# Eksploracja danych - dalsze informacje o języku R

## Ćw. 1

### Testowanie przykładowego zbioru danych.

Wczytaj zbiór airquality z datasets. Poznaj strukturę zbioru, jego rozmiar i podstawowe statystyki. Przetestuj funkcje head i tail.

```
airquality
str(airquality)
airquality[100:105, 5:6]

dim(airquality)
dim(airquality[100:105, 5:6])
dim(airquality[100:105,])
dim(airquality[,5:6])
dim(airquality[5:6])
dim(airquality[,])

head(airquality[5:6])
head(airquality[,5:6])
head(airquality[5:6,])

tail(airquality[1])
tail(airquality[[1]])
tail(airquality[[1]],n=-1)
```

Zmienne mogą być określane przez numer pozycji, jak wyżej, lub nazwą zmiennej. Wykorzystany obiekt vars służy do wyboru kolumn.

```
vars <- c("Wind", "Temp")
airquality[100:105, vars]
```

## Ćw. 2

### Wczytywanie danych z pliku tekstowego

Aby wczytać plik ręcznie, musimy wiedzieć, gdzie się on znajduje na dysku:

```
system.file("csv", "weather.csv", package="rattle")
```

Możemy podejrzeć zawartośc tego pliku za pomocą

```
fn <- system.file("csv", "weather.csv", package="rattle")
file.show(fn)
```

Można również plik wczytać z internetu:

```
ds <- read.csv("http://rattle.togaware.com/weather.csv")
```

## Ćw. 3

### Wczytywanie danych z pliku tekstowego cd.

Dane zapisane w postaci pliku tekstowego importuje się za pomocą funkcji z rodziny read.
Zasadnicze parametry:

- **nazwa pliku** — jeśli plik znajduje się w katalogu roboczym (sprawdzamy getwd()), wystarczy podać nazwę pliku wraz z rozszerzeniem. Jeśli dane znajdują się poza katalogiem roboczym, należy podać pełną ścieżkę do katalogu pliku lub adres url.
- **separator kolumn** — np. sep="," lub sep=";" lub sep="\t".
- **nagłówek** - gdy plik ma nagłówek, wówczas header=TRUE, w przeciwnym razie header=FALSE.

```
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")
```

Wczytane zbiory można wykorzystywać do analizy:

```
sum(t[, "Survived"])
```

# Ćw. 4

## Wczytywanie danych ze schowka

Otwieramy arkusz kalkulacyjny z jakimiś danymi, np. contact-lenses1.csv. Zaznaczamy część z danymi nagłówkowymi i kopiujemy je, a następnie

```
schowek<-read.table(file("clipboard"), header=TRUE,sep="\t")
```

Co należy poprawić?

# Ćw. 5

## read.table

Przykłady pochodzą z książki Zuur, Alain F, Elena N Ieno, i Erik H. W. G Meesters. **A Beginner's Guide to R**. Dordrecht; New York: Springer, 2009. Dane można pobrać ze strony książki Highland Statistics Ltd. Naukowcy w 2007 przeanalizowali dane z badań prowadzonych w dwóch parkach, Yellowstone Bison Range (USA). Celem badania było ustalenie, czy bioróżnorodność tych parków zmieniła się w czasie, a jeśli tak, to czy zmiana różnorodności biologicznej związana była ze szczególnymi czynnikami środowiskowymi. Dla naszych celów użyjemy danych z Yellowstone. Do oszacowania różnorodności biologicznej, naukowcy obliczyli liczność gatunków, definiowaną jako liczbę różnych gatunków w określonym miejscu. W badaniu zidentyfikowano około 90 gatunków. Dane mierzono 8 transektach, z których każdy transek oceniano w odstępach 4-10 lat, co dało w sumie 58 obserwacji. Wczytujemy ten zbiór:

```
Veg <- read.table(file="Vegetation2.txt",header= TRUE)
names(Veg)
str(Veg)
```

# Ćw. 6

## tapply

Poniższy kod oblicza średnie bogactwo.

```
m <- mean(Veg$R)
```

Poniższy kod oblicza średnie bogactwo dla każdego transektu:

```
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])
c(m, m1, m2, m3, m4, m5, m6, m7, m8)
```

Jest to dość kłopotliwe, aby wpisać osiem poleceń do obliczenia wartości średniej dla każdego transektu. Funkcja R tapply() wykonuje takie same operacje jak kod powyżej, ale z jednej linii kodu:

```
tapply(Veg$R, Veg$Transect, mean)
```

Możemy również użyć składni

```
tapply(X = Veg$R, INDEX = Veg$Transect, FUN = mean)
```

Funkcja tapply() rozdziela dane z pierwszej zmiennej (R), w oparciu o poziomy drugiej zmiennej (Transect). Do każdej podgrupy danych stosuje funkcję, w tym przypadku średnią, ale możemy też użyć odchylenia standardowe (funkcja sd), wariancji (funkcja var), długości (length), itp. Następujące linie kodu obliczają niektóre z tych funkcji na danych:

```
Me <- tapply(Veg$R, Veg$Transect, mean)
Sd <- tapply(Veg$R, Veg$Transect, sd)
Le <- tapply(Veg$R, Veg$Transect, length)
cbind(Me, Sd, Le)
```

Do obliczenia średniej, minimum, maksimum, odchylenia standardowego, długości wszystkich danych, nadal trzeba używać mean(Veg$R), min(Veg$R), max (Veg$R), sd (Veg$R) i długość (Veg$R). Jest to pracochłonne, jeśli chcemy obliczyć średnia z wielu zmiennych, takich jak wszystkie zmienne liczbowe danych o wegetacji. Mamy 20 zmiennych numerycznych w zbiorze danych o roślinności, w kolumnach 5-25 ramki danych Veg. Jednak nie trzeba wpisywać polecenia 20 razy.

# Ćw. 7

## lapply i sapply

R oferuje inne funkcje podobne do tapply do rozwiązania tej sytuacji: lapply() i sapply(). Zastosowanie sapply i jej wyjście znajduje się poniżej:

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

Aby zaoszczędzić miejsce, przedstawiamy wyniki dla pierwszych pięciu zmiennych. Jest ważne, aby uświadomić sobie, że tapply oblicza średnią (lub dowolną inną funkcję) dla podzbiorów obserwacji zmiennej, natomiast lapply i sapply obliczają średnią (lub inne funkcje) jednej lub większej liczby zmiennych, dla wszystkich obserwacji. Słowo FUN oznacza funkcję i musi być napisane wielkimi literami. Zamiast średniej, można używać innych funkcji jako argumentu. Jaka jest różnica między sapply i lapply? Główne różnice leżą w prezentacji końcowego wyniku, jak widać w poniższym przykładzie.

```
lapply(Veg[, 5:9], FUN= mean)
```

Wyjście lapply przedstawione jest w postaci listy, a sapply daje wektor.

# Ćw. 8

## Praca domowa

1. Utworz listę **temp** złozoną z siedmiu elementów. Każdy element jest wektorem złożonym z 5 losowych liczb rzeczywistych (postaraj się użyć funkcji lapply). Do listy **temp** dołącz nazwy dni tygodnia. Dla listy **temp**
    1. Wygeneruj wektor z minimalnymi temperaturami każdego dnia.

    2. Wygeneruj wektor z maksymalnymi temperaturami każdego dnia.
2. Wczytaj zbiór USArrests.csv, następnie podaj wektor, którego elementy to sumy przestępstw każdego rodzaju.