Course: R Language in Computational Probability and Statistics

Lecture 7: Data manipulation. Package dplyr

Lecturer: Nataliia Kruhlova

Група функцій для обробки даних

Aggregate(x,by,FUN) — розбиває таблицю на окремі набори даних, застосовує до цих наборів функцію **FUN** .

Stack(x, ...) — перетворює дані, представлені в об'єкті у в вигляді окремих стовпчиків, в таблицю даних.

Unstack(x, ...) — виконує операцію обернену до функції **stack**.

Reshape(x, ...) — перетворює таблицю даних, в якій повторні значення будь-якої величини записані в окремих стовпчиках таблиці, в таблицю, в якій повторні значення йдуть один під одним в межах одного стовпчика.

Приклад. Розглянемо дані **mtcars** і візьмемо в якості змінних, які розбивають кількісну змінну **qsec** на групи, такі змінні: **am** приймає два значення і **vs** приймає два значення. В результаті одержуємо 4 групи розбиття. Розрахуємо мінімум для кожної групи.

```
> aggregate(qsec~am+vs,mtcars,min)
  am vs  qsec
1  0  0 15.41
2  1  0 14.50
3  0  1 18.30
4  1  1 16.90
```

Пакет dplyr

Функції для роботи з фреймами часто використовують разом. Наприклад, нам потрібно дані **iris** відсортувати по змінній Sepal.Length, залишивши тільки ті значення змінних, для яких значення Petal.Length менше 1.8, а також залишити тільки два стовпчика Sepal.Length і Petal.Length. З відібраної частини набору даних вивести перших 5 рядків.

Приклад.

```
> new<-new[new$Petal.Length<1.8,c(1,3)]
> new<-iris[order(iris$Sepal.Length),]
> new<-new[new$Petal.Length<1.8,c(1,3)]
> new[1:5,]
   Sepal.Length Petal.Length
14          4.3          1.1
9           4.4          1.4
39          4.4          1.3
43          4.4          1.3
42          4.5          1.3
```

Розв'язок виглядає доволі складно. Проблема в тому, що при накладанні декількох функцій, ми можемо не зрозуміти, до якої змінної відноситься та чи інша функція. Код можна зробити більш зрозумілим, якщо створювати проміжні змінні. Але це буде зайве завантаження пам'яті змінними, не всі з яких нам будуть надалі потрібними.

Спеціально для роботи з базами даних було створено пакет **dplyr**. В ньому є оператор %>% (Ctrl + Shift + m), який дозволяє поступово розв'язувати задачу. Він все, що написано перед ним, в якості аргументу відправляє в наступну функцію.

Переваги цього пакету при роботі з фреймами:

I. виділяє найбільш загальні операції маніпулювання даними, які дозволяють зосередитися на вирішенні завдання, а не на маніпулюванні;
II. пропонує набір простих функцій, що відповідають найбільш загальним атомарним операціям маніпулювання даними;
III. використовує ефективні сховища даних, щоб максимально скоротити час очікування результату.

Основні функції

Табл. 1

Функція	**Опис**

mutate	Додавання нових змінних
select	Вибір необхідних змінних (стовпчиків)
filter	Відбір необхідних записів (рядків)
arrange	Сортування рядків
rename	Перейменування змінних
slice	Вибір записів за позицією
distinct	Повертає конкретний запис
rows_insert	Додавання нових записів
inner_join, **left_join,** **right_join,** **full_join**	Операції з’єднання
group_by	Групує дані
summarise	Дає загальну статистичну інформацію

В [1] вказано, що аргументами функцій filter і select можуть бути не тільки логічні вирази, але і такі функції:

Табл. 2

Функція	**Опис**
starts_with(expr)	Вибираються змінні, імена яких починаються на expr
ends_with(expr)	Вибираються змінні, імена яких закінчуються на expr

contains(expr)	Вибираються змінні, імена яких містять expr
num_range	Вибір стовпців з певним діапазоном чисел у їх назвах
any_vars	Вибір рядків, де хоча б один стовпець задовольняє вказану умову.
all_vars	Вибір стовпчиків, де всі стовпчики задовольняють вказану умову.
between	Вибір значень, які знаходяться у певному діапазоні
across	До конкретного набору стовпчиків застосовується умова відбору
everything	Вибір всіх стовпчиків

Функції групи slice

Табл. 3

slice()	Вибирає записи у фреймі за номерами рядків
slice_head()	Вибирає перші записи у фреймі
slice_tail()	Вибирає записи з кінця фрейму
slice_min()	Вибирає мінімум у стовпчику
slice_max()	Вибирає максимум у стовпчику
slice_random()	Вибирає випадкові рядки

Розглянемо, як за допомогою цього пакету розв’язати приклад, що згадувався вище.

Приклад.

```
> iris %>%
+   filter(Petal.Length < 1.8) %>%
+   arrange(Sepal.Length) %>%
+   select(Sepal.Length, Petal.Length)%>%
+   slice(1:5)
  Sepal.Length Sepal.Width
1          4.3         1.1
2          4.4         1.4
3          4.4         1.3
4          4.4         1.3
5          4.5         1.3
```

На першому етапі ми відправляємо дані **iris** у функцію **filter**, в якій ми вказуємо умови фільтрації. Далі результат відправляємо на сортування по змінній **Sepal.Length**. І на передостанньому етапі ми відбираємо необхідні стовпчики. В кінці виводимо необхідну кількість рядків.

Приклад. З даних **mtcars** потрібно вибрати тільки 4 стовпчики: **mpg, hp, am, vs**. Потрібно залишити тільки спостереження, для яких **mpg**<15 і **hp** > 110. Відсортуйте таблицю по змінній **mpg** і залиште перших 15 рядків. Змінну **mpg** перейменуйте в **Миль на галон**, а змінну **hp** в **Потужність**. Новий набір даних збережіть в змінну my_data.

```
> my_data <- mtcars %>%
+   select(mpg, hp, am, vs) %>%
+   filter(mpg<15,hp>110)%>%
+   arrange(desc(mpg)) %>%
+   slice(1:15)%>%
+   rename(`Миль на галон`=mpg)%>%
+   rename(`Потужність`=hp)
> my_data
                  Миль на галон Потужність am vs
Chrysler Imperial          14.7         230  0  0
Duster 360                 14.3         245  0  0
Camaro Z28                 13.3         245  0  0
Cadillac Fleetwood         10.4         205  0  0
Lincoln Continental        10.4         215  0  0
```

Часто виникає необхідність звернутися одразу до декількох змінних в наборі даних і не дуже зручно їх прописувати вручну. Можна використати функції групи **summarise**.

Приклад. Обчисліть дисперсії і медіани для всіх змінних даних Iris в залежності від виду квіток.

```
> group_by(iris, Species) %>%
+    summarise_all(list(var, median))
# A tibble: 3 × 9
  Species    Sepal.Leng…¹ Sepal…² Petal…³ Petal…⁴ Sepal…⁵ Sepal…⁶ Petal…⁷
Petal…⁸
  <fct>             <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
1 setosa           0.124  0.144   0.0302  0.0111     5       3.4     1.5
0.2
2 versicolor       0.266  0.0985  0.221   0.0391     5.9     2.8     4.35
1.3
3 virginica        0.404  0.104   0.305   0.0754     6.5     3       5.55
2
# … with abbreviated variable names ¹Sepal.Length_fn1, ²Sepal.Width_fn1,
#   ³Petal.Length_fn1, ⁴Petal.Width_fn1, ⁵Sepal.Length_fn2, ⁶Sepal.Width_fn2,
#   ⁷Petal.Length_fn2, ⁸Petal.Width_fn2
```

Давайте розглянемо ще цікаві методи пакету **dplyr** на прикладах.

Приклад. Згрупуйте дані **Iris** по видам рослин і знайдіть дисперсію кожної числової змінної:

```
> New1 <- iris %>%
+    group_by(Species)%>%
+    summarise_if(is.numeric, var)
>
> New1
# A tibble: 3 x 5
  Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
  <fct>             <dbl>       <dbl>        <dbl>       <dbl>
1 setosa           0.124       0.144        0.0302      0.0111
2 versicolor       0.266       0.0985       0.221       0.0391
3 virginica        0.404       0.104        0.305       0.0754
```

Приклад. Створіть набір даних, що містить відомості про продажі деяких товарів певними клієнтами. Обчисліть середню суму, витрачену кожним клієнтом.

```
> data_sales<- data.frame(
+   customer_id = sample(1:5,10,replace=T),
+   year = sample(2010:2024,10,replace=T),
+   sales =sample(100:1000,10,replace=T))
> average_sales <- data_sales%>%
+   group_by(customer_id) %>%
+   summarise(average_sales = mean(sales))
> average_sales
# A tibble: 4 × 2
  customer_id average_sales
        <int>         <dbl>
1           1          702.
2           2          544.
3           3          247.
4           5          649
```

Приклад. Завантажте набір даних **diamonds** з пакету **ggplot2.** Обчисліть середню ціну за карат для кожної комбінації кольору та якості, при цьому враховуючи тільки діаманти з вагою більше 1 карату.

```
> library(ggplot2)
> diamonds %>%
+   filter(carat > 1)%>%
+   group_by(color, cut) %>%
+   summarise(average_price= mean(price / carat))%>%
  color cut       average_price
  <ord> <ord>           <dbl>
1 D     Fair            5415.
2 D     Good            5785.
3 D     Very Good       6789.
4 D     Premium         6548.
5 D     Ideal           7546.
```

Список джерел

1) Grolemund, G., & Wickham, H. (2017). *R for Data Science*. O'Reilly Media.