на рисунке 23, показано, как с помощью данной функции, не создавая новых объектов, можно, например, посчитать суммарные и средние баллы студентов за сессию.

```
fio<-c("Антонов А.в.","Борисов С.Р.","Волошин И.В.","Федоров А.Е.")
fio
fr<-data.frame(fio,
               matematik=c(4,2,3,4),
               computer=c(4,3,3,5),
               history=c(4,3,4,5),
               economic=c(4,3,3,5))

fr
?transform
dat<-transform(fr,sumx=matematik+computer+history+economic,
               meanx=(matematik+computer+history+economic)/4)
dat
```

Рисунок 4 Пример создания новых переменных в фрейме

## Работа с датафреймами с использованием библиотеки dplyr

Библиотека dplyr – библиотека для удобной работы с датафреймами. С её помощью можно более быстро получать описание таблицы, сохранять полученные результаты и группировать данные по определенному признаку.

Установим библиотеку и обратимся к ней:

install.packages("dplyr")

library(dplyr)

Воспользуемся данными о качестве воздуха в Нью-Йорке, измеренными в период с май по сентябрь далекого 1973 года:

library(datasets)

Посмотрим на данные:

head(airquality)

Смысл понятен, но информацию о данных в более удобоваримом виде можно получить, используя функцию dplur glimplse(). Команда транспонирует данные (поменяет местами строки и столбцы) и укажет количество наблюдений и переменных.

glimpse(airquality)

### Основные функции dplyr и оператор %>%

Некоторые функции, встроенные в библиотеку, похожи на обычные функций, которые мы использовали на прошлом занятии. Например, функция select(), которая позволяет выбрать интересующие нас столбцы в датафрейме. Выберем только столбцы со скоростью ветра и днём.

select(airquality, Wind, Day)

Выберем все столбцы, имена которых содержат латинскую «o»

select(airquality, contains("o"))

Посмотрим на столбцы, имена которых начинаются с «Oz»

select(airquality, starts_with("Oz"))

Также с помощью select() можем исключить некоторые столбцы, которые нас не интересуют, поставив перед вектором столбцов минус (так же, как и раньше!):

select(airquality, -c(Wind, Day))

Столбцы можно выбирать по названиям, если столбцы идут подряд:

head(select(airquality, Ozone:Temp))

Если хотим отобрать интересующие нас наблюдения, нам потребуется другая функция – filter(). Например, выберем все наблюдения, когда температура поднималась выше 92 градусов по фаренгейту.

filter(airquality, Temp > 92)

Условия можно комбинировать как душе угодно. Вот хочется мне узнать, когда в июне значения измеренной солнечной радиации превышали 300 Лэнгли.

filter(airquality, Month == 6 & Solar.R > 300)

Рассмотрим еще одну команду  mutate(), которая позволяет добавлять новые переменные (столбцы). Добавим столбец с температурой в цельсиях, который назовем TempInC:

mutate(airquality, TempInC = (Temp – 32) * 5 / 9)

Функция summarise() позволяет получить быстрое представление о всех данных в одном значении, например, посчитать среднее арифметическое. В комбинации с командой группировки данных group_by() можно легко находить интересующие значения для группированных данных. Простой пример: посчитаем среднюю температуру в каждом из месяцев.

summarise(group_by(airquality, Month), ean_monthly_temp=mean(Temp, na.rm = TRUE))

na.rm = TRUE просто не учитывает все отсутствующие данные при расчете среднего.

В рамках summarise() помимо нахождения среднего арифметического, можно использовать и другие команды, возвращающие скалярные значения, например min(), max(), sum(), sd(), median().

Казалось бы, зачем использовать библиотеку dplyr, если результаты пока несильно отличаются от того, что мы делали на прошлом занятии без всяких библиотек. На самом деле, смысл использовать её есть.

В библиотеке dplyr есть особый оператор Pipe %>%, который позволяет выполнять операции пошагово. Оператор %>% позволяет совершать последовательные операции без сохранения промежуточных результатов. Допустим, мы хотим увидеть, какая максимальная температура в градусах цельсия была зарегистрирована в летние месяцы.

airquality %>%

filter(Month == c(6,7,8)) %>% #оставляем только летние месяцы

mutate(TempInC = (Temp – 32) * 5 / 9)%>% # считаем температуру в градусах цельсия

group_by(Month) %>% # группируем по месяцам

summarise(max_temp_C = max(TempInC, na.rm = TRUE)) # находим максимальное значение температуры для каждого из летних месяцев

В библиотеке dplyr есть несколько других интересных и полезных функций. Например, arrange() – функция, которая сортирует таблицу в соответствии со значениями переменной (или переменных), расположенных по возрастанию (если переменная текстовая, то по алфавиту). Отсортируем, например, таблицу по показателю Temp сначала.

airquality %>% arrange(Temp) %>% head

А теперь на последние:

airquality %>% arrange(Temp) %>% tail

**Задача.** Выбрать строки в таблице, для которых значения Ozon не пустые (не NA в переменной Ozon), выбрать температуру больше 77, отсортировать строки по дню и посмотреть на итоговую таблицу.

Решение.

```
25  airquality %>%
26    filter(!is.na(Ozone), Temp > 77) %>%
27    arrange(Day) %>%
28      View
```

Сначала мы выберем те строки в базе, для которых значения Ozon не пустые (is.na() и помним про отрицание – восклицательный знак), выберем

температуру больше 77. Затем с помощью arrange() отсортируем строки по значениям Day. И, наконец, посмотрим на полученный датафрейм через View.

## Экспорт данных из R в другие форматы

Обработанные в среде R данные могут быть сохранены во внешних файлах других форматов.

Для вывода данных в текстовые файлы с расширением +.txt используется функция write.tableQ:

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
        eol = "\n", na = "NA", dec = ".", row.names = TRUE,
        col.names = TRUE, qmethod = c("escape", "double"),
        fileEncoding = "")
```

Параметр x определяет объект данных R (вектор, матрицу, массив данных, фрейм), содержимое которого экспортируется во внешний текстовый файл.

Параметр file представляет собой строку, определяющую имя создаваемого при экспорте файла – в текущей папке R либо иной папке, полный путь к которой указан.

Значение остальных параметров во многом аналогично параметрам функции read.table(). Более точную информацию можно найти в справке.

Варианты использования функции write.table() приведены на рисунке.

```r
# Экспорт данных в текстовый файл с помощью функции write.table()
?write.table
fio<-c("Антонов А.в.","Борисов С.Р.","Волошин И.В.","Федоров А.Е.")
fio
country<-c("Россия","Белоруссия","Индия","Китай")
country
capital<-c("Москва","Минск","Дели","Пекин")
capital
popul<-c(146390000,214600000,1339180000,1401296000)
popul
fr<-data.frame(FIO=fio,Coutry=country,
               Capital=capital,
               Population=popul,
               check.rows = T,
               check.names = T,
               fix.empty.names = T)
fr
# экспорт с номерами строк
write.table(fr, file = "my_fr.txt", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
# экспорт без номеров строк
write.table(fr, file = "my_fr_1.txt", row.names = F)
```

Рисунок 5 Пример экспорта данных в текстовый файл

Для вывода данных из среды R во внешний файл MS Excel с расширением *.csv используется функция write.csv():

write.csv(x, file = "")

Параметр x определяет объект экспортируемых данных. Параметр file, как и для функции write.table(), задает в виде строки имя создаваемого файла (в текущей папке или в указанном месте).

Варианты использования функции wrlte.csvQ приведены на рисунке 25.

```r
# Экспорт данных в файл Excel с помощью функции write.csv()
?write.csv()
fio<-c("Антонов А.в.","Борисов С.Р.","Волошин И.В.","Федоров А.Е.")
fio
country<-c("Россия","Белоруссия","Индия","Китай")
country
capital<-c("Москва","Минск","Дели","Пекин")
capital
popul<-c(146390000,214600000,1339180000,1401296000)
popul
fr<-data.frame(FIO=fio,Coutry=country,
               Capital=capital,
               Population=popul,
               check.rows = T,
               check.names = T,
               fix.empty.names = T)
fr
write.csv(fr,file="my_fr_2.csv")
```

Рисунок 6 Экспорт данных в файл MS Excel

Для экспорта таблиц данных (фреймов) из среды R во внешний файл MS Excel с расширением *.xlsx используется функция write.xlsx():

wrfte.xlsx (x, file, sheetName="Sheet1",

col.names=TRUE, row.names=TRUE, append= FALSE, showNA=TRUE)

Позиционный параметр x указывает на имя фрейма, который предполагается вывести во внешний файл.

Параметр file представляет собой строку, которая содержит имя создаваемого файла. Если приведено неполное имя без указания пути к файлу, создаваемый файл сохраняется в текущей рабочей папке.

Параметр sheetName определяет имя листа книги Excel в создаваемом файле.

Параметр col,names при значении TRUE предписывает выводить в файл имснэ столбцов данных; значение FALSE – исключает вывод.

Параметр row.names при значении TRUE выводит в файл имена строи в виде чисел, заданных в текстовом формате. Предпочтительней выглядит значение параметра FALSE.

Параметр append при значении FALSE указывает на замену созданного листа, если функция выполняется повторно, Значение TRUE, если указано новое имя листа, обеспечивает добавление в книгу нового листа, В противном случае формируется сообщение об ошибке.

Параметр showNA позволяет при значении TRUE выводить в файл пустые ячейки.

Прежде чем приводить пример использования функции write.xfsxj) для экспорта полученных результатов из среды R а среду табличного процессора MS Excel, рассмотрим ряд функций, которые находят применение перед экспортом с целью представления результатов в более читаемом варианте,

Функция formatC() служит для форматирования числовых данных и превращения результата в строки:

formatC(x, digits = NULL, width = NULL, format = NULL, flag = "",
mode = NULL, big.mark = "", big.interval = 3L,
small.mark = "", smail.interval = 5L,
decimal.mark = getOption("OutDec"),
preserve.width = "individual", zero.print = NULL,
dropOtrailing = FALSE)

Основные параметры функции:

x – число ила вектор вещественных чисел, которые предполагается отформатировать:

digits – количество знаков в дробной части числа;

format – форма представления числа: "f" – с фиксированной запятой, "e" – с плавающей запятой,

Значения других параметров можно узнать, вызвав справку:

?formatC либо help(formatC),

Функция round() используется для округления числовых данных;

round(x, digits = 0)

Первый параметр функции x представляет собой исходное число.

Параметр digits задает количество знаков после запятой.

Пример

Экспорт данных – фрейма в файл MS EXCEL с расширением .xlsx.

Из таблицы данных зарплата.csv создать с помощью функции data.frame() фрейм newtable, содержащий названия регионов ЦФО и средние заработные платы работников указанных регионов за 2008 и 2017 годы. Полученные числовые значения отформатировать. Содержимое фрейма экспортировать в файл MS Excel с расширением *.xlsx на указанный лист.

Добавить в скрипт команду на вывод значения фрейма на экран. Сравнить результаты вывода фрейма на экран и в файл MS Excel.

Решение

```
# Экспорт данных в файл Excel с помощью функции write.csv()
library("xlsx")
dat<-read.table("clipboard")#,h=F,dec = ".",sep = "")
dat
region<-dat[,1]
region
x2008<-dat[,2]
x2008
x2017<-dat[,11]
rost<-x2017/x2008
rost
x2017<-round(x2017.1)
rost<-round(rost,2)
rost
p<-getwd()
newtable<-data.frame(region,x2008,x2017,rost)
newtable
file<-paste(p,"newfile.xlsx",sep="/")
file
library(writexl)
write.xlsx(newtable,
           file,
           sheetName=sheet_user,
           row.names=F,
           colnames=T,
           append=F)
```

Рисунок 7 Протокол скрипта

## Списки (lists)

Список представляет собой «вектор векторов» в терминах R. Для тех, кто знаком с программированием, может показаться, что списки похожи на массивы. Это так, но списки, в отличие от массивов, могут содержать элементы разных типов. Например, в списке в R может быть сохранен вектор имен студентов (текстовый, тип *character*) и вектор их оценок (целочисленный, тип *integer*).

Для тех, кто знаком с Python: списки в Python и списки в R похожи. Списки в R – это вложенные списки в Python.

Списком называется многомерная упорядоченная структура разнотипных данных.

Для создания списков используется функция list():

list(x1,x2,…,xn)

Параметры x1,x2,…,xn могут быть векторами, матрицами, массивами данных, фреймами и другими списками.

При создании списков обычно предварительно создаются его компоненты, впоследствии объединяемые с помощью функции list() в списки.

Пример списка с числовыми значениями:

```
L <- list(c(1, 2, 3, 4), c(5, 6, 7, 8))
L
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
```

А вот пример списка с элементами разных типов:

```
grades <- list(c("Ann", "Sam", "Tom"), c(8, 7, 5))
grades
## [[1]]
## [1] "Ann" "Sam" "Tom"
##
## [[2]]
## [1] 8 7 5
```

Так как в списках может храниться большое число разных векторов, для удобства им можно давать названия. Список grades можно было записать и так:

```
grades <- list(names = c("Ann", "Sam", "Tom"), marks = c(8, 7, 5))
```

И тогда отдельные вектора из списка можно было бы вызывать удобным образом:

```
grades$names # имена
## [1] "Ann" "Sam" "Tom"
grades$marks # оценки
## [1] 8 7 5
```

И если мы бы запросили у R структуру этого списка, мы бы увидели названия векторов, которые в него входят.

```
str(grades)
## List of 2
##  $ names: chr [1:3] "Ann" "Sam" "Tom"
##  $ marks: num [1:3] 8 7 5
```

Можно подумать: зачем нужно знать про списки, если на практике мы обычно будем сталкиваться с другими объектами – датафреймами (таблицами)? На самом деле, со списками мы тоже будем встречаться. Многие статистические функции выдают результат в виде списков. Когда результаты выводятся на экран, это не всегда заметно, но если мы захотим заглянуть внутрь, то увидим, что та же регрессионная выдача представляет собой объект, похожий на список, из которого можно выбрать вектор коэффициентов (coefficients), вектор остатков (residuals), предсказанных значений (fitted.values) и так далее.

А как обращаться к элементам списка, если вектора в нем никак не названы?

Для обращения к элементам списка необходимо использовать двойные квадратные скобки:

```
L
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
L[[1]] # первый элемент списка, вектор (1, 2, 3, 4)
## [1] 1 2 3 4
```

Если нужно обратиться к «элементу элемента» списка (например, к числу 8 в этом примере), нужно сначала указать номер вектора, в котором находится элемент, а затем номер самого элемента в этом векторе.

```
L[[2]][4] # 8 – 4-ый элемент 2-ого вектора в списке
## [1] 8
```

Можно заметить, что список похож на матрицу: для того, чтобы обратиться к элементу, нужно указать «строку» (вектор) и «столбец» (положение в векторе).

Для того чтобы добавить элемент в список, нужно четко понимать положение элемента в этом списке: будет ли это элементом самого списка или «элементом элемента»:

```
L
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
L[[3]] <- c(8, 9) # добавили в список третий вектор
L
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
##
## [[3]]
## [1] 8 9
L[[3]][3] <- 0 # добавили третий элемент третьего вектора в списке
L
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
##
## [[3]]
## [1] 8 9 0
```

Аналогичным образом можно изменять элементы списка:

```
L[[1]][1] = 99 # заменим 1 элемент 1 вектора в массиве на 99
L
## [[1]]
## [1] 99  2  3  4
##
## [[2]]
## [1] 5 6 7 8
##
## [[3]]
## [1] 8 9 0
```

Если в списке всего один элемент, при необходимости его можно быстро превратить в обычный вектор с помощью unlist():

```
small_L <- list(c("a", "b", "c"))
small_L
## [[1]]
## [1] "a" "b" "c"
small_vec <- unlist(small_L)
small_vec
## [1] "a" "b" "c"
```

То же можно делать и со списками из нескольких векторов, тогда все склеится в один длинный вектор:

```
L
## [[1]]
## [1] 99  2  3  4
##
## [[2]]
## [1] 5 6 7 8
##
## [[3]]
## [1] 8 9 0
unlist(L)
##  [1] 99  2  3  4  5  6  7  8  8  9  0
```

## Преобразование типов и структур данных

В ходе обработки данных иногда приходится выполнять преобразования данных. Возможности преобразования представлены на рисунке.

```
# Преобразования типов данных
a<-1:10
a
# Преобразование вектора a в строковый вектор
a_str<-as.character(a)
a_str
# Конвертация строкового вектора в целочисленный вектор
d<-c("1","2","3")
d
d_int<-as.integer(d)
d_int
# Конвертация целочисленного вектора в номинальную
# шкалу (factor)со значениями 0 и 1
b<-c(1,1,0,0,1,1,0)
b
b_factor<-as.factor(b)
b_factor
```

Рисунок 8 Выполнение операций преобразования данных

## Самостоятельная работа №3

### Часть 1

1. Создайте из векторов из задачи 3 список (list) и назовите его info.

names <- c("Jane", "Michael", "Mary", "George")
ages <- c(8, 6, 28, 45)
gender <- c(0, 1, 0, 1)

- Выведите на экран имя Michael (обращаясь к элементам списка, конечно).
- Выведите на экран вектор gender.
- Назовите векторы в списке name, age, gender. Выведите на экран элементы вектора name.
- Добавьте в список вектор drinks, в котором сохранены значения: juice, tea, rum, coffee.
- Добавьте в список данные по еще одному человеку: John, 2 года, мужской пол, любит молоко.

2. В R есть функция strsplit(), которая позволяет разбивать строки (текстовые переменные) на части по определенным символам.

Пусть у нас есть строка s:

s <- "a,b,c,d"

Мы хотим получить из нее вектор из 6 букв. Применям функцию:

let <- strsplit(s, ",")

Получили почти то, что хотели. Почему почти? Потому что получили не вектор, а список!

class(let)
## [1] "list"

Превратим в вектор:

unlist(let)
## [1] "a" "b" "c" "d"

Теперь все в порядке, получили вектор из четырех элементов.

Дана строка index:

index <- "0,72;0,38;0,99;0,81;0,15;0,22;0,16;0,4;0,24"

Получите из этой строки числовой вектор I.

**Часть 2**

1. Поставьте библиотеку randomNames. Обратитесь к ней через library().
2. Создайте вектор из 100 испанских имен:

   set.seed(1234) # чтобы у всех получались одинаковые результаты

   names <- randomNames(100, which.names = "first", ethnicity = 4)
3. Будем считать, что эти 100 имен – имена опрошенных респондентов.

   Создайте вектор со значениями возраста респондентов:

   ages <- sample(16:75, 100, replace = TRUE) # replace = TRUE – с повторяющимися значениями

   А также вектор polit – политические взгляды респондентов:

   views <- c("right", "left", "moderate", "indifferent")

   polit <- sample(views, 100, replace = TRUE)

   Создайте из полученных трёх векторов датафрейм.
4. Создайте столбец id с номерами респондентов.
5. Определите, сколько среди респондентов людей в возрасте от 25 до 30 лет (включительно). Определите, какую долю респондентов в нашей сымпровизированной выборке составляют люди в возрасте от 25 до 30

лет. Выразите эту долю в процентах, округлите ее до 1 знака после запятой.

6. Создайте «факторный» вектор политических взглядов polit_views. Сколько у полученного фактора уровней? Добавьте в датафрейм столбец polit_views.

**Часть 3**

1. Загрузите файл Firms.csv. Почитать про базу можно https://www.rdocumentation.org/packages/car/versions/2.1-5/topics/Ornstein. Посмотрите на таблицу.
2. Сколько в датафрейме наблюдений? Сколько переменных? Какие это переменные?
3. Сколько в датафрейме полностью заполненных строк (наблюдений)? Выведите (если такие есть) наблюдения, содержащие пропущенные значения на экран.
4. Отфильтруйте наблюдения в таблице согласно следующим критериям:
    a) фирмы с активами от 10000 до 20000 (включительно);
    b) фирмы, число управляющих позиций, совместных с другими фирмами, которых не превышает 30;
    c) фирмы транспортного сектора (TRN) под руководством управляющих из Канады (CAN);
5. Создайте переменную «натуральный логарифм активов» (log_assets) и добавьте её в датафрейм.
6. Постройте график, который может проиллюстрировать, какие паттерны пропущенных наблюдений можно зафиксировать в таблице.
7. Удалите пропущенные значения из базы данных.
8. Сохраните измененную базу данных в формате Stata (файл “Firms.dta”)

**Часть 3**

**Импорт, предобработка и преобразование данных измерений**

Выполнение задания в рамках изучения языка R должно сформировать целостное представление о потенциальных и явных возможностях обработки данных с использованием скриптового языка. В данной работе вам предстоит загрузить данные из открытого информационного источника, провести исследование загруженных данных, произвести преобразование данных к виду плоской таблицы, удобной для обработки, произвести некоторые вычисления по фрейму данных и его экспорт для возможной обработки в дальнейшем.

Необходимо загрузить данные из источника Интернет, поэтому убедитесь, что присутствует подключение к сети при загрузке данных.

Для выполнения задания понадобятся такие пакеты для обработки данных как “dplyr”, “tidyr”, “stringr”. Эти пакеты позволяют удобно преобразовывать данные к виду, который вам необходим. Для их установки выполните следующие действия:

1. Убедитесь, что папка пользователя не содержит кириллических символов или знаков препинания
2. Если содержит, убедитесь в том, что папка, в которую установлен R: “R/R-…/library” доступна для записи программными приложениями. Для этого необходимо открыть свойства папки и разрешить доступ на запись к данной папке.
3. Установите данные пакеты с помощью команды:

   install.packages(c(“dplyr”, “readr”, “stringr”))

**Задание 1.**

1. Загрузить при помощи функции read.csv() данные о положительных тестах на Covid19 из открытого репозитория github: https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv

2. Исследовать данные, произвести некоторые первоначальные действия по просмотру свойств фрейма данных: установить размерность таблицы, имена столбцов, типы данных столбцов (по возможности, определить структуру информации, с которой столкнулись)

3. Произвести слияние колонок, связанных со страной и регионом – источником информации, для унификации метаданных статистики (вам может помочь пакет tidyr).

4. Отдельно произвести создание фрейма данных с колонками:

5.

Таблица 1

Поля фрейма данных

| Страна/Регион | Широта | Долгота | Сумма заболевших | Среднее число заболевших | Стандартное отклонение числа заболевших |
|---|---|---|---|---|---|

6. С помощью библиотек с(“dplyr”, “tidyr”, “stringr”) и стандартных возможностей языка R произвести создание нового итогового фрейма данных со следующей структурой таблицы:

Таблица 2

Формат итогового фрейма данных

| Дата<br><date> | Страна1<br><int> | Страна2<br><int> | … | СтранаN<br><int> |
|---|---|---|---|---|
| YYYY-MM-DD | $X_{11}$ | $X_{12}$ | … | $X_{1N}$ |
| YYYY-MM-DD | $X_{21}$ | $X_{22}$ | … | $X_{2N}$ |
| … | … | … | … | … |
| YYYY-MM-DD | $X_{M1}$ | $X_{M2}$ | … | $X_{MN}$ |

В столбце Дата (date) должны храниться значения в стандартном для R типе данных date. Формат YYYY-MM-DD явно указывает на положение года, месяца, дня и разделителя между ними. X – это явное число подтвержденных положительных тестов на Covid19 на данный день в данной стране. Под названиями колонок “Страна1”, ... “СтранаN” подразумеваются те имена стран и регионов, которые вами предварительно созданы в пункте 3.

Для решения данной задачи вам возможно понадобятся функции: gsub(), ISOdate(), as.Date(), stringr::str_extract().

7. Итоговый фрейм данных, полученный в пункте 5, необходимо экспортировать в формат txt, csv, xlsx в поддиректорию вашего проекта с названием “data_output”.

**Важно.** Директория “data_output” должна создаваться во время выполнения скрипта, и не должна пересоздаваться во время повторного выполнения скрипта. Файлы с таблицей данных каждый раз должны перезаписываться в данной директории.

Контрольные вопросы

1. Каковы особенности фреймов
2. Приведите примеры создания фреймов
3. Приведите примеры экспорта данных из среды R во внешние файлы
4. Что собой представляют списки. Каким образом может быть создан список
5. Приведите примеры преобразования данных