WSTĘP DO ANALIZY DANYCH

Lab. 1: Wprowadzenie

v. 1.0.0

PLAN LAB. 1

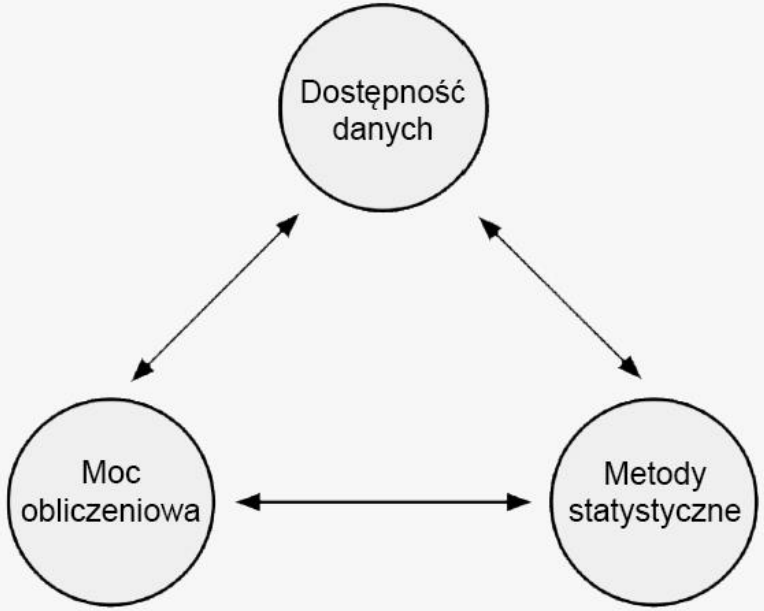
1. Sprawy organizacyjne, zasady.
2. Cel i zakres przedmiotu.
3. Podstawowe pojęcia i zagadnienia.
4. Przypomnienie/wstęp do R.

CEL I ZAKRES PRZEDMIOTU

- **Znajomość tematyki Data Science —** uczestnik zapoznany zostanie z najważniejszymi zagadnieniami interdyscyplinarnej dziedziny Data Science.
- **Umiejętności praktyczne** — uczestnik nauczy się przeprowadzać wstępną analizę surowych danych celem ocenienia ich potencjału w kontekście budowy modeli analitycznych dla różnorodnych problemów klasyfikacyjnych i regresyjnych.
- **Solidne podstawy** — przedmiot nauczy najważniejszych intuicji oraz dostarczy solidnych podstaw do dalszego rozszerzania umiejętności w zakresie analizy danych z wykorzystaniem zaawansowanych technik statystycznych i uczenia maszynowego.
- Możliwość sprawdzenia, czy Data Science jest dla mnie.

PODSTAWOWE POJĘCIA

- **Uczenie maszynowe** (Machine Learning, ML) — dziedzina, która dostarcza narzędzi w postaci algorytmów komputerowych umożliwiających przekształcenie danych w praktyczną wiedzę.
- Bez uczenia maszynowego byłoby praktycznie niemożliwym nadążyć za ogromną ilością informacji dostępnych we współczesnym świecie (tzw. era Big Data — era rejestrowania niekończącej się ilości danych; dane zbierane są na każdym kroku: czujniki, kamery, monitoring zachowań w systemach informatycznych).



Proces rozwoju uczenia maszynowego.

- **Eksploracja danych** (Data Mining) — pojęcie ściśle związane z uczeniem maszynowym (nie określa się precyzyjnie, w jakim dokładnie zakresie te dwa pojęcia się pokrywają). Przez eksplorację danych na ogół rozumie się proces analityczny wykonywany celem analizy dużych zbiorów danych poprzez **znajdowanie w nich regularnych wzorców i zależności**. W uczeniu maszynowym nacisk kładziony jest na proces „uczenia komputera", jak wykorzystać dane do rozwiązywania konkretnego zadania.

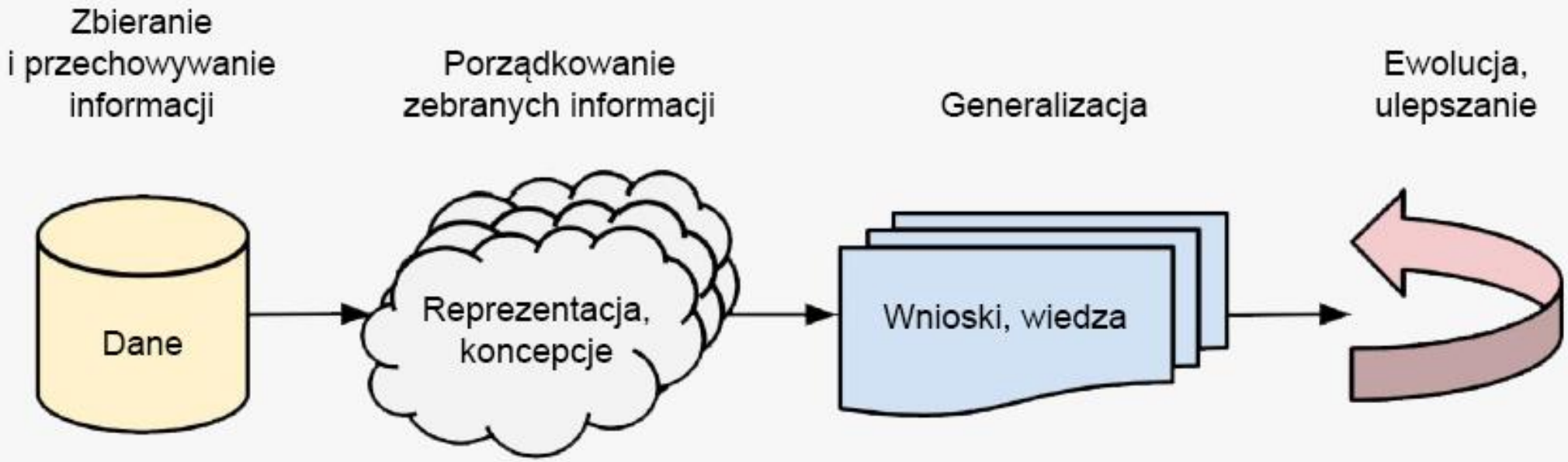
- Można powiedzieć, że do każdej eksploracji danych wykorzystuje się uczenie maszynowe, ale nie każde uczenie maszynowe wymaga eksploracji danych. Np. w przypadku algorytmu kierującego autonomicznym samochodem mówimy zwykle o czystym uczeniu maszynowym, chociaż granica oczywiście jest płynna.

- **Data Science**: dane — uczenie maszynowe — człowiek programujący proces uczenia się maszyn. Dziedzina dotycząca zarówno statystyki i programowania, jak i infrastruktury technologicznej umożliwiającej uczenie maszynowe i jego stosowanie.

- **Data Scientist** musi potrafić balansować poziom śmiałości w wykorzystaniu danych do wnioskowania na ich podstawie z ograniczeniami metod statystycznych i uczenia maszynowego. Dlatego aby być dobrym Data Scientistem, trzeba dobrze rozumieć, w jaki sposób działają metody uczenia maszynowego i statystycznej analizy danych.

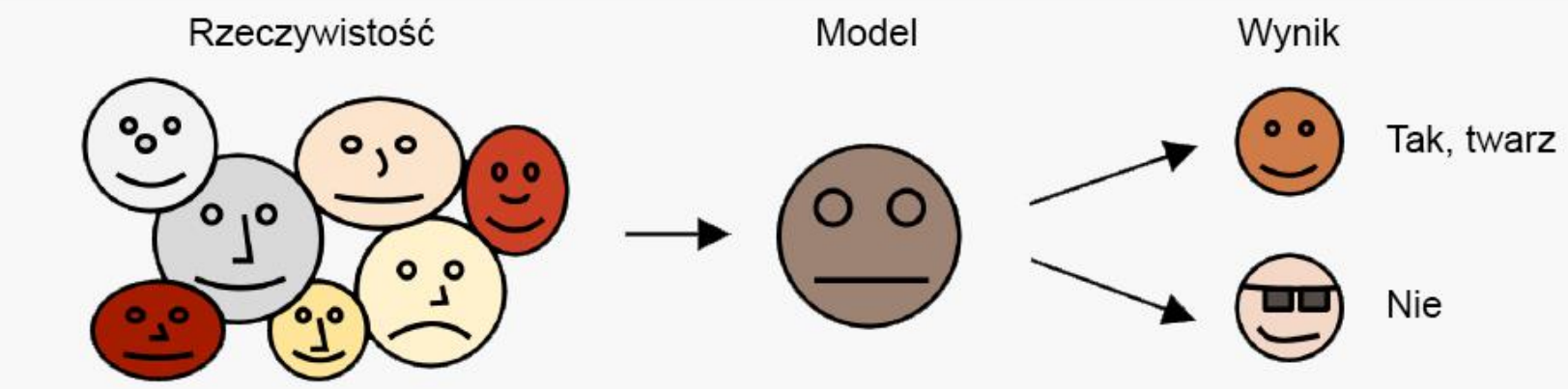
PROCES UCZENIA SIĘ

(maszyn oraz ludzi)



GENERALIZACJA - UWAGI

- Proces uogólniania podczas trenowania modelu może prowadzić do pojawienia się obciążenia (ang. *bias*; regularnego błędu).
- Algorytm nauczył się identyfikować twarz po dwóch ciemnych obiektach w kształcie kół reprezentujących oczy, umiejscowionych nad prostą linią reprezentującą usta.
- Program może mieć wówczas problem z prawidłową identyfikacją twarzy, które nie pasują do modelu (twarzy w okularach, twarzy obróconych o pewien kąt, twarzy sfotografowanych z boku itp.). Podobnie może być „obciążony" w kierunku na przykład twarzy o konkretnym odcieniu skóry, kształcie itp.



ULEPSZANIE MODELU - UWAGI

- **Dane treningowe**, **dane testowe**.
- Modele rzadko uogólniają się bezbłędnie na każdy nowy przypadek (nie są w stanie się bezbłędnie uogólnić np. z powodu szumu, który opisuje niewyjaśnione lub niewyjaśnialne różnice w danych).
- Przykładowe powody pojawiania się szumu w danych:

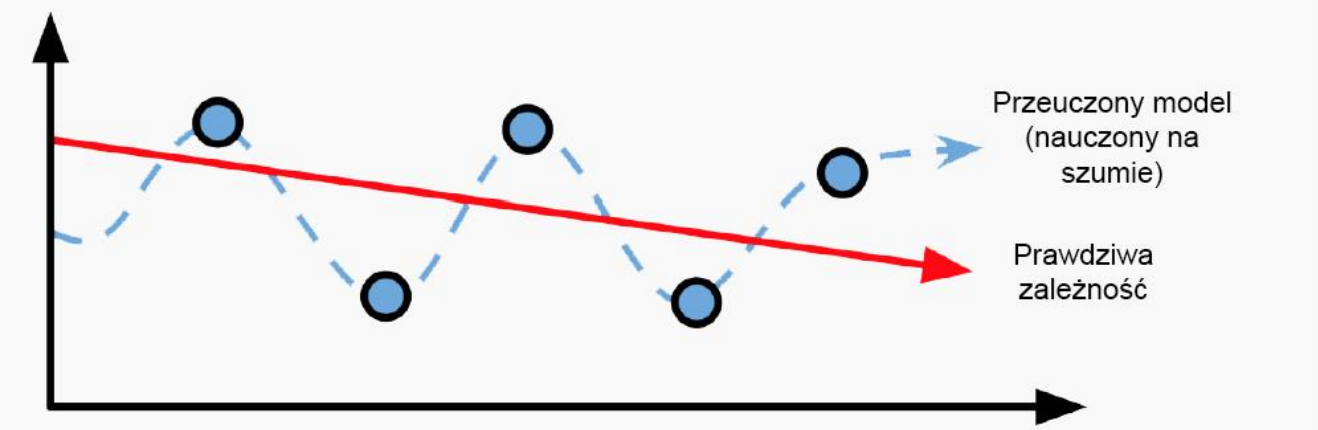
  błąd pomiaru spowodowany przez nieprecyzyjny sprzęt (np. czujnik),

  różnego rodzaju problemy z respondentami ankiet (np. zaznaczanie losowych odpowiedzi celem szybkiego wypełnienia ankiety),

  problemy z jakością danych (braki, wartości zaokrąglone/obcięte, niepoprawnie zakodowane/uszkodzone).

ULEPSZANIE MODELU - PRZEUCZENIE

- Próby modelowania szumu prowadzą do problemu **nadmiernego dopasowania modelu do danych** (ang. overfitting; **przeuczenie**).
- Przeuczony model zwykle ma bardzo dobre wyniki na zbiorze treningowym (jest nadmiernie do niego dopasowany), a słabe wyniki na zbiorze testowym. Z praktycznego punktu widzenia oznacza to, że został zidentyfikowany wzorzec w danych, który jest bezużyteczny w kontekście wnioskowania na jego podstawie dla nowych danych.



- Przykład: drzewo decyzyjne, które można położyć na danych z dokładnością 100%.

UCZENIE MASZYNOWE W PRAKTYCE

Krok 1. **Zbieranie danych**. (Połączenie ich w jedno źródło, np. plik tekstowy, baza danych).

Krok 2. **Eksploracja i przygotowanie danych**: zapoznanie się z danymi i ich charakterystyką, zrobienie „porządków".

Krok 3. **Budowa modelu**. Metody uczenia maszynowego dobieramy na podstawie charakteru postawionego do rozwiązania zadania oraz na podstawie wniosków z przeglądu danych wykonanego w poprzednim kroku.

Krok 4. **Ocena modelu** — wykonywana w różny sposób w zależności od stosowanych metod uczenia maszynowego oraz od charakteru rozwiązywanego zadania.

Krok 5. **Dopracowanie modelu**. Może okazać się, że uzyskane wyniki nie są zadowalające. Należy wówczas pomyśleć nad wykorzystaniem bardziej zaawansowanych strategii celem poprawy modelu: czasem może okazać się koniecznym przejście na inny typ modelu/inną metodę uczenia maszynowego, czasem uzupełnienie danych o dodatkowe dane, a czasami wystarczy wykonać dodatkowe prace przygotowawcze (krok 2.).

- Jeśli po tych krokach model wydaje się funkcjonować dobrze, można rozważyć jego wdrożenie.

TYPY DANYCH WEJŚCIOWYCH

- **Jednostka obserwacji** — najmniejsza jednostka, dla której mierzone są jej „właściwości" interesujące z punktu widzenia analizy (osoby, obiekty, rzeczy, transakcje, punkt czasowy, region geograficzny), np. wiadomość e-mail (model uczenia maszynowego do identyfikacji spamu).
- **Przykłady** (ang. *examples*) — kolejne jednostki obserwacji, dla których zmierzone/zebrane zostały ich „właściwości", np. konkretna wiadomość e-mail.
- **Cechy** (ang. *features*) — zmierzone/zebrane „właściwości" przykładów, np. słowa użyte w treści maila.
- Dane, które chcemy analizować z wykorzystaniem komputera, na ogół muszą być odpowiednio **ustrukturyzowane** (każdy przykład ma te same cechy, które mają zrozumiały dla komputera format).
- Podstawowe typy zmiennych (cech): **liczbowe** (ang. *numeric*), **jakościowe** (ang. *categorical*). Zmienne jakościowe mogą być **nominalne** (nieuporządkowany zbiór wartości) lub **o wartościach uporządkowanych**.

TYPY DANYCH WEJŚCIOWYCH

cechy

year	model	price	mileage	color	transmission
2011	SEL	21992	7413	Yellow	AUTO
2011	SEL	20995	10926	Gray	AUTO
2011	SEL	19995	7351	Silver	AUTO
2011	SEL	17809	11613	Gray	AUTO
2012	SE	17500	8367	White	MANUAL
2010	SEL	17495	25125	Silver	AUTO
2011	SEL	17000	27393	Blue	AUTO
2010	SEL	16995	21026	Silver	AUTO
2011	SES	16995	32655	Silver	AUTO

przykłady

Prosty zbiór danych w postaci macierzy – najpopularniejsza forma używana w uczeniu maszynowym.

TYPY METOD UCZENIA MASZYNOWEGO

- **Model predykcyjny** (ang. *predictive model*) — przewidywanie wartości wybranej zmiennej na podstawie wartości innych zmiennych. Uwaga: nie musi chodzić o przewidywanie zdarzeń w przyszłości.
- **Uczenie nadzorowane** (ang. *supervised learning*) — na podstawie przykładów.
- **Klasyfikacja** — przewidywanie kategorii przykładów (czy e-mail jest spamem, czy dana osoba ma raka, czy dana drużyna piłkarska wygra mecz, czy wnioskodawca o kredyt spłaci go). Mówimy o **klasach** lub o **poziomach** przewidywanej zmiennej. Bardzo popularne zadanie uczenia maszynowego.
- Uczenie nadzorowane może być również wykorzystywane do przewidywania wartości liczbowych (dochód, wyniki testu, ilość czegoś). Mówi się wówczas o **regresji** lub przewidywaniu numerycznym. Najpopularniejszą metodą dla tego typu zadań jest regresja liniowa (ale są też inne).

TYPY METOD UCZENIA MASZYNOWEGO

- **Uczenie nienadzorowane** (ang. *unsupervised learning*) — szukanie w zbiorze danych zależności bez wcześniej istniejących etykiet. Mówi się o modelach deskrypcyjnych. Wykorzystywane do eksploracji danych. Przykład: wykrywanie asocjacji wśród danych o zakupach (analiza koszykowa).
- **Klasteryzacja** (grupowanie, analiza skupień) — grupowanie danych (przykładów) we względnie jednorodne klasy. Przykład: identyfikacja grup klientów „o podobnych cechach", aby później kierować do nich spersonalizowane (dla danej grupy) kampanie reklamowe.
- **Meta-uczenie** (ang. *meta-learning*, *learning to learn*) — dział uczenia maszynowego zajmujący się usprawnianiem procesu uczenia w klasycznym uczeniu maszynowym; zespoły modeli (ang. *ensembles*), uczenie ze wzmacnianiem (ang. *reinforcement learning*).

ZNANE PRZYKŁADY ZASTOSOWAŃ

- Identyfikacja spamu wśród maili,
- marketing (jakie reklamy wyświetlać),
- prognoza pogody i długoterminowych zmian klimatycznych,
- redukcja nieuczciwych transakcji kartami kredytowymi,
- przewidywanie wyników wyborów,
- samochody i drony autonomiczne,
- drukowanie kuponów promocyjnych na podstawie historii zakupów klienta,
- odkrycie sekwencji genetycznych powiązanych z chorobami,
- przewidywanie zapotrzebowania na ciepłą wodę na danym obszarze,
- przewidywanie awarii,
- optymalizacja stanu magazynowego,
- optymalizacja procesów produkcyjnych,
- poza tym nieskończona liczba możliwych zastosowań w szukaniu wszelkich zależności w danych i wykorzystywaniu tych zależności.

- **„Uczenie maszynowe może być co najwyżej tak dobre, jak dane wykorzystane do uczenia”**.

DLACZEGO R DO ANALIZY DANYCH

- Jeden z najpopularniejszych języków w Data Science (obok Pythona). Python — język dla programistów, R — język dla statystyków. W kontekście Data Science w znacznym stopniu ich możliwości się pokrywają.
- Popularny w świecie nauki (ze względu na szeroki wybór bibliotek statystycznych).
- Posiada jedne z najlepszych bibliotek do wizualizacji danych.
- Stosunkowo łatwy do nauczenia się.

- Przykłady ze skryptów do przedmiotu **Wstęp do analizy danych** zostały napisane i przetestowane w R wersji 4.1.2 w systemie Microsoft Windows, ale najprawdopodobniej będą działać także na nieco starszych i na nowszych wersjach.

INSTALACJA PAKIETÓW W R

- Pakiet **RWeka** daje dostęp do obszernego zbioru algorytmów do analizy danych z pakietu **Weka** napisanego w Javie (więcej informacji nt. pakietu **Weka**: `https://www.cs.waikato.ac.nz/~ml/weka/`).
- Aby móc używać pakietu **RWeka**, trzeba mieć zainstalowane oprogramowanie Java (dostępne za darmo: `http://www.java.com`).
- Najbardziej bezpośrednim sposobem na zainstalowanie pakietu w R jest użycie funkcji `install.packages()`:

```
> install.packages("RWeka")
```

- Przy pierwszej instalacji pakietu R może poprosić o wybranie serwera lustrzanego CRAN. Najlepiej wybrać najbliższą lokalizację (aby prędkość pobierania była możliwie najwyższa).

INSTALACJA PAKIETÓW W R

- Funkcja `install.packages()` umożliwia także m.in. instalację pakietów z lokalnego pliku. Więcej informacji w pliku pomocy, który otwieramy za pomocą polecenia:

```
> ?install.packages
```

- Tym sposobem można otworzyć plik pomocy dla dowolnej funkcji w R.
- Aby załadować pakiet **RWeka** używamy:

```
> library(RWeka)
```

- Usuwanie pakietu **RWeka**:

```
> detach("package:RWeka", unload = TRUE)
```

PODSTAWOWE STRUKTURY DANYCH W R

- Struktury danych w R najczęściej używane w analizie danych:

  wektory (vectors),

  czynniki (factors),

  listy (lists),

  tablice (arrays),

  macierze (matrices)

  oraz **ramki danych** (data frames).

PODSTAWOWE STRUKTURY DANYCH W R

- **Wektor** — podstawowa struktuura danych w R, uporządkowany zbiór wartości nazywanych elementami/składowymi (wszystkie składowe jednego wektora muszą być tego samego typu).

Zadanie 1. Przeanalizować następujący kod.

```
> subject_name <- c("John Doe", "Jane Doe", "Steve Graves")
> temperature <- c(98.1, 98.6, 101.4)
> flu_status <- c(FALSE, FALSE, TRUE)
> temperature[2]
[1] 98.6
> temperature[2:3]
[1]  98.6 101.4
> temperature[-2]
[1]  98.1 101.4
> temperature[c(TRUE, TRUE, FALSE)]
[1] 98.1 98.6
```

PODSTAWOWE STRUKTURY DANYCH W R

- **Czynnik** — służy do przechowywania danych jakościowych o małej liczbie kategorii.

Zadanie 2. Przeanalizować następujący kod.

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
> blood <- factor(c("O", "AB", "A"),
+                 levels = c("A", "B", "AB", "O"))
> blood
[1] O  AB A
Levels: A B AB O
```

PODSTAWOWE STRUKTURY DANYCH W R

```
> symptoms <- factor(c("SEVERE", "MILD", "MODERATE"),
+                    levels = c("MILD", "MODERATE", "SEVERE"),
+                    ordered = TRUE)
> symptoms
[1] SEVERE   MILD     MODERATE
Levels: MILD < MODERATE < SEVERE
> symptoms > "MODERATE"
[1]  TRUE FALSE FALSE
```

- **Listy** to ciągi złożone z elementów o dowolnych typach (w odróżnieniu od wektora, który wymaga, aby wszystkie jego elementy miały ten sam typ).

PODSTAWOWE STRUKTURY DANYCH W R

Zadanie 3. Przeanalizować następujący kod (chcemy wypisać wszystkie dane dla pierwszego pacjenta z utworzonego zbioru danych — robimy to bez listy oraz z listą).

```
> subject_name[1]
[1] "John Doe"
> temperature[1]
[1] 98.1
> flu_status[1]
[1] FALSE
> gender[1]
[1] MALE
Levels: FEMALE MALE
> blood[1]
[1] O
Levels: A B AB O
> symptoms[1]
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

```
> subject1 <- list(fullname = subject_name[1],
+                  temperature = temperature[1],
+                  flu_status = flu_status[1],
+                  gender = gender[1],
+                  blood = blood[1],
+                  symptoms = symptoms[1])
> subject1
$fullname
[1] "John Doe"

$temperature
[1] 98.1

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: FEMALE MALE

$blood
[1] O
Levels: A B AB O

$symptoms
[1] SEVERE
Levels: MILD < MODERATE < SEVERE
```

```
> subject1[2]
$temperature
[1] 98.1

> subject1[[2]]
[1] 98.1
> subject1$temperature
[1] 98.1
> subject1[c("temperature", "flu_status")]
$temperature
[1] 98.1

$flu_status
[1] FALSE
```

PODSTAWOWE STRUKTURY DANYCH W R

- **Ramki danych** — najważniejsza struktura danych w kontekście analizy danych; podobna do arkusza kalkulacyjnego czy bazy danych — zawiera wiersze i kolumny. W terminach języka R ramkę danych można rozumieć jako listę wektorów lub czynników, z których każdy ma tyle samo elementów.

Zadanie 4. Przeanalizować następujący kod (tworzymy ramkę danych dla naszego zbioru dotyczącego pacjentów, łącząc utworzone wcześniej wektory; zwrócić uwagę na parametr `stringsAsFactors = FALSE`).

```
> pt_data <- data.frame(subject_name, temperature, flu_status, gender,
+                       blood, symptoms, stringsAsFactors = FALSE)
> pt_data
subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
2     Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
```

```
> pt_data$subject_name
[1] "John Doe"     "Jane Doe"     "Steve Graves"
> pt_data[c("temperature", "flu_status")]
temperature flu_status
1        98.1      FALSE
2        98.6      FALSE
3       101.4       TRUE
> pt_data[2:3]
temperature flu_status
1        98.1      FALSE
2        98.6      FALSE
3       101.4       TRUE
> pt_data[1, 2]
[1] 98.1
> pt_data[c(1, 3), c(2, 4)]
temperature gender
1        98.1   MALE
3       101.4   MALE
```

```
> pt_data[, 1]
[1] "John Doe"     "Jane Doe"     "Steve Graves"
> pt_data[1, ]
subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
> pt_data[ , ]
subject_name temperature flu_status gender blood symptoms
1     John Doe        98.1      FALSE   MALE     O   SEVERE
2     Jane Doe        98.6      FALSE FEMALE    AB     MILD
3 Steve Graves       101.4       TRUE   MALE     A MODERATE
> pt_data[c(1, 3), c("temperature", "gender")]
temperature gender
1        98.1   MALE
3       101.4   MALE
> pt_data[-2, c(-1, -3, -5, -6)]
temperature gender
1        98.1   MALE
3       101.4   MALE

> pt_data$temp_c <- (pt_data$temperature - 32) * (5 / 9)
> pt_data[c("temperature", "temp_c")]
temperature   temp_c
1        98.1 36.72222
2        98.6 37.00000
3       101.4 38.55556
```

PODSTAWOWE STRUKTURY DANYCH W R

- **Macierz** to struktura danych, która reprezentuje dwuwymiarową tabelę z wierszami i kolumnami. Macierz w R może zawierać tylko jeden typ danych (tak jak wektor). Macierze najczęściej wykorzystywane są do operacji matematycznych i zawierają same liczby.

Zadanie 5. Przeanalizować następujący kod.

```
> m <- matrix(c(1, 2, 3, 4), nrow = 2)
> m
[,1] [,2]
[1,]    1    3
[2,]    2    4
> m <- matrix(c(1, 2, 3, 4), ncol = 2)
> m
[,1] [,2]
[1,]    1    3
[2,]    2    4
```

PODSTAWOWE STRUKTURY DANYCH W R

```
> m <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
> m
[,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> m <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
> m
[,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> m[1, 1]
[1] 1
> m[3, 2]
[1] 6
> m[1, ]
[1] 1 4
> m[, 1]
[1] 1 2 3
```

- **Tablice** są natomiast wielowymiarowymi tabelami danych (mogą zawierać nie tylko wiersze i kolumny jak macierze, ale także dodatkowe warstwy danych). W ramach niniejszego przedmiotu nie będziemy potrzebowali używać tablic.

ZAPISYWANIE, ŁADOWANIE, USUWANIE W R

(podstawowe funkcje)

- Funkcja `save()` służy do zapisywania struktur danych do pliku (o rozszerzeniu **.RData**), np.:

```
> save(x, y, z, file = "mydata.RData")
```

- Natomiast aby załadować struktury danych zapisane w pliku, używamy funkcji `load()`:

```
> load("mydata.RData")
```

- Jeśli chcemy szybko zakończyć sesję w R, możemy użyć polecenia `save.image()`, które zapisuje bieżącą sesję do pliku o nazwie **.RData**. R automatycznie otworzy sesję z tego pliku przy kolejnym jego uruchomieniu.

- Funkcja `ls()` zwraca wektor nazw wszystkich zmiennych znajdujących się w pamięci. Na przykład po wykonaniu wszystkich poprzednich zadań z tego skryptu funkcja `ls()` zadziała w następujący sposób:

```
> ls()
[1] "blood" "flu_status" "gender" "m"
[5] "subject_name" "subject1" "symptoms"
[9] "temperature"
```

- R automatycznie usuwa wszystkie struktury danych po zakończeniu sesji. Jeśli chcemy zwolnić pamięć wcześniej, możemy w tym celu użyć funkcji `rm()`, np.:

```
> rm(m, subject1)
```

- Jako argument funkcji `rm()` możemy użyć także wektora nazw obiektów do usunięcia, np. jeśli chcemy wyczyścić całą sesję, wykonujemy:

```
> rm(list = ls())
```

IMPORTOWANIE I ZAPISYWANIE DO CSV

- Pliki tekstowe są popularnym sposobem przechowywania danych.
- Prawdopodobnie najpopularniejszym formatem przechowywanie danych w plikach tekstowych jest format CSV (ang. *comma-separated values*); przecinek jest najpopularniejszym separatorem danych w tym formacie.
- Plik CSV z danymi medycznymi, które zostały utworzone w trakcie poprzednich ćwiczeń w tym skrypcie, mógłby wyglądać w następujący sposób:

```
subject_name,temperature,flu_status,gender,blood_type
John Doe,98.1,FALSE,MALE,O
Jane Doe,98.6,FALSE,FEMALE,AB
Steve Graves,101.4,TRUE, MALE,A
```

- Aby załadować plik CSV o nazwie **pt_data.csv** (znajdujący się w katalogu roboczym R), możemy użyć funkcji `read.csv()`:

```
> pt_data <- read.csv("pt_data.csv", stringsAsFactors = FALSE)
```

Opcja `stringsAsFactors = FALSE` została użyta, aby zapobiec konwersji przez R wszystkich zmiennych tekstowych na czynniki (jeśli mamy pewność, że każda kolumna w pliku CSV jest rzeczywiście czynnikiem, opcja ta nie jest oczywiście potrzebna).

- Domyślnie R zakłada, że plik CSV zawiera nagłówek z nazwami kolumn znajdujących się w zbiorze danych. Jeśli nasz plik CSV nie zawiera nagłówka, należy to zaznaczyć za pomocą opcji `header = FALSE`:

```
> mydata <- read.csv("mydata.csv", stringsAsFactors = FALSE, header = FALSE)
```

- Aby zapisać ramkę danych jako plik CSV, najlepiej użyć funkcji `write.csv()`:

```
> write.csv(pt_data, file = "pt_data.csv", row.names = FALSE)
```

R domyślnie w pliku CSV wprowadza nazwy wierszy, co zwykle nie jest do niczego potrzebne — używamy wówczas opcji `row.names = FALSE)`.