SPRAWOZDANIE

Zajęcia: Zbiory Big Data i Eksploracja Danych

Prowadząca: dr inż. Ruslana Ziubina

Laboratorium nr 2 Data rozpoczęcia: 3.11.2023 Temat: Eksploracja danych – dalsze informacje o języku R	Rafał Klinowski Informatyka II stopień, stacjonarne, Semestr 2, gr. a

Poszczególne ćwiczenia będą wykonywane w pliku źródłowym edytowanym przy pomocy środowiska RStudio, opisanego w poprzedniej części laboratorium.

Ćw. 1.

Testowanie przykładowego zbioru „airquality” – wczytanie, uzyskanie podstawowych informacji. Sposoby dostępu do danych w data.frame (na przykład niektórych kolumn czy wierszy), sprawdzenie rozmiaru danych (ile wierszy i kolumn), uzyskanie przykładowych wierszy, wybór warunkowy.

```
> str(airquality)
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
> airquality[100:105, 5:6]
    Month Day
100     8   8
101     8   9
102     8  10
103     8  11
104     8  12
105     8  13
```

Rysunek 1. Uzyskanie podstawowych informacji o kolumnach zbioru „airquality” oraz wierszy 100-105 z kolumnami 5-6.

```
> dim(airquality)
[1] 153   6
> dim(airquality[,])
[1] 153   6
> dim(airquality[5:6])
[1] 153   2
```

Rysunek 2. Przykłady użycia funkcji dim() do sprawdzenia rozmiaru danych (ilości kolumn i wierszy).

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> head(airquality[5:6])
  Month Day
1     5   1
2     5   2
3     5   3
4     5   4
5     5   5
6     5   6
```

Rysunek 3. Przykłady użycia funkcji head() zwracającej kilka pierwszych wierszy danych.

```
> tail(airquality)
    Ozone Solar.R Wind Temp Month Day
148    14      20 16.6   63     9  25
149    30     193  6.9   70     9  26
150    NA     145 13.2   77     9  27
151    14     191 14.3   75     9  28
152    18     131  8.0   76     9  29
153    20     223 11.5   68     9  30
> tail(airquality[1])
    Ozone
148    14
149    30
150    NA
151    14
152    18
153    20
> tail(airquality[[1]])
[1] 14 30 NA 14 18 20
```

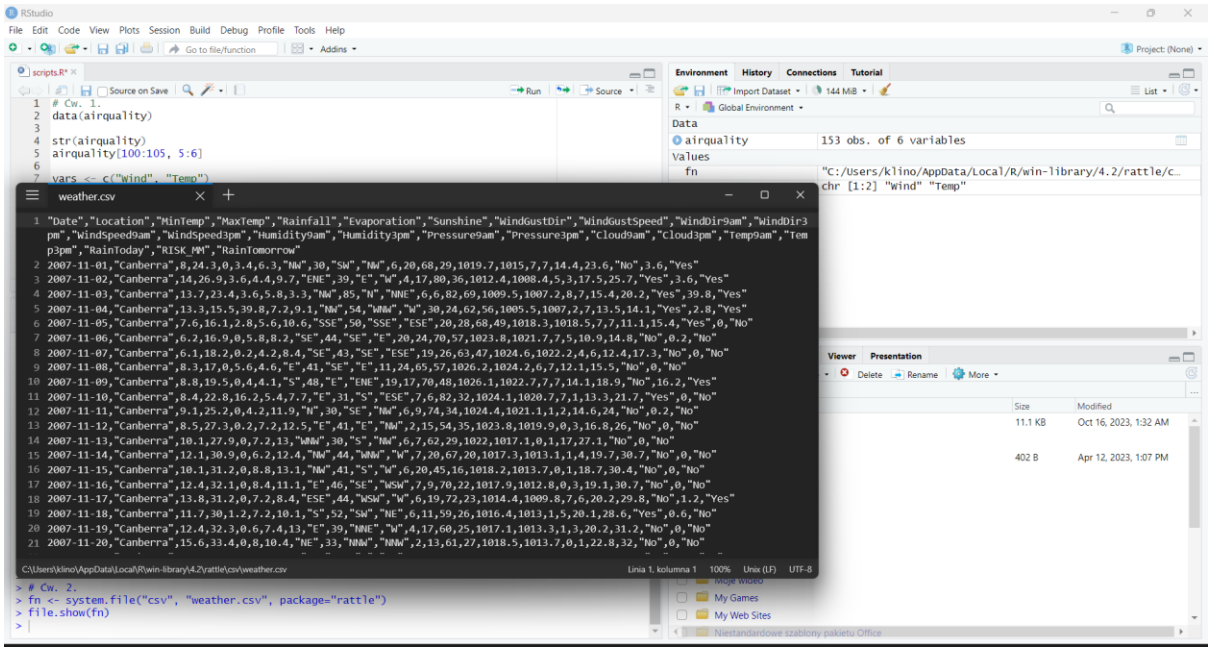
Rysunek 4. Przykłady użycia funkcji tail() zwracającej kilka ostatnich wierszy danych.

```
> vars <- c("Wind", "Temp")
> airquality[100:105, vars]
    Wind Temp
100 10.3   90
101  8.0   90
102  8.6   92
103 11.5   86
104 11.5   86
105 11.5   82
```

Rysunek 5. Wykorzystanie wektora nazw kolumn do wyboru ich ze zbioru danych.

Ćw. 2.

Wczytywanie danych z pliku tekstowego (np. pliku wartości oddzielonych przecinkami – CSV) z lokacji na dysku lub w Internecie. Funkcja „read.csv()”.



Rysunek 6. Otwarcie i pokazanie pliki tekstowego przy pomocy „file.show()”.

```
> data <- read.csv("https://rattle.togaware.com/weather.csv")
> head(data)
        Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir WindGustSpeed
1 2007-11-01 Canberra     8.0     24.3      0.0         3.4      6.3          NW            30
2 2007-11-02 Canberra    14.0     26.9      3.6         4.4      9.7         ENE            39
3 2007-11-03 Canberra    13.7     23.4      3.6         5.8      3.3          NW            85
4 2007-11-04 Canberra    13.3     15.5     39.8         7.2      9.1          NW            54
5 2007-11-05 Canberra     7.6     16.1      2.8         5.6     10.6         SSE            50
6 2007-11-06 Canberra     6.2     16.9      0.0         5.8      8.2          SE            44
  WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am
1         SW         NW            6           20          68          29      1019.7
2          E          W            4           17          80          36      1012.4
3          N        NNE            6            6          82          69      1009.5
4        WNW          W           30           24          62          56      1005.5
5        SSE        ESE           20           28          68          49      1018.3
6         SE          E           20           24          70          57      1023.8
  Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm RainToday RISK_MM RainTomorrow
1      1015.0        7        7    14.4    23.6        No     3.6          Yes
2      1008.4        5        3    17.5    25.7       Yes     3.6          Yes
3      1007.2        8        7    15.4    20.2       Yes    39.8          Yes
4      1007.0        2        7    13.5    14.1       Yes     2.8          Yes
5      1018.5        7        7    11.1    15.4       Yes     0.0           No
6      1021.7        7        5    10.9    14.8        No     0.2           No
```

Rysunek 7. Otwarcie pliku tekstowego z lokalizacji w Internecie.

Ćw. 3.

Wczytywanie danych z pliku tekstowego do ramki danych. Funkcje „read.table()” oraz „read.csv()”. Opcjonalne parametry obu funkcji dotyczące wczytywania danych.

```
> # setwd(...)
> t1 <- read.table("titanic.csv", header=TRUE, sep=",")
> t2 <- read.csv("titanic.csv", header=TRUE, sep=",", quote="\"", dec=".",
+                fill=TRUE, comment.char="")
> t3 <- read.csv("titanic.csv")
> head(t2)
  PassengerId Survived Pclass                                                 Name    Sex Age
1           1        0      3                              Braund, Mr. Owen Harris   male  22
2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38
3           3        1      3                               Heikkinen, Miss. Laina female  26
4           4        1      1       Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35
5           5        0      3                             Allen, Mr. William Henry   male  35
6           6        0      3                                     Moran, Mr. James   male  NA
  SibSp Parch           Ticket    Fare Cabin Embarked
1     1     0        A/5 21171  7.2500              S
2     1     0         PC 17599 71.2833   C85        C
3     0     0 STON/O2. 3101282  7.9250              S
4     1     0           113803 53.1000  C123        S
5     0     0           373450  8.0500              S
6     0     0           330877  8.4583              Q
```

Rysunek 8. Wczytanie danych z pliku CSV na 3 sposoby. Możliwe jest sprecyzowanie wiele parametrów opisujących w jaki sposób dane zostaną wczytane i zinterpretowane. Konieczne jest najpierw ustawienie katalogu roboczego.

```
> sum(t2[, "Survived"])
[1] 342
```

Rysunek 9. Przykładowa operacja przeprowadzona na wczytanych danych.

Ćw. 4.

Wczytywanie danych ze schowka systemu operacyjnego (ang. „clipboard”). Umożliwia ono skopiowanie danych w systemie, a następnie zapisanie ich jako obiekt w R.

```
> schowek <- read.table(file("clipboard"), header=TRUE, sep="\t")
Komunikat ostrzegawczy:
W poleceniu 'read.table(file("clipboard"), header = TRUE, sep = "\t")':
  niekompletna końcowa linia znaleziona przez 'readTableHeader' w 'clipboard'
> schowek
    Age SpectaclePrescrip Astigmatism TearProdRate ContactLens
1 young             myope          no      reduced        none
2 young             myope          no       normal        soft
3 young             myope         yes      reduced        none
```

Rysunek 10. Wczytanie danych ze schowka.

Uzyskane ostrzeżenie jest spowodowane niekompletnym znakiem końca linii. Aby się go pozbyć, należy skopiować dane w taki sposób, by nie kopiować znaków końca linii – na przykład kopiując całość pliku.

Ćw. 5.

Funkcja read.table wykorzystana do wczytania danych z pliku do tabeli. Wyświetlenie podstawowych informacji o wczytanych danych.

```
> Veg <- read.table(file="Vegetation2.txt", header=TRUE)
> names(Veg)
 [1] "TransectName" "Samples"      "Transect"     "Time"         "R"            "ROCK"
 [7] "LITTER"       "ML"           "BARESOIL"     "FallPrec"     "SprPrec"      "SumPrec"
[13] "WinPrec"      "FallTmax"     "SprTmax"      "SumTmax"      "WinTmax"      "FallTmin"
[19] "SprTmin"      "SumTmin"      "WinTmin"      "PCTSAND"      "PCTSILT"      "PCTOrgC"
> str(Veg)
'data.frame':   58 obs. of  24 variables:
 $ TransectName: chr  "A_22_58" "A_22_62" "A_22_67" "A_22_74" ...
 $ Samples     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Transect    : int  1 1 1 1 1 1 1 2 2 2 ...
 $ Time        : int  1958 1962 1967 1974 1981 1994 2002 1958 1962 1967 ...
 $ R           : int  8 6 8 8 10 7 6 5 8 6 ...
 $ ROCK        : num  27 26 30 18 23 26 39 25 24 21 ...
 $ LITTER      : num  30 20 24 35 22 26 19 26 24 16 ...
 $ ML          : int  0 0 0 0 4 0 4 0 2 1 ...
 $ BARESOIL    : num  26 28 30 16 9 23 19 33 29 41 ...
 $ FallPrec    : num  30.2 99.6 43.4 54.9 24.4 ...
 $ SprPrec     : num  75.4 56.1 65 58.7 87.6 ...
 $ SumPrec     : num  125.5 95 112.3 70.3 81.8 ...
 $ WinPrec     : num  39.6 107.4 76.7 90.7 46 ...
 $ FallTmax    : num  17 14.6 18.4 17.2 18.5 ...
 $ SprTmax     : num  15.8 15.2 12.8 14 14.3 ...
 $ SumTmax     : num  25.2 24.9 25.5 26.7 26 ...
 $ WinTmax     : num  3.47 1.16 3.09 2.46 5.72 ...
 $ FallTmin    : num  0.49 -0.18 1.23 1.43 1.09 ...
 $ SprTmin     : num  0.36 0.18 -1.86 -0.53 0.75 ...
 $ SumTmin     : num  6.97 6.4 7.12 7.2 6.9 ...
 $ WinTmin     : num  -8.54 -10.76 -8.5 -8.28 -7.56 ...
 $ PCTSAND     : int  24 24 24 24 24 24 24 20 20 20 ...
 $ PCTSILT     : int  30 30 30 30 30 30 30 34 34 34 ...
 $ PCTOrgC     : num  0.0346 0.0346 0.0346 0.0346 0.0346 ...
```

Rysunek 11. Wczytanie zbioru danych „Vegetation2" przy pomocy funkcji read.table() i wypisanie podstawowych informacji o kolumnach.

Ćw. 6.

Przeprowadzanie operacji na zbiorze danych. Skorzystanie z funkcji tapply().

tapply() pozwala na proste zastosowanie danej funkcji na zbiorze danych wybranych ze zbioru głównego – zamiast wykonywać podobną funkcję, zmieniając zaledwie nazwy kolumn czy wierszy, możliwe jest zastosowanie tapply(), aby zrobić to szybciej uzyskując identyczny efekt.

```
> m <- mean(Veg$R)
> m1 <- mean(Veg$R[Veg$Transect == 1])
> m2 <- mean(Veg$R[Veg$Transect == 2])
> m3 <- mean(Veg$R[Veg$Transect == 3])
> m4 <- mean(Veg$R[Veg$Transect == 4])
> m5 <- mean(Veg$R[Veg$Transect == 5])
> m6 <- mean(Veg$R[Veg$Transect == 6])
> m7 <- mean(Veg$R[Veg$Transect == 7])
> m8 <- mean(Veg$R[Veg$Transect == 8])
> ms <- c(m1, m2, m3, m4, m5, m6, m7, m8)
> ms
[1]  7.571429  6.142857 10.375000  9.250000 12.375000 11.500000 10.500000 11.833333
```

Rysunek 12. Użycie funkcji mean() do policzenia kilku średnich dla różnych wartości danej kolumny.

```
> tapply(Veg$R, Veg$Transect, mean)
       1        2        3        4        5        6        7        8
 7.571429 6.142857 10.375000 9.250000 12.375000 11.500000 10.500000 11.833333
```

Rysunek 13. Kod źródłowy realizujący powyższe zadanie w jednej linii z użyciem funkcji tapply().

```
> tapply(X = Veg$R, INDEX = Veg$Transect, FUN = mean)
       1        2        3        4        5        6        7        8
7.571429 6.142857 10.375000 9.250000 12.375000 11.500000 10.500000 11.833333
```

Rysunek 14. Alternatywny zapis funkcji z nazwami parametrów.

```
> Me <- tapply(Veg$R, Veg$Transect, mean)
> Sd <- tapply(Veg$R, Veg$Transect, sd)
> Le <- tapply(Veg$R, Veg$Transect, length)
> params <- c(Me, Sd, Le)
> params
       1        2        3        4        5        6        7        8
7.5714286 6.1428571 10.3750000 9.2500000 12.3750000 11.5000000 10.5000000 11.8333333
       1        2        3        4        5        6        7        8
1.3972763 0.8997354 3.5831949 2.3145502 2.1339099 2.2677868 3.1464265 2.7141604
       1        2        3        4        5        6        7        8
7.0000000 7.0000000 8.0000000 8.0000000 8.0000000 8.0000000 6.0000000 6.0000000
```

Rysunek 15. Obliczenie kilku parametrów przy pomocy tapply() i wyświetlenie ich.

Ćw. 7.

Funkcje lapply() i sapply().

Obie funkcje są podobne do funkcji tapply() – pierwsza z nich daje w wyniku listę, druga z nich daje wektor kilku wartości. Różnicą jest również to, że obie funkcje – w przeciwieństwie to tapply() – wykonują podaną funkcję dla jednej lub więcej zmiennych dla wszystkich obserwacji, podczas gdy tapply() wykonuje ją dla podzbiorów obserwacji zmiennej.

```
> sapply(Veg[, 5:9], FUN=mean)
        R      ROCK    LITTER        ML  BARESOIL
 9.965517 20.991379 22.853448  1.086207 17.594828
> lapply(Veg[, 5:9], FUN=mean)
$R
[1] 9.965517

$ROCK
[1] 20.99138

$LITTER
[1] 22.85345

$ML
[1] 1.086207

$BARESOIL
[1] 17.59483
```

Rysunek 16. Wykorzystanie funkcji sapply() i lapply() oraz porównanie ich wyników.

Ćw. 8.

1. Utwórz listę temp…

Do utworzenia listy wykorzystano funkcję lapply() z parametrem „sample” (losowa generacja danych), a następnie dodano do jej wyniku nagłówki.

```
> temp <- lapply(c(5,5,5,5,5,5,5), FUN=sample)
> names(temp) <- c("Poniedziałek", "Wtorek", "Środa", "Czwartek", "Piątek", "Sobota",
+                  "Niedziela")
> temp
$Poniedziałek
[1] 4 2 5 3 1

$Wtorek
[1] 3 4 2 1 5

$Środa
[1] 5 4 2 1 3

$Czwartek
[1] 2 4 3 5 1

$Piątek
[1] 5 2 3 1 4

$Sobota
[1] 4 1 5 2 3

$Niedziela
[1] 4 5 3 1 2
```

Rysunek 17. Utworzenie listy przy pomocy funkcji lapply(). Jako parametr X przekazano 7-elementowy wektor zawierający liczby 5 (argumenty dla funkcji sample()).

2. Wygeneruj wektor z minimalnymi temperaturami każdego dnia…
3. Wygeneruj wektor z maksymalnymi temperaturami każdego dnia…

```
> mins <- sapply(temp, FUN=min)
> maxs <- sapply(temp, FUN=max)
>     mins
Poniedziałek       Wtorek        Środa     Czwartek       Piątek       Sobota    Niedziela
           1            1            1            1            1            1            1
> maxs
Poniedziałek       Wtorek        Środa     Czwartek       Piątek       Sobota    Niedziela
           5            5            5            5            5            5            5
```

Rysunek 18. Wykorzystanie funkcji sapply() do zwrócenia najmniejszych i największych temperatur w każdy z dni.

4. Wczytaj zbiór USArrests.csv...

Do wygenerowania sum wykorzystano funkcję sapply(), która obliczy sumy dla każdej z kolumn (nie podano w funkcji jako parametr ograniczenia). W zbiorze danych występowały cztery kolumny – „Murder”, „Assault”, „UrbanPop” i „Rape”, podczas gdy wiersze reprezentowały poszczególne stany. W wyniku uzyskano sumę poszczególnych kolumn bez rozróżnienia stanów.

```
> data(USArrests)
> results <- sapply(USArrests, FUN=sum)
> results
  Murder  Assault UrbanPop     Rape
   389.4   8538.0   3277.0   1061.6
```

Rysunek 19. Wykorzystanie funkcji sapply() z przekazanym argumentem „sum” do obliczenia osobno sum dla każdej z kolumn.

Wnioski.

Funkcje tapply(), sapply() i lapply(), z którymi zapoznałem się w ramach tego laboratorium, znacznie ułatwiają pracę z danymi i wykonywanie tych samych operacji wielokrotnie na różnych podzbiorach danych. Za ich pomocą można znacząco uprościć wykonywanie operacji takich jak: obliczenie sumy, średniej i wiele więcej, dla wszystkich lub niektórych kolumn zbioru danych, co bez wykorzystania tych funkcji wymagałoby wielokrotnego powtarzania tego samego kodu.

Wczytywanie zbiorów w R jest wyjątkowo proste i intuicyjne. W środowisko wbudowane jest kilka funkcji, których można użyć do wczytania danych z pliku tekstowego, CSV lub ramki danych, zarówno z lokacji na komputerze, jak i w Internecie. W związku z tym praca ze zbiorami danych odbywa się znacznie prościej, niż w innych podobnych środowiskach czy językach.

Całość laboratorium została przeprowadzona w RStudio, które znacznie ułatwia nie tylko tworzenie poleceń (dzięki kolorowaniu składni oraz podpowiadaniu nazw), ale również ich powtórzenie (w przypadku uruchomienia więcej niż raz) oraz podświetlenie zarówno ich wyników, jak i danych, na podstawie których te wyniki zostały uzyskane. Środowisko posiada również podgląd danych w formie tabeli lub jako wartość, co przydaje się w celu podejrzenia ich wyglądu i rozmiaru, jak również do weryfikacji wpisywanych poleceń.