Indeksowanie

Statystyka I z R

Bartosz Maćkiewicz

Indeksowanie

W tej części kursu będziemy zajmowac się indeksowaniem. Bardzo często chcemy z naszych danych "wybrać" tylko niektóre z nich - te obserwacje, które nas interesują. R jest językiem oferującym ogromne możliwości manipulacji danymi. Można "filtrować" dane na wiele sposobów, my jednak zajmiemy się zupełnie podstawowymi sposobami. Wykorzystamy podstawowe konstrukcje i nauczymy się tworzyć z nich bardziej skomplikowane.

Zaprezentuje tutaj pewne podejście do kwesti indeksowania/filtrowania danych, dzięki któremu żadne tego typu zadanie nie będzie straszne. Wymagać to będzie jednak pewnej koncentracji.

Wektory

Zacznijmy od prostego przypadku - stwórzmy pięcioelementowy wektor z liczbami.

```
x <- c(1001,1002,1003,1004,1005)
x
```

```
## [1] 1001 1002 1003 1004 1005
```

W jaki sposób wydobyć z tego wektora **tylko** jego trzeci element? Bardzo prosto! Możemy w tym celu użyć nawiasów kwadratowych.

```
x[3]
```

```
## [1] 1003
```

W przeciwieństwie do niektórych nie mamy eleganckiej możliwości wybierania elementów "od końca". Możemy jednak posłużyć się pewnym trikiem i wykorzystać funkcję `length`, która zwraca długość wektora.

```
# length(x) zwraca długość x, odejmujemy jeden, aby wybrać przedostatni
x[length(x)-1]
```

```
## [1] 1004
```

Świetnie nam poszło! Co jednak, gdybyśmy chcieli wybrać trzecią, czwartą i piątą liczbę? Spróbujmy w nawiasy kwadratowe wstawić wektor z więcej niż jednym elementem, który utworzymy za pomocą funkcji `c()`.

```
x[c(3,4,5)]
```

```
## [1] 1003 1004 1005
```

Bingo! W poprzednim notatniku poznaliśmy sposób generowania wektorów kolejnych liczb. Jednym z nich była konstrukcja `dolna_granica:gorna_granica`. Sprawdźmy w takim razie, czy zadziała ona w naszym przypadku.

```
x[3:5]
```

```
## [1] 1003 1004 1005
```

Doskonale! Dla wielu osób obeznanych z innymi językami programowania taka konstrukcja wygląda bardzo znajomo. Musimy jednak mieć świadomość, że w przypadku R podawanie *dwukropkowych* zakresów w nawiasach kwadratowych **nie jest** specjalną składnią do wybierania wycinków wektorów. Składnia z dwukropkiem tworzy po prostu wektor i ten właśnie wektor przekazujemy w nawiasach kwadratowych, aby wybrać wycinek naszego oryginalnego wektora.

Omówione do tej pory przykłady mogą wydawać się dość trywialne. W praktyce, analizując dane, rzadko wiemy, który element wektora chcemy wybrać. Zwykle chcemy kierować się jakimś kryterium. Rozważmy prosty przypadek. Załóżmy, że mamy wektor, w którym znajduje się wiek (w latach) uczestników badania. Moglibyśmy chcieć wybrać tylko te wartości, które są większe lub równe niż 18 po to, aby potem sprawdzić jak długi jest nasz wektor i dzięki temu dowiedzieć się, ilu mieliśmy uczestników badania.

```
age <- c(14, 18, 22, 18, 19, 33, 45, 14, 17, 16, 23, 26, 42)
```

Moglibyśmy oczywiście wyświetlić naszą zmienną i wybrać z niej tylko interesujące nas obserwacje, korzystając z poznanego przed chwilą sposobu. To jednak bardzo niepraktyczne - musielibyśmy przecież ręcznie liczyć, na których pozycjach w naszym wektorze znajdują się liczby większe niż 18! Czy da się to zrobić bardziej przebiegle? Okazuje się, że tak.

Przyjrzyjmy się temu, jak działają w R operatory arytmetyczne takie jak >, <, >=, itp.

```
age >= 18
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE
```

Otrzymaliśmy wektor wartości logicznych (`logical`). Wartość logiczna `TRUE` znajduje się na tych pozycjach, dla których w oryginalnym wektorze `age` nasz warunek (>=) był spełniony. Z kolei `FALSE` znajduje się na tych pozycjach, gdzie warunek spelniony nie był.

Okazuje się, że do indeksowania możemy użyć nie tylko wektorów z pozycjami, które nas interesują, ale również wektorów z wartościami logicznymi.

```
age
```

```
##  [1] 14 18 22 18 19 33 45 14 17 16 23 26 42
```

```
age >= 18
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE
```

```
age[age >= 18]
```

```
## [1] 18 22 18 19 33 45 23 26 42
```

W tym wypadku wyrażenie to zwraca te elementy z wektora `age`, które znajdują się na pozycjach, na których w wektorze `age >= 18` występuje wartość `TRUE`. Na początku może wydawać się to dość skomplikowane, ale podejście to otwiera niesamowicie dużo możliwości.

Wiemy już że w nawiasach kwadratowych mogą znajdować się dwa rodzaje wartości - albo wektor typu `integer` albo wektor typu `logical`.

Najtrudniejsze za nami.

Macierze

Czy możemy wykorzystać nasze umiejętności w sytuacji, gdy chcemy wybrać wartości z dwuwymiarowej macierzy? Zasada jest bardzo podobna. Musimy w nawiasach kwadratowych umieścić dwie wartości - numer(y) wiersza i numer(y) kolumny. Zilustrujmy to przykładem:

```
macierz <- matrix(1:20, 5,4)
macierz
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

```
# Pamiętamy o zasadzie: *wiersze pierwsze*!
macierz[3,4] # trzeci wiersz i czwarta kolumna
```

```
## [1] 18
```

Co jeśli chcemy wybrać wiersze lub kolumny w całości? R oferuje tutaj dość przyjazną składnie. Jeżeli zostawimy po którejś stronie przecinka w nawiasie kwadratowym puste miejsce, R potraktuje to jako wszystkie kolumny lub wiersze.

Przykładowo, jeśli chcemy wybrać drugą, trzecią oraz czwartą kolumnę, to możemy to musimy powiedzieć R, że chcemy: - wybrać wszystkie wiersze, - wybrać kolumny 2, 3 oraz 4.

Wynikiem takiej operacji będzie nowa macierz składająca się z 3 kolumn oraz takiej liczby wierszy, jaką miala oryginalna macierz.

```
# wszystkie wiersze, ostatnie trzy kolumny (używamy pustego miejsca)
macierz[,2:4]
```

```
##      [,1] [,2] [,3]
## [1,]    6   11   16
## [2,]    7   12   17
## [3,]    8   13   18
## [4,]    9   14   19
## [5,]   10   15   20
```

Dokładnie w ten sam sposób możemy zastosować do wierszy:

```
macierz[1:3,] # analogicznie pierwsze trzy wiersze
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
```

Sprobujmy zrobić coś nieco trudniejszego. Załóżmy, że chcemy stworzyć nową macierz, w której znajdą się wszystkie kolumny z oryginalnej macierzy, ale wyłącznie dla tych wierszy, dla których w drugiej kolumnie występowała wartość większa niż 6.

Początkowo może brzmieć to skomplikowanie, ale wystarczy, że wykorzystamy nasze dodatkowe umiejętności. Odpowiednie wyrażenie wygląda tak:

```
macierz[macierz[,2] >6,]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    7   12   17
## [2,]    3    8   13   18
## [3,]    4    9   14   19
## [4,]    5   10   15   20
```

Jak widzimy wyrażenie to prawidłowo wybrało trzeci oraz czwarty wiesz.

Spróbujmy prześledzić powyższy przykład krok po kroku.

Oto nasza macierz w całej okazałości:

```
macierz
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

Jej drugą kolumnę możemy wybrać, korzystając ze znanego nam sposobu.

```
macierz[ ,2]
```

```
## [1]  6  7  8  9 10
```

Następnie musimy sprawdzić, które wartości w tej kolumnie są większe niż 6. Wykorzystamy do tego standardowy operator arytmetyczny $>$.

```
macierz[ ,2] > 6
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE
```

Wiemy teraz, że w drugiej kolumnie wartości większe niż 6 znajdują się w trzecim i czwartym wierszu. Zauważmy, że wyrażenie to zwróciło wektor typu `logical`. Wobec tego możemy go użyć do wybrania z oryginalnej macierzy odpowiednich wierszy, w pewnym sensie składając wszystkie te wyrażenia w jedną całość.

```
macierz[macierz[,2] > 7, ]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3    8   13   18
## [2,]    4    9   14   19
## [3,]    5   10   15   20
```

Nie było to takie straszne!

Listy

Listy w przeciwieństwie do wektorów przechowują sekwencje wartości różnych typów. Możemy w niej przechowywać wektory różnych typów o różnej długości a nawet inne listy. W tym sensie są podobne do list z niektórych innych języków programowania. Listy tworzymy za pomocą funkcji `list`.

```
lista <- list(1:10, # pierwszy element: wektor typu `integer`
              # drugi element: wektor typu `character`
              c('Entliczek', "Pętliczek", 'Czerwony', 'Stoliczek'),
              TRUE, # trzeci element: wektor typu `logical`
              TRUE, # czwarty element: wektor typu `logical`
              list( # piąty element: lista
              # pierwszy element: wektor typu `character`
                c('Lista', "zagnieżdżona", "w", "drugiej", "liście")
              ))
lista
```

```
## [[1]]
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## [[2]]
```

```
## [1] "Entliczek" "Pętliczek" "Czerwony"  "Stoliczek"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [[5]][[1]]
## [1] "Lista"        "zagnieżdżona" "w"            "drugiej"      "liście"
```

Elementy listy możemy wybrać za pomocą podwójnego nawiasu kwadratowego `[[]]`. Przykładowo:

```
lista[[3]] # trzeci element listy
```

```
## [1] TRUE
```

W przypadku zagnieżdżonych list bardzo często chcemy wybrać pewien element pewnego elementu listy. Wówczas musimy użyć znanej już nam techniki. Spróbujmy wybrać słowo "Czerwony".

```
lista[[2]][3] # trzeci element drugiego elementu listy
```

```
## [1] "Czerwony"
```

Podobnie, jeżeli mamy listy zagnieżdżone, możemy użyć kilku indeksów jednocześnie. Wybierzmy więc słowo "drugiej":

```
lista[[5]][[1]][4] # czwarty element pierwszego elementu listy piątego elementu listy
```

```
## [1] "drugiej"
```

Elementom listy możemy nadawać nazwy. Dlatego w niektórych sytuacjach możemy traktować je jak słowniki z innych języków programowania, przechowujące pary klucz-wartość.

Nazwane listy również tworzymy przy pomocy funkcji `list`.

```
lista_nazwana <- list(imiona = c('Marek', 'Kasia', 'Sławek'),
                      czesci_ciala = c('Noga', 'Ręka', 'Palec', 'Głowa')
                      )
lista_nazwana
```

```
## $imiona
## [1] "Marek"  "Kasia"  "Sławek"
##
## $czesci_ciala
## [1] "Noga"  "Ręka"  "Palec" "Głowa"
```

W przypadku list z nazwanymi elementami, możemy wybierać je za pomocą liczb albo nazw.

```
lista_nazwana[[1]] # możemy wybierać cały czas za pomocą liczb
```

```
## [1] "Marek"  "Kasia"  "Sławek"
```

```
lista_nazwana[['imiona']] # ale często wygodniej będzien nam użyć nazw
```

```
## [1] "Marek"  "Kasia"  "Sławek"
```

```
lista_nazwana$imiona # krótszy, często spotykany zapis
```

```
## [1] "Marek"  "Kasia"  "Sławek"
```

Ramki danych (*data frame*)

W przypadku ramek danych, to wybieranie z nich elementów działa bardzo podobnie, co wybieranie elementow z macierzy oraz z list.

Muszę uprzedzić, że pod koniec lekcji kod stanie się niestety strasznie "makaroniasty". Istnieje wiele bibliotek do R, które ułatwiają tego rodzaju zadania. Zależy mi jednak, aby państwo uważnie prześledzili poniższe przykłady i spróbowali zrozumieć każdy krok oraz nauczyli się przeprowadzać "dekonstrukcje" poszczególnych elementów długiego i zagmatwanego wyrażenia.

W tym przykładzie będziemy korzystać z ramki danych `mtcars` dostępnej w standardowej instalacji R. Ten zbiór danych zawiera informacje o kilkunastu modelach samochodów. Możemy podejrzeć pierwsze 6 wierszy za pomocą funkcji `head`.

```
data(mtcars)
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Rozpoczniemy od przyjrzenia się naszemu zbiorowi danych. Zobaczmy, ile ma kolumn, ile wierszy i jak nazywają się poszczególne kolumny.

```
dim(mtcars)
```

```
## [1] 32 11
```

Mamy 32 wiersze oraz 11 kolumn. Pierwsza kolumna (z nazwami samochodów) to nazwa wiersza (`rownames`).

```
colnames(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

Za pomocą pojedynczych nawiasów kwadratowych możemy wybrać z danej ramki danych pewien jej podzbiór. Wybierzmy pierwszą kolumnę.

```
head(mtcars[1]) # domyślnie head zwraca pierwsze 6 wierszy
```

```
##                   mpg
## Mazda RX4         21.0
## Mazda RX4 Wag     21.0
## Datsun 710        22.8
## Hornet 4 Drive    21.4
## Hornet Sportabout 18.7
## Valiant           18.1
```

Spróbujmy wybrać drugą, czwartą i szóstą kolumnę umieszczając w nawiasach kwadratowych trzyelementowy wektor. Funkcja `tail` jest bardzo podobna do funkcji `head` i umożliwia wybranie ostatnich 6 wierszy.

```
tail(mtcars[c(2,4,6)])
```

```
##                cyl  hp    wt
## Porsche 914-2    4  91 2.140
## Lotus Europa     4 113 1.513
## Ford Pantera L   8 264 3.170
```

```
## Ferrari Dino     6 175 2.770
## Maserati Bora    8 335 3.570
## Volvo 142E       4 109 2.780
```

Jeżeli chcemy, to ramki danych indeksować możemy dokładnie tak jak macierze. Wybierzmy wartość z drugiego wiersza (Mazda RX4 WAG) czwartej kolumny (`hp` czyli liczba koni mechanicznych).

```
mtcars[2,4]
```

```
## [1] 110
```

Za pomocą podwójnych nawiasów kwadratowych możemy wybrać całą kolumnę jako wektor.

```
mtcars[[3]]
```

```
##  [1] 160.0 160.0 108.0 258.0 360.0 225.0 360.0 146.7 140.8 167.6 167.6 275.8
## [13] 275.8 275.8 472.0 460.0 440.0  78.7  75.7  71.1 120.1 318.0 304.0 350.0
## [25] 400.0  79.0 120.3  95.1 351.0 145.0 301.0 121.0
```

Do indeksowania możemy używać oczywiście również nazw. Wybierzmy dwie kolumny - opisujące liczbę biegów i liczbę koni mechanicznych.

```
head(mtcars[c('hp', 'gear')])
```

```
##                   hp gear
## Mazda RX4         110    4
## Mazda RX4 Wag     110    4
## Datsun 710         93    4
## Hornet 4 Drive    110    3
## Hornet Sportabout 175    3
## Valiant           105    3
```

Przekazując w nawiasach kwadratowych nazwę kolumny otrzymujemy ją jako wektor.

```
mtcars[['hp']]
```

```
##  [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230  66  52
## [20]  65  97 150 150 245 175  66  91 113 264 175 335 109
```

Nic nie stoi na przeszkodzie, żebyśmy wybrali zarówno kolumny, jak i wiersze. Robimy to tak samo jak w przypadku macierzy, oddzielając wartości przecinkiem.

```
mtcars[2:5, c(3,5)]
```

```
##                   disp drat
## Mazda RX4 Wag      160 3.90
## Datsun 710         108 3.85
## Hornet 4 Drive     258 3.08
## Hornet Sportabout  360 3.15
```

Ale uwaga! Jeśli wybierzemy tylko jedną konkretną wartość jako kolumnę (a nie zakres) to nie otrzymamy ramki danych z jednym elementem, ale konkretną wartość odpowiedniego typu. Bywa to czasami uciążliwe.

Zilustrujmy to na przykładzie. Jeżeli wybierzemy drugi wiersz kolumny 3, 4 i 5, to R zwróci nam ramkę danych.

```
mtcars[2,3:5]
```

```
##               disp  hp drat
## Mazda RX4 Wag  160 110  3.9
```

Z kolei jeśli wybierzemy wiersze 3, 4 i 5 oraz drugą kolumnę, to R zwróci nam wektor.

```
mtcars[3:5,2]
```

```
## [1] 4 6 8
```

Za pomocą poznanych narzędzi możemy łatwo filtrować dane. Wyobraźmy sobie, że interesuje nas ramka danych z tylko dwoma wartosciami - liczbą koni mechanicznych oraz mil na galon benzyny. Chcemy jednak wybrać wyłącznie te samochody, które mają więcej niż 150 koni mechanicznych. Operację tę możemy wykonać w dwóch krokach:

1. Zastanawiamy się jak wydobyć z naszej ramki danych tylko te wiersze, w których wartość hp jest większa niż 150.
2. Zastanawiamy się, jak stworzyć ramkę w której będą tylko te dwie kolumny.

Do pierwszego zadania wykorzystamy sposób analogiczny do tego poznanego przy macierzach, drugie zrobimy sposobem poznanym przed chwilą.

Stwórzmy nową ramkę danych, w której będą tylko samochody mające więcej niż 150 koni mechanicznych.

```
mt_cars_150hp = mtcars[mtcars[['hp']] > 150,] # Krok 1 - filtrujemy
```

Następnie z tej nowoutworzonej ramki danych wybierzmy dwie kolumny - `mpg` (*miles per galon* czyli liczba mil, które można przejechać na jednym galonie benzyny) oraz `hp`.

```
mt_cars_150hp[c('mpg','hp')] # Krok 2 - wybieramy
```

```
##                      mpg  hp
## Hornet Sportabout   18.7 175
## Duster 360          14.3 245
## Merc 450SE          16.4 180
## Merc 450SL          17.3 180
## Merc 450SLC         15.2 180
## Cadillac Fleetwood  10.4 205
## Lincoln Continental 10.4 215
## Chrysler Imperial   14.7 230
## Camaro Z28          13.3 245
## Pontiac Firebird    19.2 175
## Ford Pantera L      15.8 264
## Ferrari Dino        19.7 175
## Maserati Bora       15.0 335
```

Obie operacje możemy również wykonać w jednym kroku.

```
mtcars[mtcars[['hp']] > 150, c('mpg','hp')]
```

```
##                      mpg  hp
## Hornet Sportabout   18.7 175
## Duster 360          14.3 245
## Merc 450SE          16.4 180
## Merc 450SL          17.3 180
## Merc 450SLC         15.2 180
## Cadillac Fleetwood  10.4 205
## Lincoln Continental 10.4 215
## Chrysler Imperial   14.7 230
## Camaro Z28          13.3 245
## Pontiac Firebird    19.2 175
## Ford Pantera L      15.8 264
## Ferrari Dino        19.7 175
## Maserati Bora       15.0 335
```

Do zawartości kolumn możemy dostać się także za pomocą znaku dolara (`$`) oraz ich nazw. Sposób ten jest równoważny z wybieraniem wartości za pomocą podwójnego nawiasu kwadratowego (`mtcars[["mpg"]]`). Rezultatem tej operacji jest nie ramka danych (jak w przypadku `mtcars["mpg"]`), lecz wektor.

```
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

Zobaczmy, jakie wartości w kolumnie 'mpg' występują dla samochodów, które mają powyżej 150 koni mechanicznych. Użyjemy do tego dokładnie takiej samej techniki jak w przypadku macierzy, zastępujemy jednak liczbę określającą kolumnę jej nazwą.

```
mtcars[mtcars[['hp']] > 150,'mpg']
```

```
##  [1] 18.7 14.3 16.4 17.3 15.2 10.4 10.4 14.7 13.3 19.2 15.8 19.7 15.0
```

Załóżmy, że że chcemy się dowiedzieć, jaka jest najwyższa wartość w tej kolumnie. Użyjemy więc funkcji `max`, która zwraca największą wartość w danym wektorze.

```
max(mtcars[mtcars[['hp']] > 150,'mpg']) # analogicznie moglibyśmy użyć funkcji min
```

```
## [1] 19.7
```

Nie wiemy jednak co to za samochód! Oczywiście, moglibyśmy po prostu zajrzeć do naszej oryginalnej ramki danych, ale chcielibyśmy, żeby R zrobił to za nas. Wiemy, jaka jest największa wartość w kolumnie `mpg` dla samochodów, które mają więcej niż 150 koni mechanicznych - jest to 19.7. W takim razie możemy wybrać z naszej ramki danych tylko ten wiesz, w którym wartość w kolumnie `mpg` jest równa właśnie 19.7.

```
mtcars[mtcars$mpg == max(mtcars[which(mtcars[['hp']] > 150),'mpg']), ]
```

```
##               mpg cyl disp  hp drat   wt qsec vs am gear carb
## Ferrari Dino 19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
```

Spróbujmy rozbić to na kilka mniejszych kroków.

Krok pierwszy: Wybranie wartości z kolumny `mpg` dla samochodów, które mają więcej niż 150 koni mechanicznych:

```
mpg_for_over_150hp <- mtcars[which(mtcars[['hp']] > 150),'mpg']
mpg_for_over_150hp
```

```
##  [1] 18.7 14.3 16.4 17.3 15.2 10.4 10.4 14.7 13.3 19.2 15.8 19.7 15.0
```

Krok drugi: ustalenie, jaka jest najwyższa wartość w tym wektorze:

```
max_mpg_for_over_150hp <- max(mpg_for_over_150hp)
max_mpg_for_over_150hp
```

```
## [1] 19.7
```

Krok trzeci: sprawdzenie, dla których wierszy w ramce danych `mtcars` wartość w kolumnie `mpg` wynosi `max_mpg_for_over_150hp`:

```
which_max_mpg_for_over_150hp <- mtcars$mpg == max_mpg_for_over_150hp
which_max_mpg_for_over_150hp
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

Krok czwarty: użycie wynikowego wektora typu `logical` do wybrania interesującego nas wiersza:

```
mtcars[which_max_mpg_for_over_150hp, ]
```

```
##               mpg cyl disp  hp drat   wt qsec vs am gear carb
## Ferrari Dino 19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
```

Wszystko to możemy (jeżeli chcemy) zrobić w jednym kroku, ale kod może być nieczytelny - szczególnie dla początkujących programistów w R!

```
mtcars[mtcars$mpg == max(mtcars[which(mtcars[['hp']] > 150),'mpg']), ]
```

```
##               mpg cyl disp  hp drat   wt qsec vs am gear carb
## Ferrari Dino 19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
```

Zauważmy, że w ten sposób możemy konstruować bardzo skomplikowane reguły wybierania wartości z ramek danych. Możemy używać dowolnych operatorów arytmetycznych oraz logicznych, dzięki czemu możemy tworzyć złożone warunki. Czasami jednak (jak w tym przykładzie) warto rozbić kod na kilka kroków.