ПРАКТИЧЕСКАЯ РАБОТА № 2. Создание, заполнение, загрузка и обработка хранилища данных

Цель работы – научиться применять методы создания, заполнения, загрузки и обработки хранилищ данных с использованием стандартных средств языка программирования R и библиотеки dplyr.

Задачи:

1. Изучить реализацию стандартных средств языка R и библиотеки dplyr для создания, заполнения и загрузки таблиц данных.
2. Реализовать методы обработки данных с использованием стандартных средств языка программирования R и библиотеки dplyr.

Теоретические сведения

Структура данных data.frame

Таблица данных (data.frame) – это стандартный способ представления данных на языке программирования R, который представляет собой прямоугольную двумерную таблицу. В строках содержатся наблюдения, а в столбцах – исследуемые переменные. Таблицы данных наследует как свойства матрицы, так и списка переменных разных типов. Формально, data.frame — это именованный список (list), все строки (row.names) и столбцы которого имеют одинаковую длину.

Создание таблицы данных data.frame производится одноименной функцией:

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE, fix.empty.names = TRUE,
           stringsAsFactors = FALSE)
```

Аргументы функции `data.frame()`:

– ... – означает, что допустимы дополнительные аргументы, не указанные в данном списке. Дополнительные аргументы позволяют создавать новые переменные, при этом названия аргументов будут соответствовать названиям переменных;

– `row.names` – вектор имён строк, представляет собой либо вектор с именами строк итоговой таблицы, либо число – номер столбца исходной таблицы с названиями строк; либо имя столбца считываемой таблицы, где приведены названия строк; если этот параметр не задан, то строки в итоговой таблице будут пронумерованы;

– `check.rows` – логический аргумент, при значении `TRUE` имена наблюдений будут проверены на синтаксическую правильность и отсутствие дублирования;

– `check.names` – логический аргумент, при значении `TRUE` имена переменных будут проверены на синтаксическую правильность и отсутствие дублирования;

– `fix.empty.names` – логический аргумент, при значении `TRUE` пустые имена переменных будут заданы автоматически;

– `stringsAsFactors` – логический аргумент, при значении `TRUE` строковые переменные будут автоматически преобразованы в факторные.

Пример 2.1. Рассмотрим пример создания нового набора данных

```
> df <- data.frame(x = 1:4, y = LETTERS[1:4], z = c(T, F))
> df
  x y     z
1 1 A  TRUE
2 2 B FALSE
3 3 C  TRUE
4 4 D FALSE
```

Отобразим структуру данных

```
> str(df)
'data.frame':  4 obs. of  3 variables:
 $ x: int  1 2 3 4
 $ y: chr  "A" "B" "C" "D"
 $ z: logi  TRUE FALSE TRUE FALSE
```

В данном случае создается набор данных из 3 переменных (числовой, строковый и логический). Логическая переменная `z` отличается тем, что она получена из вектора длины 2, но в таблице данных, содержатся 4 значения, то есть наблюдения в переменной `z` циклически дублируется.

Ряд небольших наборов данных содержатся в стандартной библиотеке datasets, которые используются в качестве примеров обработки и отображения данных с использованием различных функций языка R.

Доступ к количеству строк и столбцов осуществляется через функции nrow, ncol, dim:

```
> nrow(df)
[1] 4
> nrow(df)
[1] 4
> ncol(df)
[1] 3
> dim(df)
[1] 4 3
```

Таблицы данных `data.frame` наследуют некоторые особенности от списка, а некоторые от матрицы – это касается и правил индексирования.

Рассмотрим индексацию в матричном стиле

- положительная – для выбора индексов;
- отрицательная – для исключения индексов;
- логическая – для выбора по условию;
- по именам – для обращения к переменным по имени;

Пример 2.2. Рассмотрим примеры индексации

Отобразим только 3 и 4 наблюдения, 1 переменная – исключается.

```
> df[3:4, -1]
  y     z
3 C  TRUE
4 D FALSE
```

Отобразим только 1 переменную.

```
> df[, 1]
[1] 1 2 3 4
```

Обратимся к нечётным наблюдениям, переменные – `z` и `x`.

```
> df[c(F, T), c("z", "x")]
      z x
2 FALSE 2
4 FALSE 4
```

Другой способ обращения к переменным осуществляется через значок $.

```
> df$z
[1]  TRUE FALSE  TRUE FALSE
```

Можно обратится к переменной в стиле, который соответствует обращению к элементу списка (list):

```
> df[["x"]]
[1] 1 2 3 4
```

В таблицах данных удобно производить фильтрацию наблюдений по условию.

Пример 2.3. Рассмотрим примеры фильтрации

Отобразим наблюдения, которые удовлетворяют условию `x > 2`

```
> df[df$x > 2, ]
  x y     z
3 3 C  TRUE
4 4 D FALSE
```

Фильтрация реализована функцией `subset`. В данном случае нет необходимости в дублировании названия и не нужно заключать переменные в кавычки. Такую же фильтрацию, что и выше, можно получить с использованием `subset`

```
> subset(df, x > 2)
  x y     z
3 3 C  TRUE
4 4 D FALSE
```

Третий аргумент `select` позволяет выбрать указанные переменные:

```
> subset(df, x > 2, select = c(x, y))
  x y
3 3 C
4 4 D
```

Для объединения двух таблиц данных используются функции `rbind` – для объединения по строкам и `cbind` – для объединения по столбцам.

Пример 2.5. Рассмотрим пример объединения таблиц по строкам. Создадим

```
> df1 <- data.frame(x = 5:8, y = LETTERS[5:8], z = c(T, F))
> rbind(df, df1)
  x y     z
1 1 A  TRUE
2 2 B FALSE
3 3 C  TRUE
4 4 D FALSE
5 5 E  TRUE
6 6 F FALSE
7 7 G  TRUE
8 8 H FALSE
```

и по столбцам

```
> cbind(df, df1)
  x y     z x y     z
1 1 A  TRUE 5 E  TRUE
2 2 B FALSE 6 F FALSE
3 3 C  TRUE 7 G  TRUE
4 4 D FALSE 8 H FALSE
```

Для объединения двух таблиц данных по ключу используется функция `merge`.

Пример 2.6. Рассмотрим пример объединения таблиц по ключу. Создадим новый набор данных

```
> df2 <- data.frame(x = c(3, 2, 6, 1), s = c(100, 1000, 300, 500))
> df2
  x    s
1 3  100
2 2 1000
3 6  300
4 1  500
```

и объединим их по ключу (переменной `x`):

```
> merge(df, df2, by = "x")
  x y     z    s
1 1 A  TRUE  500
```

```
2 2 B FALSE 1000
3 3 C  TRUE  100
```

Пример 2.7. Рассмотрим пример со встроенным набором данных `mtcars`, содержащемся в стандартной библиотеке datasets

```
> df <- mtcars
> typeof(df)
[1] "list"
> colnames(df)
 [1] "mpg" "cyl""disp" "hp"   "drat" "wt" "qsec" "vs" "am" "gear" "carb"
> rownames(df)
 [1]"Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" "Hornet Sportabout"
 [6] "Valiant" "Duster 360"  "Merc 240D" "Merc 230" "Merc 280"
[11] "Merc 280C" "Merc 450SE" "Merc 450SL" "Merc 450SLC" "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial" "Fiat 128" "Honda Civic" "Toyota
Corolla"
[21] "Toyota Corona" "Dodge Challenger" "AMC Javelin" "Camaro Z28" "Pontiac Fir
ebird"
[26] "Fiat X1-9" "Porsche 914-2" "Lotus Europa" "Ford Pantera L" "Ferrari Dino"
[31] "Maserati Bora" "Volvo 142E"
```

Все наборы данных, содержащиеся в стандартной библиотеке datasets относятся представлены в виде структуры data.frames.

Набор данных `mtcars` содержит 11 различные технико-экономические показатели (переменные) 32 автомобилей (наблюдения).

Функция `rownames()`выводит в виде списка наименование наблюдений. Другая функция `colnames(df)` – информацию о переменных. Выведем с помощью функции `str(df)` информацию описание 11 переменных в наборе данных `mtcars`.

```
> str(df)
'data.frame':  32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Далее представлено описание этих переменных:

[, 1] `mpg` Расстояние (в милях) преодолеваемое автомобилем на 1 галлоне (американский) топлива
[, 2] `cyl` Количество цилиндров
[, 3] `disp` Измещение (кубический дюйм)
[, 4] `hp` Полная мощность в лошадиных силах
[, 5] `drat` Передаточное отношение задней оси
[, 6] `wt` Вес (1000 фунтов)

[, 7] `qsec` Время на 1/4 мили
[, 8] `vs` Тип двигателя (0 = V-образный, 1 = прямой)
[, 9] `am` Тип трансмиссии (0 = автоматическая, 1 = ручная)
[,10] `gear` Количество передач переднего хода
[,11] `carb` Количество карбюраторов

Как видно из описания, все переменные – непрерывные (num, т.е. числовые). Часть переменных, например, количество цилиндров, передач переднего хода и карбюраторов, можно рассматривать в дискретной **порядковой** шкале, а тип двигателя и трансмиссии – в дискретной **номинальной** шкале. В данном случае порядковая и номинальная шкалы были уточнены как «дискретные» для уточнения того, что они отличаются от непрерывной. В дальнейшем, не будем уточнять, что порядковая и непрерывная шкалы – являются дискретными.

Сформируем из `mtcars` новый набор данных `mtcars2` с учетом наличия дискретных переменных.

```
> mtcars2 <- within(mtcars, {
+   vs <- factor(vs, labels = c("V", "S"))
+   am <- factor(am, labels = c("automatic", "manual"))
+   cyl  <- ordered(cyl)
+   gear <- ordered(gear)
+   carb <- ordered(carb)
+ })
> str(mtcars2)
'data.frame':  32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : Ord.factor w/ 3 levels "4"<"6"<"8": 2 2 1 2 3 2 3 1 1 2 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : Factor w/ 2 levels "V","S": 1 1 2 2 1 2 1 2 2 2 ...
 $ am  : Factor w/ 2 levels "automatic","manual": 2 2 2 1 1 1 1 1 1 1 ...
 $ gear: Ord.factor w/ 3 levels "3"<"4"<"5": 2 2 2 1 1 1 1 2 2 2 ...
 $ carb: Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<..: 4 4 1 1 2 1 4 2 2 4 ...
```

Новый набор данных `mtcars2` будет обрабатываться с учетом их шкалы. Выведем результаты предварительной обработки данных с помощью команды `summary`.

```
> summary(mtcars2)
      mpg        cyl         disp             hp             drat      
 Min.   :10.40   4:11   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
 1st Qu.:15.43   6: 7   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
 Median :19.20   8:14   Median :196.3   Median :123.0   Median :3.695  
 Mean   :20.09          Mean   :230.7   Mean   :146.7   Mean   :3.597  
```

```
 3rd Qu.:22.80          3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
 Max.   :33.90          Max.   :472.0   Max.   :335.0   Max.   :4.930
       wt             qsec       vs              am     gear   carb
 Min.   :1.513   Min.   :14.50   V:18   automatic:19   3:15   1: 7
 1st Qu.:2.581   1st Qu.:16.89   S:14   manual   :13   4:12   2:10
 Median :3.325   Median :17.71                         5: 5   3: 3
 Mean   :3.217   Mean   :17.85                                4:10
 3rd Qu.:3.610   3rd Qu.:18.90                                6: 1
 Max.   :5.424   Max.   :22.90                                8: 1
```

Библиотека трансформации и обработки данных dplr

Рассмотрим базовые инструменты предоставляемые библиотекой `dplyr` и их применение к структурам данных типа data.frame.

Для исследования основных операций манипулирования данными в `dplyr` воспользуемся набором данных `nycflights13`. Он содержит данные 336776 рейсов из Нью Йорка за 2013 год, и предоставлен, как написано в `?nycflights13`, [бюро по перемещениям США](#).

```
library(nycflights13)
dim(flights)
#> [1] 336776     16
head(flights)
#> Source: local data frame [6 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Библиотека `dplyr` может работать с `data.frame` непосредственно, но при работе с большими объёмами данных стоит конвертировать их в `tbl_df`, обёртку вокруг `data.frame` которая не станет выводить кучу данных на экран.

В библиотеке `dplyr` содержатся функции для всех основных операций манипулирования данным:

- `filter()`
- `slice()`
- `arrange()`
- `select()`
- `rename()`
- `distinct()`
- `mutate()`
- `transmute()`
- `summarise()`

- `sample_n()`
- `sample_frac()`

Операция `filter()` позволяет выбрать подмножество строк из `data.frame`. Первый аргумент – имя набора данных, второй и последующие – условия фильтра в контексте этого набора данных.

Например, мы можем выбрать все вылет 1-го января следующим образом:

```
filter(flights, month == 1, day == 1)
#> Source: local data frame [842 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Операция `filter()` работает аналогично `subset()` за тем исключением, что можно передать любое количество условий для фильтра, которые будут объединены вместе через `&` (логическое И), а также использовать любые логически связки:

```
filter(flights, month == 1 | month == 2)
```

Для выборки строк по позициям используется `slice()`:

```
slice(flights, 1:10)
#> Source: local data frame [10 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Операция `arrange()`, работает аналогично `filter()` за исключением того, что вместо выбора строк она переупорядочивает их. Функция получает на вход имя таблицы данных и список имён колонок (или более сложное выражение) для упорядочения по ним. Если будет указано больше одной колонки, то каждая следующая колонка будет упорядочиваться в пределах каждого отдельного набора значений предыдущих:

```
arrange(flights, year, month, day)
#> Source: local data frame [336,776 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Чтобы задать порядок по убыванию необходимо использовать `desc()`:

```
arrange(flights, desc(arr_delay))
#> Source: local data frame [336,776 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   9      641      1301     1242      1272      HA  N384HA
#> 2  2013     6  15     1432      1137     1607      1127      MQ  N504MQ
#> 3  2013     1  10     1121      1126     1239      1109      MQ  N517MQ
#> 4  2013     9  20     1139      1014     1457      1007      AA  N338AA
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Операция `arrange()` – это простая обёртка над `order()`, которая позволяет упростить команду.

При работе с данными обычно интересуют только некоторые столбцы. Операция `select()` позволяет выбрать некоторые из них:

```
# Select columns by name
select(flights, year, month, day)
#> Source: local data frame [336,776 x 3]
#>
#>    year month day
#> 1  2013     1   1
#> 2  2013     1   1
#> 3  2013     1   1
#> 4  2013     1   1
#> ..  ...   ... ...
# Select all columns between year and day (inclusive)
select(flights, year:day)
#> Source: local data frame [336,776 x 3]
#>
#>    year month day
#> 1  2013     1   1
#> 2  2013     1   1
#> 3  2013     1   1
#> 4  2013     1   1
#> ..  ...   ... ...
# Select all columns except those from year to day (inclusive)
select(flights, -(year:day))
#> Source: local data frame [336,776 x 13]
#>
#>    dep_time dep_delay arr_time arr_delay carrier tailnum flight origin
#> 1       517         2      830        11      UA  N14228   1545    EWR
#> 2       533         4      850        20      UA  N24211   1714    LGA
```

```
#> 3       542         2      923        33      AA  N619AA   1141    JFK
#> 4       544        -1     1004       -18      B6  N804JB    725    JFK
#> ..      ...       ...      ...       ...     ...     ...    ...    ...
#>    dest
#> 1   IAH
#> 2   IAH
#> 3   MIA
#> 4   BQN
#> ..  ...
#> Variables not shown: air_time (dbl), distance (dbl), hour (dbl), minute
#>   (dbl)
```

Эта функция работает подобно аргументу `select` в `base::subset()`. Отдельная функция добавлена для поддержания стройности идеологии `dplyr`, заключающейся в наличии функций каждая из которых делает только одну конкретную операцию и наиболее подходящим образом.

С `select()` используется ряд вспомогательных, такие как `starts_with()`, `ends_with()`, `matches()` и `contains()`. Они упрощают получение больших блоков колонок которые удовлетворяют некоторому критерию.

При помощи `select()` можно переименовывать колонки используя именованные аргументы:

```
select(flights, tail_num = tailnum)
#> Source: local data frame [336,776 x 1]
#>
#>    tail_num
#> 1    N14228
#> 2    N24211
#> 3    N619AA
#> 4    N804JB
#> ..      ...
```

но поскольку она отбрасывает все явно не указанные аргументы, используйте вместо неё для этого `rename()`:

```
rename(flights, tail_num = tailnum)
#> Source: local data frame [336,776 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tail_num
#> 1  2013     1   1      517         2      830        11      UA   N14228
#> 2  2013     1   1      533         4      850        20      UA   N24211
#> 3  2013     1   1      542         2      923        33      AA   N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6   N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...      ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

При помощи `select()` можно переименовывать колонки используя именованные аргументы Чаще всего `select()` используют для выяснения какие значения принимают значения колонки. В частности это удобно использовать

совместно с `distinct()`, операцией, которая возвращает только строки с уникальными значениями.

```
distinct(select(flights, tailnum))
#> Source: local data frame [4,044 x 1]
#>
#>    tailnum
#> 1   N14228
#> 2   N24211
#> 3   N619AA
#> 4   N804JB
#> ..     ...
distinct(select(flights, origin, dest))
#> Source: local data frame [224 x 2]
#>
#>    origin dest
#> 1     EWR  IAH
#> 2     LGA  IAH
#> 3     JFK  MIA
#> 4     JFK  BQN
#> ..    ...  ...
```

Бывает полезно не только выбрать интересующие колонки, но и вычислить на их основании значения некоторой другой колонки. Для этих целей предназначена `mutate()`:

```
mutate(flights,
  gain = arr_delay - dep_delay,
  speed = distance / air_time * 60)
#> Source: local data frame [336,776 x 18]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl), gain (dbl), speed (dbl)
```

Операция `dplyr::mutate()` работает аналогично `base::transform()`. Главное отличие между `mutate()` и `transform()` в том что первая может ссылаться в вычислениях на колонку которая создаётся в рамках того же вызова функции:

```
mutate(flights,
  gain = arr_delay - dep_delay,
  gain_per_hour = gain / (air_time / 60)
)
#> Source: local data frame [336,776 x 18]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl), gain (dbl),
#>   gain_per_hour (dbl)
```

```
transform(flights,
  gain = arr_delay - delay,
  gain_per_hour = gain / (air_time / 60)
)
#> Error: object 'gain' not found
```

Если вам нужно сохранить только новые колонки используйте `transmute()`:

```
transmute(flights,
  gain = arr_delay - dep_delay,
  gain_per_hour = gain / (air_time / 60)
)
#> Source: local data frame [336,776 x 2]
#>
#>    gain gain_per_hour
#> 1     9      2.378855
#> 2    16      4.229075
#> 3    31     11.625000
#> 4   -17     -5.573770
#> ..  ...           ...
```

Ещё одна операция – `summarise()`, она просто сворачивает таблицу данных в одну строку:

```
summarise(flights,
  delay = mean(dep_delay, na.rm = TRUE))
#> Source: local data frame [1 x 1]
#>
#>      delay
#> 1 12.63907
```

Эта функция в точности соответствует `plyr::summarise()`.

Вы можете воспользоваться `sample_n()` или `sample_frac()` для получения случайного набора строк, фиксированного числа с `sample_n()` или в фиксированной пропорции от всего набора с `sample_frac()`.

```
sample_n(flights, 10)
#> Source: local data frame [10 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1  21      652        -8      946       -11      B6  N636JB
#> 2  2013     5  17     2144        63       47        31      UA  N841UA
#> 3  2013    12   6     1521        -8     1721       -19      EV  N11165
#> 4  2013     2   1      703        -2     1035        14      B6  N591JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
sample_frac(flights, 0.01)
#> Source: local data frame [3,368 x 16]
#>
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     4  21      715         0      903       -13      DL  N309US
#> 2  2013    12  18     2041        71     2313        38      AA  N482AA
#> 3  2013     3  30     1833        13     2113       -22      AS  N549AS
#> 4  2013     2  17     1859        75     2051       103      UA  N78509
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

Аргумент `replace = TRUE` позволяет повторно выбирать одну и ту же строку (как при бутстрепинге), опционально можно указать вес в итоговой выборке каждой исходной строки при помощи аргумента `weight`.

Порядок выполнения работы

Задания необходимо выполнить встроенными средствами языка программирования R и функциями, реализованными в библиотект dplyr.

Информация о таблицах данных (data.frame) и библиотеке dplyr содержится в документации:

- введение в язык программирования R https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Data-frames;
- библиотека dplyr https://cran.r-project.org/web/packages/dplyr/index.html.

Список заданий для самостоятельного выполнения

В таблице 2.1 представлены варианты наборов данных для самостоятельной работы. Задание на работу:

1. Выполните команды представленные в теоретической части работы.
2. Загрузить данные указанные в табл. 2.1 в R.
3. Представить описание данных (название, физическое описание, тип переменной, единицы измерения).
4. Приведите примеры выполнения операций над данные из datasets над структурой data.frame в соответствии со своим вариантом (табл. 2.1).
5. Приведите примеры выполнения операций над внешними данными с использованием библиотеки dplyr в соответствии со своим вариантом (табл. 2.1).
6. Оцените время выполнения команд с использованием стандартных функций R и библиотеки dplyr.

Примерные варианты индивидуальных заданий

Таблица 2.1. Варианты наборов данных для обработки

№	Данные из datasets	Внешние данные
	airmiles	Latest Global Temperatures (https://www.kaggle.com/datasets/csafrit2/latest-global-temperatures)
	AirPassengers	Finance companies in India (https://www.kaggle.com/datasets/anuragbantu/finance-companies-in-india)
3.	airquality	Black Friday Sales EDA(https://www.kaggle.com/datasets/pranavuikey/black-friday-sales-eda)
4.	anscombe	Football Transfers from 1992/93 to 2021/22 seasons(https://www.kaggle.com/datasets/cbhavik/football-transfers-from-199293-to-202122-seasons)
5.	attenu	World, Region, Country GDP/GDP per capita (https://www.kaggle.com/datasets/tmishinev/world-country-gdp-19602021)
6.	attitude	Air Pollution Level (https://www.kaggle.com/datasets/totoro29/air-pollution-level)
7.	austres	Stack Overflow Developer Survey 2011-2022 (https://www.kaggle.com/datasets/yasirabdaali/stack-overflow-developer-survey-20112022)
8.	beaver1	Financing Healthcare (https://www.kaggle.com/datasets/programmerrdai/financing-healthcare)
9.	beaver2	Law School Admissions Bar Passage (https://www.kaggle.com/datasets/danofer/law-school-admissions-bar-passage)
10.	beavers	Vending Machine Sales (https://www.kaggle.com/datasets/awesomeasingh/vending-machine-sales)
	BJsales	Global Companies dataset (https://www.kaggle.com/datasets/narayan63/global-companies-dataset)
	BJsales.lead	Smoke Detection Dataset (https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset)
	BOD	Netflix Data: Cleaning, Analysis and Visualization (https://www.kaggle.com/datasets/ariyoomotade/netflix-data-cleaning-analysis-and-visualization)
	cars	World Population by Countries Dataset (1960-2021) (https://www.kaggle.com/datasets/kaggleashwin/population-dataset)

	ChickWeight	Online Retails Sale Dataset (https://www.kaggle.com/datasets/rohitmahulkar/online-retails-sale-dataset)
	chickwts	Spotify unpopular songs (https://www.kaggle.com/datasets/estienneggx/spotify-unpopular-songs)
	CO2	Chocolate Bar Ratings (https://www.kaggle.com/datasets/evangower/chocolate-bar-ratings)
	co2	Airbnb Open Data (https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata)
	crimtab	IMDB Movies (https://www.kaggle.com/datasets/totoro29/imdb-movies)
	DNase	Youtube Statistics (https://www.kaggle.com/datasets/advaypatil/youtube-statistics)

Список литературы

1. *Уикем Х., Гроулмунд Г.* Язык R в задачах науки о данных импорт, подготовка, обработка, визуализация и моделирование данных. – СПб.: Диалектика, 2018. – 592 с.
2. *Митина О.А.* Языки программирования для статистической обработки данных [Электронный ресурс]: Учебное пособие / Митина О.А., Юрченков И.А. — М.: МИРЭА – Российский технологический университет, 2020. — 191 с.
3. *Шишкин В.А.* Математические пакеты: R [Электронный ресурс] : учебное пособие / В. А. Шишкин ; Пермский государственный национальный исследовательский университет. – Электронные данные. – Пермь, 2023. – 15,0 Мб ; 225с. – Режим доступа: http://www.psu.ru/files/ docs/science/books/uchebnie-posobiya/SHishkin-Matematicheskie-pakety-R.pdf
4. *Зарядов И.С.* Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика: Учебно-методическое пособие М.: Издательство Российского университета дружбы народов, 2010. – 207 с.