UNIWERSYTET PEDAGOGICZNY
im. Komisji Edukacji Narodowej w Krakowie

INSTYTUT INFORMATYKI

Kierunek: INFORMATYKA

specjalność: Multimedia i technologie internetowe

Adrian Wlizło

Nr albumu: 137973

Zastosowanie języka proceduralnego w bazie danych w celu statystycznej analizy i wizualizacji danych

Praca inżynierska
napisana pod kierunkiem
dr hab. inż. Tomasza Hachaja

Kraków 2021

Streszczenie

Do technologii użytych w mojej pracy inżynierskiej należą: język programowania R, język proceduralny PL/R oraz system do zarzadzania relacyjno-obiektową bazą danych PostgreSQL. Ważne jest aby odpowiednio przygotować środowisko do pracy. Należy zainstalować środowisko języka R z jego klientem graficznym R-Studio. Oprócz tego instalujemy klienta PostgreSQL do zarządzania bazą danych oraz rozszerzenie PL/R dla języka proceduralnego. W mojej pracy korzystam z bazy danych wirusa COVID-19. Język proceduralny Pl/R pozwala między innymi na pisanie zwykłych funkcji jak i funkcji agregujących. Używając języka PL/R otrzymujemy dostęp do wszystkich pakietów i funkcjonalności języka R. Przy jego użyciu możemy stworzyć analizę statystyczną i wizualizację danych dostosowaną w pełni do potrzeb konsumenta. Jedną z metod odczytania wizualizacji danych jest zapis do pliku. R umożliwia zapisywanie między innymi do plików typu png lub pdf. Ważnym aspektem w analizie jest wykonanie testu rozkładu danych. Jednym z nich jest test Shapiro-Wilka. Służy on do sprawdzenia czy wybrany rozkład danych jest podobny do rozkładu normalnego zwanego inaczej rozkładem Gaussa. Analiza głównych składowych jest przydatnym rozwiązaniem w analizie i wizualizacji danych.

Abstract

The technologies used in my engineering work include: programming language R, procedural language PL/R and a system for managing relational-objective database PostgreSQL. It is important to properly prepare the environment for work. You should install R-language environment with its R-Studio graphical client. In addition, we install the PostgreSQL client for database management and the PL/R extension for procedural language. In my work I use the COVID-19 virus database. The procedural language Pl/R allows, among others, to write ordinary functions as well as aggregation functions. Using PL/R language we get access to all packages and functionalities of R language. Using it we can create statistical analysis and data visualization fully adapted to the needs of the consumer. One of the methods of reading data visualization is writing to a file. R enables saving to png or pdf files. An important aspect of the analysis is to perform a data distribution test. One of them is the Shapiro-Wolf test. It is used to check if the chosen data distribution is similar to the normal distribution called otherwise Gauss distribution. Principal Component Analysis is a useful solution in data analysis and visualization.

Spis treści

Wstęp

Język proceduralny w bazie danych można zastosować na wiele sposobów. Jednym z nich jest zastosowanie do analizy statystycznej. Co prawda istnieją aplikacje do zarządzania bazami danych, które posiadają np. tworzenie wykresów, lecz jak większość aplikacji są ograniczone dla użytkownika. Istniej rozszerzenie PL/R, które pozwala nam na skonfigurowanie bazy danych z językiem R. Dzięki temu rozszerzeniu mamy możliwość pisania funkcji w języku proceduralnym bazy danych, które wykorzystują w pełni możliwości języka R. W ten sposób można stworzyć np. analizę i wizualizację danych, która nie jest ograniczona w żaden sposób. Temat ten nie jest bardzo popularny, lecz ma bardzo duży wachlarz możliwości z uwagi na to, że obecnie żaden klient baz danych nie jest w stanie zaoferować dobrej analizy i wizualizacji danych, którą można elastycznie dostosować do zapotrzebowania.

Głównym celem pracy inżynierskiej jest stworzenie przykładowej implementacji zastosowania języka proceduralnego do przeprowadzenia analizy statystycznej i wizualizacji danych za pomocą PL/R w bazie danych. Do celów szczegółowych należy opisanie zastosowania języka proceduralnego PL/R, test rozkładu danych, rysowanie wykresów 2D oraz rysowanie wykresów wielowymiarowych zrzutowanych do 2D przy pomocy PCA.

Praca zawiera trzy rozdziały. Pierwszym jest "Środowisko pracy" w którym opisane jest oprogramowanie użyte do realizacji tematu pracy. Rozdział zawiera część teoretyczną. Drugim rozdziałem jest "Zastosowanie PL/R". Zawiera część teoretyczną dotyczącą zastosowania języka proceduralnego PL/R w bazie danych. Ostatni trzeci rozdział "Analiza statystyczna i wizualizacja danych" zawiera część teoretyczną jak i empiryczną, który opisuje analizę statystyczną, wizualizację danych, test rozkładu danych oraz analizę głównych składowych.

Rozdział 1

Środowisko pracy

Zanim przejdę do zastosowania języka proceduralnego w celu stworzenia analizy statystycznej i wizualizacji danych w niniejszym rozdziale wprowadze omówienie użytych technologii, przygotowanie środowiska pracy oraz bazy danych wykorzystanej do stworzenia analizy statystycznej i wizualizacji danych. W niniejszej pracy zostanie użyty system do zarządzania bazą danych PostgreSQL jak i jej klient pgadmin, język programowania R, język proceduralny PL/R oraz pakiet "factoextra", który zawiera w sobie funkcje PCA do języka R. Aby móc używać pakietów R innych niż domyślne, należy je najpierw zainstalować np. przy użyciu środowiska programistycznego R-Studio.

1.1 Użyte technologie

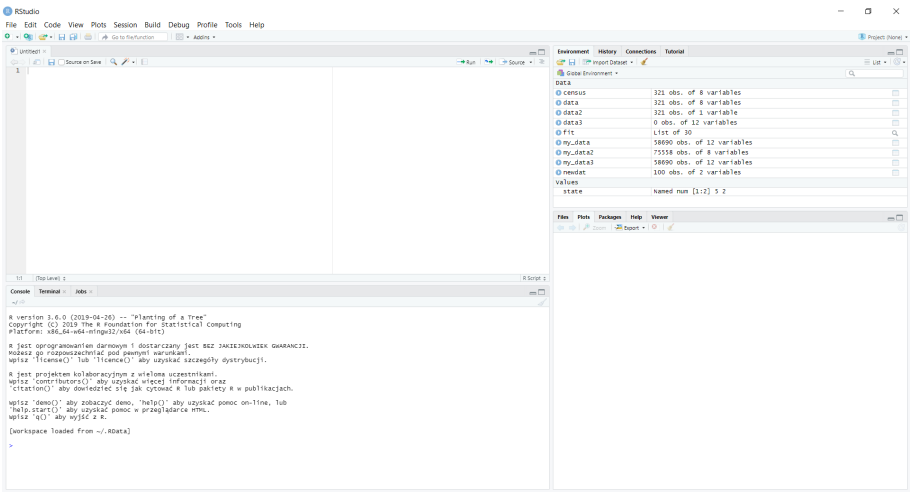
1.1.1 Język programowania R

R [1] jest językiem i środowiskiem przeznaczonym do obliczeń statystycznych oraz wizualizacji danych dostępnym jako wolne oprogramowanie[1)]. Jego twórcami są Ross Ihaka oraz Robert Gentleman. Został stworzony w 1995 roku na Uniwersytecie w Auckland w Nowej Zelandii. Jest projektem GNU podobnym do środowiska języka S, które opracowane zostało w Bell Laboratories (dawniej nosiło nazwę AT&T, natomiast obecnie Lucent Technologies) przez Johna Chambersa i jego współpracowników. Dlatego też język R można uznać za inną implementację języka S. Mimo kilku ważnych różnic, wiele kodu napisanego w języku S działa w R. Od 1997 roku rozwojem R kieruje zespół "R Core Team". Zespół ten składa się z około 17 osób, a jedną z nich jest John Chambers, który jest twórcą języka S. Oprócz tego w procesie

[1)] wolne oprogramowanie - jest to oprogramowanie, które może być uruchamiane oraz zmieniane i poprawiane przez użytkowników. Źródło: Wikipedia[28]

tworzenia bierze udział wiele użytkowników systemu oraz naukowców, którzy tworzą między innymi pakiety oraz biblioteki. Język ten zapewnia szeroką gamę technik statystycznych takich jak modelowanie liniowe i nieliniowe, testy statystyczne, analizę szeregów czasowych czy też grupowanie. Oprócz tego zapewnia również wiele technik graficznych. Jedną z mocniejszych stron R jest łatwość tworzenia dobrze zaprojektowanych wykresów o wysokiej jakości. Posiada różnież duży zasób symboli matematycznych jak i wzorów. Często inne programy do analizy danych charakteryzują bardzo specyficzne i mało elastyczne narzędzia natomiast środowisko R jest w pełni zaplanowanym i spójnym systemem. Środowisko języka R można rozszerzać za pomocą pakietów[2)]. Razem z dystybujcą R na start dostarczanych jest około ośmiu pakietów, natomiast wiele innych dostępnych jest za pośrednictwem witryn rodzinnych CRAN[3)]. Mimo że R do pewnego stopnia pozwala na stosowanie programowania funkcyjnego oraz obiektowego to jest on przede wszystkim językiem interpretowanym. W odróżnieniu od języka programowania Java, czy C/C++ kod nie jest kompilowany przy pomocy kompilatora, a jest wprowadzany do interpretera, który interpretuje go i wykonuje.

Podstawowym narzędziem jest konsola interpretera, w której wproawdzane są kolejne linie komend. Przykładowym darmowym środowiskiem programistycznym języka R jest RStudio 1.1, który pozwala na łatwiejsze przechowywanie danych wgrywanie ich.



Rysunek 1.1: Środowisko programistyczne RStudio. Źródło: Własne

[2)]pakiet - rzeczy zapakowane, paczka, plik. Źródło: Słownik języka polskiego[20]

[3)]CRAN - jest to sieć serwerów na całym świecie, które przechowują aktualne wersje dokumentacji dla R. Źródło: Strona cran.r-project.org [13]

1.1.2 Język proceduralny PL/R

PL/R [5] jest językiem proceduralnym, który umożliwia na pisanie funkcji w PostgreSQL a następnie wykonywanie ich w języku programowania R. Za jego pomocą można pisać funkcje które nie istnieją natywnie w PostgreSQL. Oferuje on większość możliwości funkcji, jakie oferuje środowisko języka R. Dostęp do bazy danych możliwy jest przy użyciu inferfejsu SPI. Jest to interfejs programowania serwera PostgreSQL. Zgłaszanie wiadomości odbywa się za pośrednictwem **elog()**. Dostęp do tworzenia funkcji w języku proceduralnym PL/R powinien mieć tylko administrator bazy danych. Niestety posiada on ograniczenie implementacyjne, które powoduje że procedur języka PL/R nie można użyć do tworzenie funkcji wejścia lub wyjścia dla nowych typów danych.

1.1.3 PostgreSQL

PostgreSQL[21] jest systemem do zarządzania obiektowo-relacyjną bazą danych. Jest to jeden z bardziej polularnych systemów. Początkowo opracowany został na University Of California w Berkeley i opublikowany pod nazwą projektu "Ingres". Pierwszą wersję zespół pod nadzorem Michaela Stonebrakera wypuścił w czerwcu 1989. Początkowo była ona udostępniona dla niewielkiej ilości użytkowników. 30 czerwca 1994 roku została wypuszczona finalna wersja systemu pod licencją typu MIT. W tym samym roku dwóch studentów z University of California w Berkeley podjęło się usprawnienia "POSTGRES". Zmienili oni jego język zapytań z POSTQUEL na SQL, który był bardziej popularny i relacyjny. Nazwali go wtedy "Postgres95". W miarę rozwoju i zwiększania funkcjonalności, system do zarządzania bazą danych otrzymał ostatecznie nazwe PostgreSQL. Finalnie PostgreSQL został wypuszczony w styczniu 1997 roku w wersji 6.0. Obecnie nad rozwojem systemu pracuje grupa "The PostgreSQL Global Development Group". Zaletami systemu do zarządzania bazami danych są:

1. Hybrydowość:
   Jest jednym z niewielu systemów zarządzania bazami danych, który oferuje najlepsze rozwiązania z modelu relacyjno-obiektowego.

2. Otwarte Oprogramowanie:
   Jest on pod licencją typu open source. Licencja ta umożliwia darmowe korzystanie jak i modyfikowanie kodu źródłowego.

3. Niezawodność:
   Usprawniany jest od ponad 20 lat. W skutek tego jest jednym z najbardziej niezwodnych i funkcjonalnych systemów do zarządzania bazą danych.

1.2 Przygotowanie środowiska do pracy

Aby korzystać z PL/R, należy w pierwszej kolejności zainstalować środowisko R-Language oraz PostgreSQL. Lista kompatybilnych ze sobą wersji przedstawiona w tablicy 1.1

Tablica 1.1: Kompatybilne wersje z biblioteką PL/R. Źródło: Własne

PL/R	Postgresql	R
8.4	11	6.3.0
8.4	10	6.3.0
8.4	9.6	6.3.0
8.4	9.5	6.3.0
8.4	9.4	6.3.0

1.2.1 R i RStudio

Należy pobrać i zainstalować środowisko języka R. Aby można było korzystać z biblioteki PL/R należy pobrać[23] starszą wersję języka R 3.6.0, która jest kompatybilna z biblioteką PL/R. Po zainstalowaniu środowiska należy uruchomić RGUI i wykonać polecenie "*update.packages*()", które zaaktualizuje pakiety środowiska R. Następnym krokiem jest instalacja środowiska programistycznego języka R jakim jest RStudio. Można je pobrać ze strony[15] twórcy.

1.2.2 Postgresql

Kolejnym krokiem jest pobranie systemu zarządzania bazami danych Postgresql. Aby możliwe było korzystanie z biblioteki PL/R należy pobrać Postgresql w wersji 10. Oprogramowanie jest dostępne na stronie[22] twórcy.

1.2.3 PL/R

Teraz przejdziemy do instalacji biblioteki PL/R. Można ją pobrać z githuba[6]. Po pobraniu oraz wypakowaniu odpowiedniego pliku dla wersji środowiska języka R oraz bazy danych, należy je skopiować do odpowiednich lokalizacji w których zainstalowany jest postgresql. Lokalizacje docelowe przedstawione są w tablicy 1.2.

Tablica 1.2: Lokalizacja docelowa plików biblioteki PL/R. Źródło: Własne

Nazwa pliku	Miejsce docelowe
plr.dll	PostgreSQL/<wersja>/lib
plr.control	PostgreSQL/<wersja>/share/extension
plr–8.3.0.18–8.4.sql	PostgreSQL/<wersja>/share/extension
plr–8.4.sql	PostgreSQL/<wersja>/share/extension
plr–unpackaged–8.4.sql	PostgreSQL/<wersja>/share/extension
plr.pdb	PostgreSQL/<wersja>/symbols
PLR_LICENSE	PostgreSQL/<wersja>/

Następnie należy ustawić zmienne środowiskowe. Aby to zrobić należy przejść do Panelu sterowania -> System i zabezpieczenia -> System -> Zaawansowane -> Zmienne środowiskowe. W zmiennych środowiskowych należy dodać lokalizacje folderów bin i lib Postrgresql oraz lokalizacje środowiska R. Nazwy oraz wartości zmiennych środowiskowych przedstawione w tablicy 1.3.

Tablica 1.3: Zmienne środowiskowe. Źródło: Własne

Nazwa zmiennej	Wartość
R_HOME	C:/Program Files/R/R-3.6.0
Path	C:/Program Files/R/R-3.6.0/bin/x64
Path	C:/Program Files/PostgreSQL/10/bin
Path	C:/Program Files/PostgreSQL/10/lib

Po ustawieniu zmiennych środowiskowych należy ponownie uruchomić usługę PostgreSQL. Można to zrobić w panelu sterowania -> Usługi. Jeżeli zmiany nie zaczęły obowiązywać konieczne może być ponowne uruchomienie komputera. Następnie należy uruchomić aplikacje pgAdmin 4, przejść do wybranaj bazy danych i w edytorze zapytań wpisać polecenie "$CREATE\ EXTENSION\ plr$;", które załaduje funkcjonalności PL/R do bazy danych. Teraz można sprawdzić czy PL/R działa

poprawnie testując działania przykładowej funkcji przedstawionej w bloku kodu 1.2 wraz z jej wywołaniem. Natomiast wynik wykonania funkcji przedstawiony w tabeli 1.4. Funkcja służy do zwracania tekstu.

```
CREATE OR REPLACE FUNCTION plr_array (text, text)
RETURNS text[] AS '
$libdir/plr','
plr_array
' LANGUAGE C WITH (isstrict);
select plr_array('hello','world');
```

Rys. 1.2: Kod funkcji plr_array. Źródło: Strona autora[4]

Tablica 1.4: Oczekiwany wynik. Źródło: Własne

	plr_array text[]
1	{hello,world}

1.3 Baza danych

1.3.1 Opis

W swojej pracy będę wykorzystywał dane dotyczące wirusa COVID-19. Baza danych pobrana została ze strony data.europa.eu[7]. Dane użyte do implementacji dostępne są w załaczniku oraz w repozytorium na platformie Github[11] Jej autorem jest Europejskie Centrum ds. Zapobiegania i Kontroli Chorób. Zbiór danych zawiera ogólnodostępne dane na temat COVID-19. Dnia 12 lutego 2020 roku nowy koronawirus został nazwany 2019-nCoV (SARS-CoV-2), a wywoływana nim choroba nazywa się obecnie COVID-19. Ostatnia aktualizacja danych w bazie użytej do implementacji pochodzi z dnia 19 października 2020 roku.

1.3.2 Załadowanie pliku do bazy danych

Po pobraniu pliku w formacie cvs zawierającego bazę danych, należy stworzyć nową tabelę zawierającą nazwy danych z pobranego pliku. Kod do stworzenia tabeli znajduje się w bloku kodu 1.3. Typy danych:

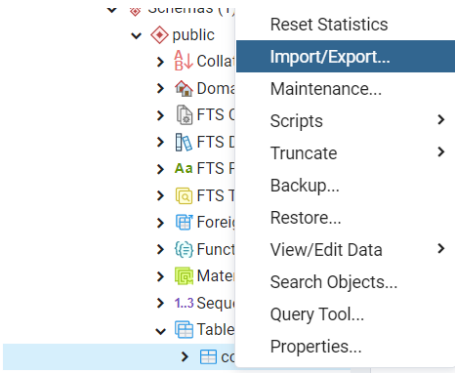
- dateRep - data

- day - dzień
- month - miesiąc
- year - rok
- cases - nowe potwierdzone przypadki
- deaths - nowe zgony
- countriesAndTerritories - kraj
- geoId - identyfikator geogreficzny
- countryterritoryCode - krajowy kod
- popData2019 - liczba populacji
- continentExp - kontynent
- Cumulative_number_for_14_days_of_COVID_19_cases_per_100000 - liczba nowych przypadków na 100000 mieszkańcow.

```
CREATE TABLE covid(
        dateRep DATE,
        day INT,
        month INT,
        year INT,
        cases INT,
        deaths INT,
        countriesAndTerritories VARCHAR (100),
        geoId VARCHAR (2),
        countryterritoryCode VARCHAR (7),
        popData2019 INT,
        continentExp VARCHAR (50),
        Cumulative_number_for_14_days_of_COVID_19_cases
        _per_100000 DOUBLE PRECISION
);
```

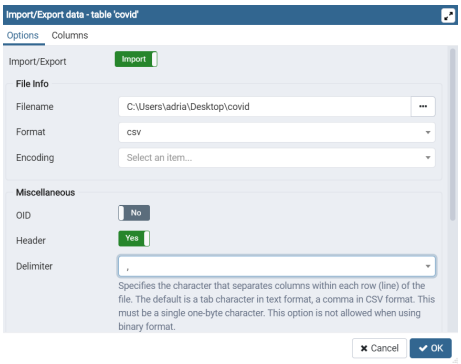
Rys. 1.3: Kod zapytania do stworzenia tabeli. Źródło: Własne

Teraz można przejść do zaimportowania danych. Należy rozwinąć zakładke ”Tables”, odszukać wcześniej utworzoną tabele, nacisnąć na nią prawym przyciskiem myszy i wybrać ”Import/Export...”. Rysunek 1.4.



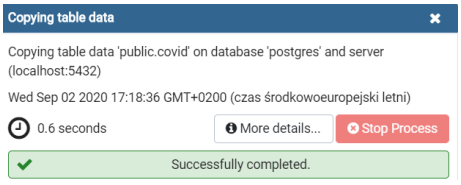
Rysunek 1.4: Lista wyboru. Źródło: Własne

Następnie należy zmienić u samej góry opcje z Export na Import oraz wybrać lokalizację pobranego wcześniej pliku bazy danych Covid-19. Ustawić format pliku na csv oraz włączyć nagłówki i wybrać odpowiedni znak oddzielający kolumny. Rysunek 1.5.



Rysunek 1.5: Okna importu. Źródło: Własne

Jeżeli wszystko wykonaliśmy dobrze, zostaniemy powiadomieni o poprawnym wykonaniu zadania oraz czasie jego wykonania. Rysunek 1.6.



Rysunek 1.6: Okno potwierdzenia wykonanego importu. Źródło: Własne

Rozdział 2

Zastosowanie PL/R

2.0.1 Funkcje i argumenty

Aby utworzyć funkcje[2] w języku PL/R należy użyć standardowej składni języka R bez nawiasów klamrowych. Ciało funkcji języka R zaprezentowane w bloku kodu 2.1, natomiast języka PL/R w bloku kodu 2.2.

```
1 mojafunkcja <- funkcja (argument) {cialo funkcji}
```

Rys. 2.1: Kod ciała funkcji w R. Źródło: Własne

```
1 CREATE OR REPLACE FUNCTION nazwa funkcji(typ argumentu)
2 RETURNS zwraca AS '
3     cialo funkcji
4 'LANGUAGE plr;
```

Rys. 2.2: Kod ciała funkcji w PL/R. Źródło: Własne

Ciało funkcji jest fragmentem skryptu R. W momencie wywoływania funkcji, wartości argumentów są przekazywane jako zmienne arg1, arg2, ..., argN. Od PostgreSQL wersji 8.0, możliwe jest jawne nazywanie argumentów podczas tworzenia funkcji. Argument jawnie nazwany będzie dostępny dla skryptu R zamiast zmiennej argN. Przykład w bloku kodu 2.3.

```
CREATE OR REPLACE FUNCTION
r_max (pierwsza integer,druga integer)
RETURNS integer AS'
    if (pierwsza > druga)
        return(pierwsza)
    else
        return(druga)
' LANGUAGE plr;
select countriesandterritories, r_max(covid.cases,
covid.deaths) from covid;
```

Rys. 2.3: Kod zwracania większej z dwóch wartości. Źródło: Własne

Po wykonaniu powyższego zapytania powinna zostać zwrócona taleba z nazwą państw oraz wyższą wartością z dwóch przekazanych do stworzonej funkcji w PL/R r_max.

2.0.2 Przekształcanie wartości danych

Wartości argumentów[3] wejściowych dostarczone do funkcji PL/R, są przekształcane na odpowiedni format w R. Obsługiwane są maksymalnie tablice trójwymiarowe. Typy argumentów przedstawione w tablicy 2.1

Tablica 2.1: Typy argumentów. Źródło: Strona autora[3]

PostgreSQL	R
boolean	logical
int2, int4	integer
int8, float4, float8, cash, numeric	numeric
bytea	object
everything else	character

Również wartości zwracane są w pierwszej kolejności przekształcane do R, więc wszystko co zostanie rozpoznane jako ciąg znaków zwróci wynik. Tablica 2.2.

Tablica 2.2: Podsumowanie wymiarowości. Źródło: Strona autora[3]

Typ PgSQL	Typ R	Wynik	Przykład
scalar	array, matrix, vector	pierwsza kolumna pierwszego rzędu	c (1,2,3) w R zwraca 1 w PostgreSQL
setof scalar	1D array, greater than 2D array, vector	wielorzędowy, 1 zestaw kolumn	array (1:10) w R zwraca 10 wierszy w PostgreSQL
scalar	data.frame	tekstowa reprezentacja wektora pierwszej kolumny	data.frame (c (1,2,3)) in R zwraca 'c (1, 2, 3)'
setof scalar	2D array, matrix, data.frame	columns > 1, błąd; columns == 1, wiele wierszy, 1 zestaw kolumn	(as.data.frame (array (1: 10, c (2,5)))) [, 1] in R zwraca 2 wiersze wartości skalarnej
array	1D array, greater than 3D array, vector	Tablica 1D	array (1: 8, c (2,2,2,2)) w R zwraca 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8
array	2D array, matrix, data.frame	Tablica 2D	array (1: 4, c (2,2)) w R zwraca 1,3, 2,4
array	3D array	Tablica 3D	array (1: 8, c (2,2,2)) w R zwraca 1,5, 3,7, 2,6, 4,8
composite	1D array, greater than 2D array, vector	pierwszy rząd, 1 kolumna	array (1: 8, c (2,2,2)) w R zwraca 1 wiersz wartości skalarnej
setof composite	1D array, greater than 2D array, vector	wielorzędowy, 1 zestaw kolumn	array (1: 8, c (2,2,2)) w R zwraca 8 wierszy wartości skalarnej
composite	2D array, matrix, data.frame	pierwszy wiersz, wielokolumnow	array (1: 4, c (2,2)) w R zwraca 1 wiersz z 2 kolumnami
setof composite	2D array, matrix, data.frame	zestaw wielokolumnowy, wielokolumnowy	array (1: 4, c (2,2)) w R zwraca 2 wiersze po 2 kolumny

Rozdział 3

Analiza statystyczna i wizualizacja danych

3.1 Test rozkładu danych

3.1.1 Test Shapiro-Wilka

Test Shapiro-Wilka[26] używany jest w statystyce do testowania czy rozkład zmiennej jest podobny do rozkładu normalnego. Za pomocą tego testu możemy sprawdzić czy rozkład naszych zmiennych jest zbliżony do rozkładu normalnego. Testy na normalność rozkładu są wymagane w momencie korzystania z testów parametrycznych takich jak testyt-Studenta lub analizy wariancji. Przeprowadzając test chcemy otrzymać wynik nie istotny statystycznie[25] ($np.\quad p > 0,05$). Jeżeli wartość obliczona jest mniejsza niż wartość krytyczna to otrzymujemy wtedy wynik istotny statystycznie, czyli nasza zmienna nie posiada rozkładu zbliżonego do rozkładu normalnego. Wzór[31] testu Shapiro-Wilka na postać:

$$W = \frac{[\sum_i a_i(n)(X_{n-i+1} - X_i)]^2}{\sum_{j=1}^{n}(X_j - \overline{X})^2} \tag{3.1}$$

Wzór można opisać następująco:
W- wynik testu
$a_i(n)$- stała $X_{n-i+1} - X_i$- różnica pomiędzy skrajnymi obserwacjami
j- kolejne obserwacje w próbie
i- kolejne różnice między skrajnymi obserwacjami
$\overline{X}$- średnia
Wartości $a_i(n)$ można obliczyć według poniższego wzoru:

$$a = [a_1, a_2, \ldots, a_n] = \frac{m'V^{-1}}{\sqrt{m'V^{-1}V^{-1}m}} \tag{3.2}$$

Aby wykonać test Shapiro-Wilka przy pomocy języka proceduralnego należy w pierszej kolejności stworzyć tablice pomocniczą w bazie danych do przechowywania wartości, które otrzymamy po wykonaniu testu. Kod zaprezentowany w bloku kodu 3.1.

```
create table shapiro(
w double precision,
p_value double precision
);
```

Rys. 3.1: Kod tworzenie tablicy shapiro. Źródło: Własne

Kiedy tablica została utworzona można przejść do napisania funkcji w PL/R. Funkcja ta przyjmuje dwa parametry na wejściu i zwraca tablice danych języka R do utworzonej tabeli o nazwie ”shapiro”. Test ten można bardzo łatwo przeprowadzić w języku R z uwagi na to, że funkcja *shapiro.test*() znajduje się w domyślnych pakietach środowiska R. Kod funkcji shapiro zaprezentowany w bloku kodu 3.2.

```
CREATE OR REPLACE FUNCTION f_shapiro(text, text)
RETURNS setof shapiro AS '
sql<-paste("select ", arg1," from covid where
countriesandterritories=' '", arg2, "' '", sep="")
rs <- pg.spi.exec(sql)
wartosc <- rs[,1]
wynik <- shapiro.test(wartosc)
ww<-wynik[1]
pp<-wynik[2]
return(data.frame(w = ww,p_value = pp))
' LANGUAGE plr;
```

Rys. 3.2: Kod funkcji summary. Źródło: Własne

Do wywołania funkcji języka proceduralnego służy zapytanie SQL przedstawione w bloku kodu 3.3.

```
SELECT * from f_shapiro('deaths','Poland');
```

Rys. 3.3: Kod wywołania funkcji f_shapiro. Źródło: Własne

Po wywołaniu funkcji zostanie wypisana tabela z wynikiem testu, czyli wartościami zwróconymi przez funkcję. Wynik został przedstawiony w tablicy 3.1.

Tablica 3.1: Wynik testu Shapiro-Wilka. Źródło: Własne

w	p
0.4722	4.8939e-027

Z obserwacji można stwierdzić, że rozkład analizowanej zmiennej odnośnie zgonów spowodowanych wirusem covid-19 jest różny od rozkładu normalnego. Wynik ten jest istotny statystycznie, czyli zmienna dotycząca zgonów nie jest rozkładem normalnym.

3.2 Analiza statystyczna

Analiza statystyczna[12] jest nauką polegającą na zbieraniu i badaniu danych. Analiza przeprowadzana jest w celu określenia podstawowych schematów oraz zebrania jak największej wiedzy z uzyskanych danych. Służy ona między innymi do wyciągania istotnych wniosków dla firm. Do podstawowych metod, które pozwalają na szybki przegląd danych należy:

- średnia arytmetyczna
- odchylenie standardowe
- regresja liniowa
- określanie rozmiaru próby danych
- kwartyle
- wariancja
- mediana
- minimum
- maksimum
- zakres próby danych
- rozstęp międzykwartylowy
- suma

3.2.1 Średnia arytmetyczna

Średnia arytmetyczna[33] jest to suma zbioru liczb podzielona przez wielkość zbioru. Wyraża się wzorem:

$$\overline{x} = \frac{[x_1 + x_2 + x_3 + \cdots + x_n}{n} \tag{3.3}$$

gdzie:

$\overline{x}$- średnia

x_n- obserwacja w próbie

n- liczba obserwacji w próbie

3.2.2 Odchylenie standardowe

Odchylenie standardowe[19] jest to miara zmienności zaobserwowanych wyników. Informuje o wielkości rozrzutu wyników wokół średniej. Wyrażony jest wzorem[29]:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(X - \overline{X})^2}{N - 1}} \tag{3.4}$$

gdzie:

SD- odchylenie standardowe

$\overline{X}$- średnia

X- obserwacja w próbie

N- liczba obserwacji w próbie

3.2.3 Regresja liniowa

Regresja liniowa[24] jest jednym z wariantów regresji w statystyce. Zakłada ona, że jest zależnością liniową pomiędzy zmienną objaśnianą a objaśniającą. Założeniem regrasji liniowej jest, że wzrostowi jednej zmiennej towarzyszy wzrost lub spadek drugiej zmiennej. Analizę regresji liniowej przeprowadza się w celu wyliczenia współczynników regresji takich, aby model przewidywał najlepszą wartość zmiennej zależnej oraz oszacowania błędu, aby był jak najmniejszy. Regrasja liniowa dla jednego predyktora wyrażona jest wzorem[30]:

$$Y = b_1 X + a \tag{3.5}$$

gdzie:

Y- przewidywana regresja liniowa

X- predyktor w modelu
b_1- współczynnik dla predyktora
a- wyraz wolny

3.2.4 Rozmiar próby danych

Rozmiar próby danych jest zliczeniem wielkości próby w danych. Jest to nic innego jak liczebność próby danych, czyli liczba danych z jakimi mamy do czynienia w wybranej próbie danych.

3.2.5 Kwartyle

Kwartyl[17] określa wartość podziału grupy badanej w zależności od założonych proporcji. W kwartylach obserwacja dzielona jest na cztery równe części.

Tablica 3.2: Podział obserwacji w kwartylach. Źródło: Własne

Kwartyl	podział w stosunku
Pierwszy (Q1)	25%-75%
Drugi (Q2)	50%-50%
Trzeci (Q3)	75%-25%
Czwarty (Q4)	100%-0%

Kwartyl pierwszy oznacza, że 25% obserwacji jest równa lub mniejsza od wartości pierwszego kwartyla, natomiast 75% jest równa lub większa. W przypadku kwartyla drugiego tak naprawdę mamy doczynienia z medianą, ponieważ obserwacja dzielona jest na w stosunku 50%-50%. Kwartyl trzeci podobnie jak pierwszy dzieli obserwacje w różnych proporcjach. Oznacza to, że w kwartylu trzecim 75% obserwacji jest równa lub mniejsza od wartości pierwszego kwartyla, natomiast 25% jest równa lub większa. Ostatni kwartyl czwarty jest rzadko podawany ponieważ jego wynik jest maksymalny, więc nie stanowi on wartościowej informacji.

3.2.6 Wariancja

Wariancja[27] jest to miara zmienności zaobserwowanych wyników. Informuje o wielkości zróżnicowania wyników w zbiorze. Wyrażona jest wzorem[32]:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(X - \overline{X})^2}{N-1}} \tag{3.6}$$

gdzie:
SD^2- odchylenie standardowe
$\overline{X}$- średnia
X- obserwacja w próbie
N- liczba obserwacji w próbie

3.2.7 Mediana

Mediana[18] jest to jedna z bardziej popularnych miar w statystyce. Nazywana jest inaczej jako wartość środkowa zbioru. Grupa obserwacji dzielona jest na dwie równe grupy, mediana przyjmuje wartość środkową. Jeżeli liczba danych w zbiorze jest parzysta, to mediana równa jest średniej arytmetycznej dwóch liczb środkowych.

3.2.8 Minimum i maksimum

Minimum jest to najmniejsza wartość w zbiorze danych, natomiast maksimum przyjmuje największą wartość ze zbioru danych.

3.2.9 Zakres próby danych

Zakres próby danych jest to wyznaczenie wartości zakresu posiadanych danych. Oblicza się ją odejmując od wartości maksymalnej wartość minimalną ze zbioru danych.

3.2.10 Rozstęp międzykwartylowy

Rozstęp międzykwartylowy jest to wartość różnicy kwartyla pierwszego i trzeciego. Oblicza się go odejmując od kwartyla 75% kwartyl 25% ze zbioru daych.

3.2.11 Suma

Suma jest to zsumowana wartość wszystkich próbek ze zbioru danych. Wynik uzyskuje się poprzez dodanie do siebie wszystkich wartości z badanego zbioru danych.

3.2.12 Implementacja funkcji agregujących

Analizę statystyczną można przeprowadzić na dwa sposoby. Pierwszym sposobem jest użycie funkcji agregujących w języku proceduralnym PL/R. Przy ich pomocy możemy zaagregować każdą z podstawowych metod analizy statystycznej w osobnej funkcji a następnie wywołać je wszystkie lub tylko wybrane w zapytaniu SQL do bazy danych. Jako przykład funkcji agregującej posłużę się funkcją do obliczenia średniej arytmetycznej. Do jej obliczenia skorzystałem z funkcji "*mean*()" Pakietu R, która służy do obliczania średniej arytmetycznej. Funkcja oraz jej agregacja przedstawiona w bloku kodu 3.4.

```
1  CREATE OR REPLACE FUNCTION r_mean(float8[])
2  RETURNS float8[] AS '
3  mean(arg1)
4  ' LANGUAGE 'plr';
5  CREATE AGGREGATE mean (
6  sfunc = plr_array_accum,
7  basetype = float8,
8  stype = float8[],
9  finalfunc = r_mean
10 );
```

Rys. 3.4: Kod funkcji agregującej średnią. Źródło: Własne

Wszystkie funkcje agregujące załączone w pliku funkcje_agregujące.sql. Do wywołania funkcji służą zapytania SQL w języku proceduralnym. Agregacja pozwala nam na wywołanie dowolnej ilości fukcji do analizy statystycznej, również w dowolnej kolejności. Przykładowe wywołanie funkcji w bloku kodu 3.5. Funkcja zwracająca wyniki analizy statystycznej nowych przypadków wirusa covid-19 dla Polski.

```
SELECT countriesandterritories as państwo,
quantile(cases) as kwartyl_25_50_75,
mean(cases) as średnia,
var(cases) as wariancja,
sd(cases) as odchylenie,
median(cases) as mediana,
min(cases) as minimum,
max(cases) as maksimum,
range(cases) as zakres,
length(cases) as liczba_danych,
iqr(cases) as rozstęp_międzykwartylowy,
sum(cases) as suma
FROM covid
WHERE countriesandterritories = 'Poland'
GROUP BY countriesandterritories;
```

Rys. 3.5: Kod wywołania funkcji agregujących. Źródło: Własne

W wyniku zapytania otrzymujemy dane zaprezentowane w tabeli 3.3.

Tablica 3.3: Wyniki wywołania funkcji agregujących. Źródło:Własne

państwo	Poland
kwartyl_25_50_75	301.5,421,810
średnia	2983.87
wariancja	41842419.36
odchylenie	6468.57
mediana	421
minimum	0
maksimum	27875
zakres	27875
liczba_danych	259
rozstęp_międzykwartylowy	508.5
suma	772823

3.2.13 Implementacja funkcji summary

Przed przystąpieniem do funkcji summary należy w pierwszej kolejności utworzyć tabele do przechowywania wyników funkcji summary. Stworzenie tabeli zaprezentowane w bloku kodu 3.6.

```
1  CREATE TABLE summarytab (
2  państwo varchar(100),
3  średnia double precision,
4  odchylenie_standardowe double precision,
5  minimum double precision,
6  maksimum double precision,
7  zakres double precision,
8  ilość_danych double precision,
9  kwartyl25 double precision,
10 kwartyl75 double precision
11 );
```

Rys. 3.6: Kod tablicy summary. Źródło: Własne

Drugim sposobem jest użycie funkcji, która zawiera w sobie wszystie obliczenia podstawowej analizy statystycznej. Funkcja została zaprezentowana w bloku kodu 3.7 jako funkcja summary, która odzwierciedla funkcję "*summary*()". Funkcja ta jest dostępna w standardowym pakiecie języka R. Funkcja ta oblicza średnią, odchylenie standardowe, minimum, maksimum, zakres danych, liczbe danych oraz kwartyl 25% i 75%.

```
CREATE OR REPLACE FUNCTION summary(text, text)
returns SETOF summarytab as '
sql<-paste("select ", arg1, "from covid where
countriesandterritories=' '", arg2, "' '", sep="")
rs <- pg.spi.exec(sql)
rng <- range(rs[,1])

quan <-quantile(rs[,1])
names(quan) <- c()

return(data.frame(panstwo = arg2, srednia =
mean(rs[,1]),odchylenie_standardowe = sd(rs[,1]),
minimum = rng[1], maksimum = rng[2], zakres =
rng[2] - rng[1], liczba_danych = length(rs[,1]),
kwartyl25 = quan[2], kwartyl75 = quan[4]))
' LANGUAGE 'plr';
```

Rys. 3.7: Kod funkcji summary. Źródło: Własne

Do wywołania funkcji służy zapytanie SQL w języku proceduralnym. W przykładzie funkcja oczekuje dwóch parametrów. Pierwszym z nich jest nazwa danych, na których wykonywana będzie analiza statystyczna, natomiast drugim parametrem jest określenie kraju, dla którego ma być wykonana funkcja summary. Wywołanie funkcji przedstawione w bloku kodu 3.8 Funkcja zwracająca te same wartości co funkcja summary w R, czyli średnią, odchylenie standardowe, minimum, maksimum, zakres, liczba danych oraz kwartyl 25% i 75%.

```
select * from summary('cases','Poland');
```

Rys. 3.8: Kod wywołania funkcji summary. Źródło: Własne

W wyniku zapytania otrzymamy dane przedstawione w tabeli 3.4.

Tablica 3.4: Wyniki wywołania funkcji summary. Źródło: Własne

państwo	Poland
średnia	2983.87
odchylenie_standardowe	6468.57
minimum	0
maksimum	27875
zakres	27875
liczba_danych	259
kwartyl_25	301.5
kwartyl_75	810

3.2.14 Wnioski

Z otrzymanych wyników możymy wyciągnąć następujące wnioski. Średnia dzienna liczba nowych zakażeń wirusem covid-19 wynosi 2983.87. Łącznie w Polsce wirusem covid-19 zaraziło się 772823 osoby. Zakres wartości jest równy 27875 Najmniejsza dzienna liczba zakażeń koronawirusem wynosiła 0, natomiast największa 27875. Liczba danych na których wykonywana była analiza statystyczna wynosi 259. Z kwartyla pierwszego, czyli podziału w stosunku od 25% do 75% wynika, że 75% przypadków była większa lub równa 301.5, natomiast z kwartyla trzeciego, czyli podziału w stosunku od 75% do 25% wynika, że 25% przypadków była większa lub równa wartości 810. Znając mediane z tabeli 3.3 wynoszącą 421 możemy wywnioskować, że największy wzrost zakażeń wirusem covid-19 występował w drugiej połowie badanej próbki. Odchylenie standardowe jest ponad dwukrotnie większe niż średnia, na skutek tego możemy zauważyć, że wartość średnia nie niesie za sobą żadnych istotnych informacji. Na podstawie tych danych możemy wywnioskować również, że rozkład zachorowań, czyli nowych przypadków nie jest normalny.

3.3 Wizualizacja danych

Wizualizacja danych służy do przedstawienia informacji w postaci graficznej. W pracy inżynierskiej skupie się na wizualizacji statycznej czyli wszelkie informacje w postaci graficznej będą zapisywane do pliku graficznego.

3.3.1 Zapisywanie rysunków

Środowisko języka R oferuje możliwość zapisania informacji w postaci graficznych do pliku. Pakiet podstawowy, który jest zainstalowany podczas instalacji środowiska umożliwia zapis między innym do pliku w formacie png oraz pdf. W implementacji wizualizacji danych skorzystałem z zapisu pliku w formacie png. Przy zastosowaniu języka proceduralnego plik graficzny zostaje zapisany w docelowym folderze serwera. Aby sprawdzić lokalizacje zapisywania plików należy wywołać zapytanie "$SHOW data\_directory$;". W wyniku tego zapytania dostaniemy informacje zwrotną z lokalizacją foldera, w którym będą zapisywane wszystkie pliki graficzne. W talicy 3.5 został przedstawiony przykładowy wynik zwrotny.

Tablica 3.5: Wyniki wywołania zapytania z lokalizacją folderu data.
Źródło: Własne

	data_directory
1	C:/Program Files/PostgreSQL/10/data

3.3.2 Wykres

Wykres[8] jest graficzną formą przedstawienia danych. Istnieje wiele typów wykresów, do których należą mniedzy innymi wykres punktowy, liniowy, słupkowy czy kołowy. W implementacji skorzystam z wykresów liniowych. Zastosowując język proceduralny PL/R możemy stworzyć w nim funkcje rysującą wykres zapisany do pliku graficznego na serwerze. Implementacja wykresów funkcji "f_graph" dwuparametrowej w bloku kodu 3.9 oraz trzyparametrowej w bloku kodu 3.10.

```
CREATE OR REPLACE FUNCTION f_graph(text, text)
RETURNS setof void AS '
sql<-paste("select daterep,",arg1 ," from covid where
countriesandterritories=' '", arg2, "' ' order by
daterep ASC", sep="")
rs <- pg.spi.exec(sql)
png("myplot.png")
plot(rs[,2],xlab=arg2,ylab="nowe przypadki",type="l",
col="blue")
dev.off()
' LANGUAGE plr;
```

Rys. 3.9: Kod funkcji dwuparametrowej f_graph do rysowania wykresu. Źródło: Własne

Funkcja przyjmuje dwa parametry. Pierwszym z nich jest określenie danych jakie mają być zaprezentowane na histogramie, natomiast drugi parametr odpowiada za kraj, dla którego ma być przeprowadzona wizualizacja. Funkcja w pierwszej kolejności wykonuje zapytanie do bazy danych i przypisuje odowiednie dane do ramki danych. Następnie tworzy plik o nazwie ”myplot” z formatem danych typu png, do którego zapisywany jest wykres liniowy. Aby wywołać funkcje należy wykonać zapytanie w języku proceduralnym zaprezentowane w bloku kodu 3.11.

```
CREATE OR REPLACE FUNCTION f_graph(text, text, text)
RETURNS setof void AS '
sql<-paste("select day,",arg1 ," from covid where
countriesandterritories=' '", arg3, "' ' and
month=' '", arg2, "' 'order by day ASC", sep="")
rs <- pg.spi.exec(sql)
xopis<-paste("miesiac", arg2)
png("myplot2.png")
plot(rs[,1],rs[,2],xlab=xopis,ylab="nowe przypadki"
,type="l",col="blue")
dev.off()
' LANGUAGE plr;
```

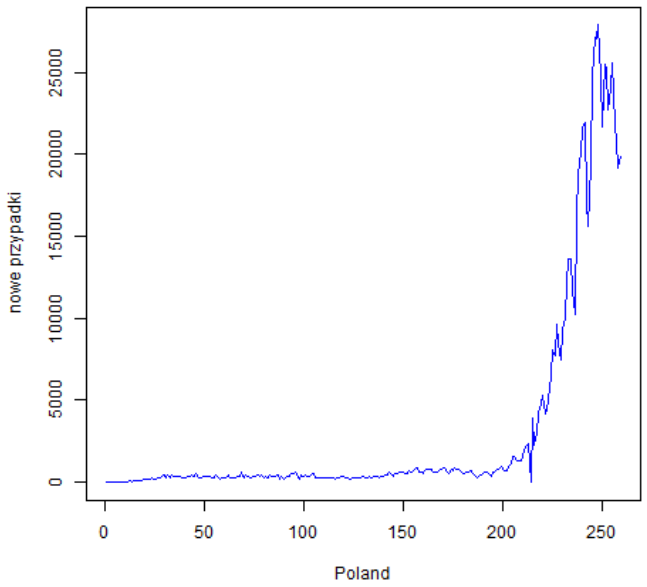
Rys. 3.10: Kod funkcji trzyparametrowej f_graph do rysowania wykresu. Źródło: Własne

Funkcja przyjmuje trzy parametry. Pierwszym z nich jest określenie danych jakie mają być zaprezentowane na histogramie natomiast drugi odpowiada liczbie miesiąca, dla którego mają być zaprezentowane dane. Trzeci parametr odpowiada za kraj dla którego ma być przeprowadzona wizualizacja. Funkcja w pierwszej kolejności wykonuje zapytanie do bazy danych i przypisuje odowiednie dane do ramki danych. Następnie tworzy plik o nazwie "myplot2" z formatem danych typu png, do którego zapisywany jest wykres liniowy. Aby wywołać funkcje należy wykonać zapytanie w języku proceduralnym zaprezentowane w bloku kodu 3.11.

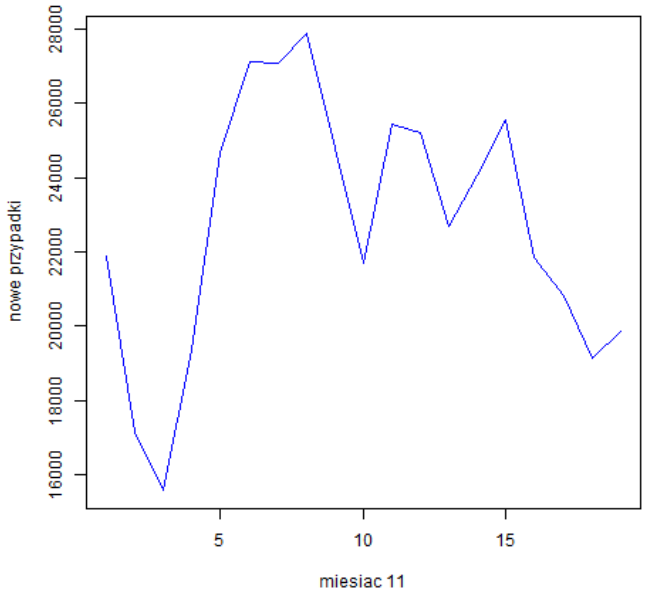
```
1 SELECT * from f_graph('cases','Poland');
2 SELECT * from f_graph('cases','11','Poland');
```

Rys. 3.11: Kod zapytania SQL do wywołania funkcji f_graph. Źródło: Własne

W wyniku wywołania funkcji dwuparametrowej otrzymamy wykres liniowy zaprezentowany na rysunku 3.12. Przedstawia on liczbe nowych przypadków zakażenia wirusem covid-19 od początku pandemii do ostaniej aktualizacji danych w bazie. Natomiast funkcja trzyparametrowa rysuje wykres liniowy przedstawiony na rysunku 3.13. Funkcja ta przedstawia liczbe nowych przypadków zakażenia wirusem covid-19 dla danego miesiąca. Obie funkcje przedstawiają dane dla Polski.



Rysunek 3.12: Wykres funkcji dwuparametrowej. Źródło: Własne



Rysunek 3.13: Wykres funkcji trzyparametrowej. Źródło: Własne

3.3.3 Histogram

Histogram[16] jest jednym z bardziej popularnych wykresów statystycznych. Za jego pomocą możemy przedstawić liczebność obserwacji w danych w przedziałach. Zastosowując język proceduralny PL/R możemy stworzyć w nim funkcje rysującą histogram zapisany do pliku graficznego na serwerze. Implementacja histogramu funkcji "*hist*" w bloku kodu 3.15. Jednakże przed tym w pierwszej kolejności należy utworzyć tabele przechowująca wartości zwrócone, z których funkcja korzysta do narysowania histogramu. Utworzenie tabeli zaprezentowane w bloku kodu 3.14.

```
1 CREATE TABLE hist (
2 miesiąc double precision,
3 dane integer
4 );
```

Rys. 3.14: Kod zapytania SQl do utworzenia tabeli hist. Źródło: Własne

```
CREATE OR REPLACE FUNCTION hist(text, text)
returns setof hist as '
sql<-paste("select sum(", arg1, "),
month from covid where
countriesandterritories=' '", arg2, "' 'group by
month order by month ASC", sep="")
rs <- pg.spi.exec(sql)
png("myhist.png");
barplot(rs[,1],
main = "Laczna liczba przypadkow w miesiacach",
xlab = "Lacznie nowych przypadkow",
ylab = "miesiac",
xlim=c(0,300000),
names.arg = c("Marzec", "Kwiecien", "Maj", "Czerwiec",
"Lipiec","Sierpien","Wrzesien","Pazdziernik",
"Listopad"), col = "darkred", horiz = TRUE)
dev.off()
return(data.frame(miesiac = rs[,2],dane=rs[,1]))
' language 'plr';
```

Rys. 3.15: Kod funkcji hist do rysowania histogramu. Źródło: Własne

Funkcja przyjmuje dwa parametry. Pierwszym z nich jest określenie danych jakie maja być zaprezentowane na histogramie, natomiast drugi parametr odpowiada za kraj, dla którego ma być przeprowadzona wizualizacja. Funkcja w pierwszej kolejności wykonuje zapytanie do bazy danych i przypisuje odowiednie dane do ramki danych. Następnie tworzy plik o nazwie "myhist" z formatem danych typu png, do którego zapisywany jest histogram. Aby wywołać funkcje należy wykonać zapytanie w języku proceduralnym zaprezentowane w bloku kodu 3.16.

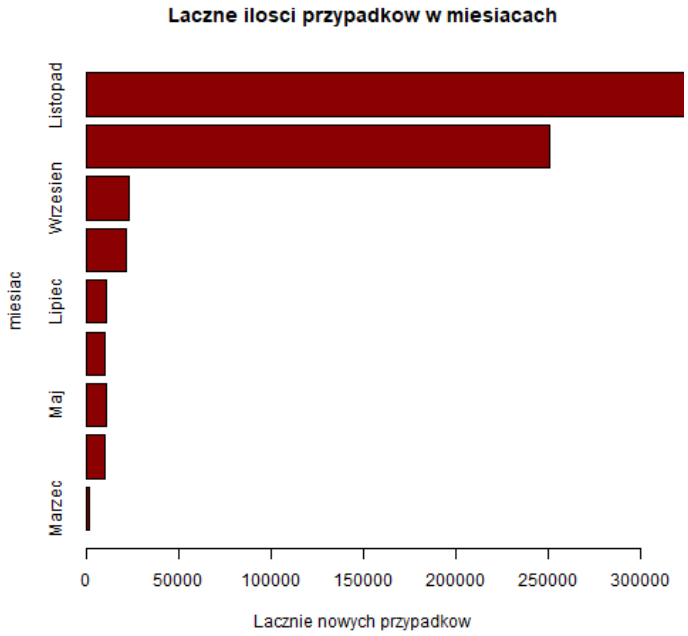
```
select * from hist('cases','Poland');
```

Rys. 3.16: Kod zapytania SQL do wywołania funkcji hist. Źródło: Własne

W wyniku zapytania otrzymamy tabele 3.6 ze zwróconymi wynikami, na których opierał się histogram oraz prezentacje graficzną histogramu na rysunku 3.17 znajdującą się w docelowym folderze na serwerze.

Tablica 3.6: Wyniki zwrotne funkcji hist. Źródło: Własne

miesiąc	dane
3	2055
4	10585
5	10931
6	10583
7	10877
8	21839
9	23092
10	250872
11	431989



Rysunek 3.17: Histogram. Źródło: Własne

3.4 Principal Component Analysis

3.4.1 Czym jest Principal Component Analysis

Principal Component Analysis[10] (skrót PCA) jest to analiza głównych składowych. PCA polega na wyznaczeniu nowej bazy przestrzeni, w której poszczególne wersy uporządkowane są względem malejącej wariancji danych. Metoda może być wykorzystana między innymi do redukcji liczby wymiarów zbioru danych (zmniejszenia liczby zmiennych wymaganych do reprezentacji zbioru) przy jednoczesnym zachowaniu wysokiego współczynnika wariancji. Zestawy danych reprezentowane przez mniejszą liczbę wymiarów są łatwiejsze do eksploracji i wizualizacji danych oraz ułatwiają i przyśpieszają analizę danych dla algorytmów uczenia maszynowego[1)]. Analiza PCA opiera się o wyznaczeniu osi, które zachowały największą wartość wariancji zbioru.

3.4.2 Standaryzacja

Standaryzacja[9] służy do ujednolicenia zakresu ciągu zmiennych w taki sposób, aby każda z nich miała jednakowy udział w analizie. Może być ona konieczna jeśli nie chcemy wziąć pod uwagę różnic między zakresami zmiennych początkowych. Należy pamiętać, że większe zakresy zmiennych będą dominować nad zmiennymi o małych zakresach. Przekształcenie każdego wymiaru danych do porównywalnych skal zróżnicowania wartości pomoże zapobiec różnicom między zakresami. Kiedy standaryzacja dobiegnie końca to wszystkie zmienne zostają przekształcone do tej samej skali. Standaryzacje można opisać wzorem, w którym od wartości odejmujemy średnią a następnie dzielimy to przez odchylenie standardowe.

$$z = \frac{wartosc - srednia}{odchylenie \quad standardowe} \tag{3.7}$$

3.4.3 Macierz kowariancji

Macierz kowariancji[9] obliczana jest w celu zidentyfikowania zależności pomiędzy zmiennymi. Obliczana jest w celu identyfikacji zmiennych zależnych (silnie skorelowanych), które, jeśli występują wspólnie, nie wnoszą dodatkowych istotnych informacji. Jest to macierz symetryczna "$p * p$", gdzie p przedstawia liczbę wymiarów. Macierz ta zawiera kowariancje ze wszystkimi możliwymi parami zmiennych początkowych. Przykładowo dla zbioru trójwymiarowego ze zmiennymi x, y i z macierz

[1)] uczenie maszynowe - jest to technologia, która na podstawie analizy danych uczy komputery wykonywania zadań. Źródło: Strona SAP[14]

kowariancji na postać:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} \tag{3.8}$$

Kowariancja zmiennej ze sobą w rzeczywistości jest jej wariancją ($Cov(a,a) = Var(a)$) co powoduje, że po przekątnej otrzymujemy wariancje każdej zmiennej początkowej. Podsumowując można powiedzieć, że macierz kowariancji nie jest niczym więcej niż tabelą podsumowującą korelacje między wszystkimi parami zmiennych.

3.4.4 Wektory i wartości własne miacierzy kowariancji

Wektory i wartości własne[9] jest to pojęcie algebraiczne. Oblicza się je z macierzy kowariancji, żeby określić główne składniki danych. Występują zawsze w parach, czyli każdemu wektorowi własnemu odpowiada jakaś wartość własna. Liczba wektorów i wartości równa jest liczbie wymiarów danych. Czyli na przykład dwuwymiarowy zestaw danych posiada 2 zmienne, 2 wektory własne oraz 2 wartości własne. Wektory własne macierzy kowariancji są w rzeczywistości kierunkami osi. Natomiast wartości własne są współczynnikami dołączonymi do wektorów własnych. Wartości te przechowują wielkości wariancji, która przenoszona jest w każdym głownym składniku. Kiedy uszeregujemy od najwyższej do najniższej wektory własne w kolejności ich wartości to w wyniku tej operacji otrzymamy główne składniki w kolejności ich istotności.

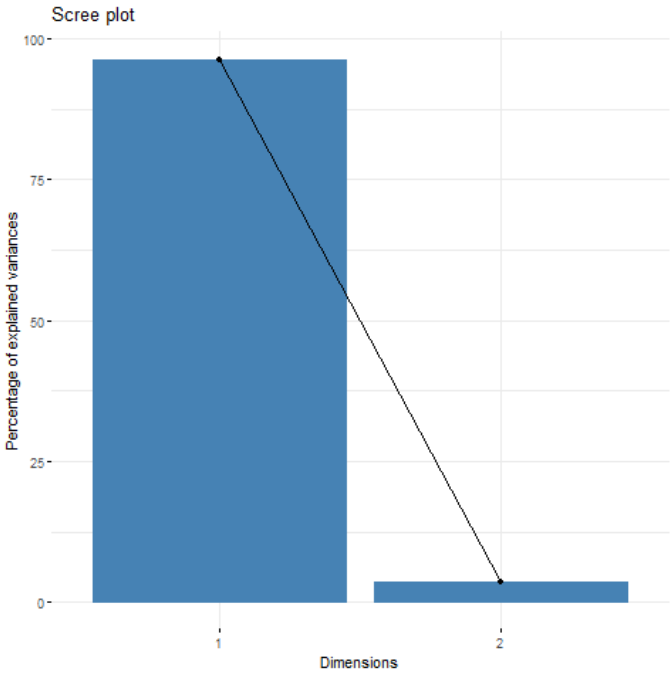
3.4.5 Główne składniki

Główne składniki są to nowe nieskorelowane zmienne, które zbudowane są jako kombinacje liniowe zmiennych początkowych. Konstruowane są w taki sposób, że odpowiadają możliwie największej wariancji w zbiorze danych. Przykładowo, do pierwszego składnika przypisywana jest możliwie największa wariancja zbioru danych, następnie drugiemu składnikowi, który jest nieskorelowany z pierwszym głównym składnikiem, przypisuje się następną możliwie największą wariancję. Cała operacja trwa do momentu obliczenia sumy p głównyh składników, która równa jest pierwotnej liczbie zmiennych. Więc jeżeli nasze dane są pięciowymiarowe to dadzą one 5 głównych komponentów.Takie przyporządkowanie informacji pozwala zmniejszyć wymiarowość bez utraty dużej ilości informacji. Innymi słowy można powiedzieć, że główne składniki reprezentują kierunki danych, które tłumaczą maksymalną wielkość wariancji. Można więc pomyśleć że są to takie nowe osie, które zapewniają

najlepszy kąt widzenia i oceny danych.Dla danych dotyczących nowych przypadków zarażeń i zgonów wywołanych wirusem covid-19 wizualizacja przedstawiona na rysunku 3.19, natomiast funkcja jak i jej wywołanie przedstawione we fragmencie kodu 3.18.

```
create or replace function procent_wariancji()
returns float8[] as '
sql<-paste("SELECT cases, deaths
FROM covid
where countriesandterritories = ''Poland''
order by cases", sep="")
rs <- pg.spi.exec(sql)
prco<-prcomp(rs, scale = FALSE)
library(factoextra)
png("procent_wariancji.png")
fviz_eig(prco)
dev.off()
' language 'plr';
select * from procent_wariancji();
```

Rys. 3.18: Kod funkcji procent_wariancji do wizualizacji procentu wariancji dla każdego pc. Źródło: Własne



Rysunek 3.19: Wizualizacja procentu wariancji dla każdego pc. Źródło: Własne

3.4.6 Wektory cech

Zbiór wartości cech nazywany jest wektorem cech[9].Zawiera on wektory własne składników, które są kolumnami macierzy. W wyniku tej operacji możemy stwierdzić, że jest to pierwszy krok służący w redukcji wymiarowości. Możemy teraz napisać funkcję przedstawioną na fragmencie kodu 3.20, która utworzy nam wektor cech. W wyniku wywołania funkcji otrzymamy dane przedstawione w tabeli 3.7.

```
create or replace function wektor_cech()
returns float8[] as '
sql<-paste("SELECT cases, deaths
FROM covid
where countriesandterritories = ''Poland''
order by cases", sep="")
rs <- pg.spi.exec(sql)
prco<-prcomp(rs, scale = FALSE)
prco[["rotation"]]
' language 'plr';
select * from wektor_cech();
```

Rys. 3.20: Kod funkcji wektor_cech do tworzenia wektora cech. Źródło: Własne

Tablica 3.7: Wyniki wektora cech. Źródło: Własne

PC1	PC2
0.99991133219478	-0.013316446540265
0.013316446540265	0.99991133219478

3.4.7 Przekształcanie danych wzdłóż osi głównych komponentów

Z wykorzystaniem wektora cech, który został utworzony za pomocą wektorów własnych macierzy kowariancji możemy przekształcić dane[9] z oryginalnej osi na dane, które reprezentowane są przez główne składowe. Stąd też wzięła się nazwa "Analiza głównych składowych". Do wykonania przekształcenia można użyć pomnożenia transpozycji oryginalnego zestawu danych z transpozycją wektora cech.

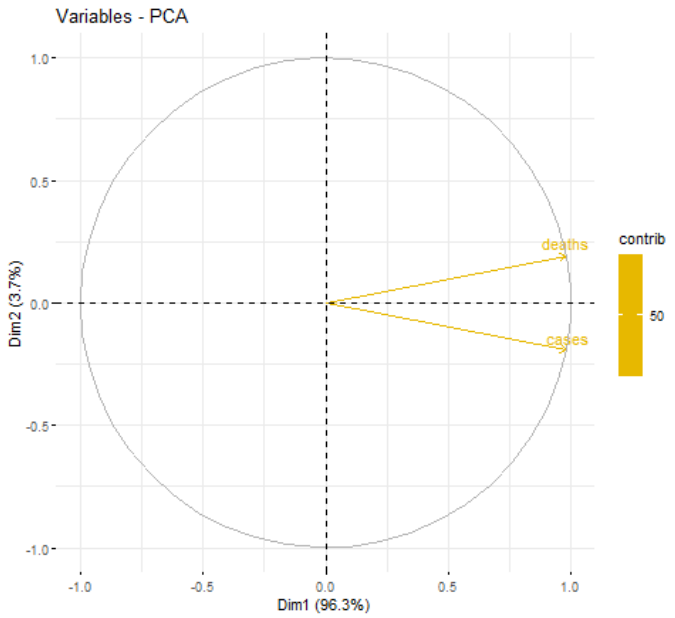
$$FinalDataSet = FeatureVector^T * StandarizedOriginalDataSet^T \tag{3.9}$$

Dla przykładu utworzymy teraz wykres zmiennych. Na takim wykresie dodatnio skolerowane zmienne wskazują tą samą stronę, natomiast ujemnie skolerowane zmienne wskazują przeciwną stronę wykresu. Do stworzenie wykresu potrzebna będzie funkcja "*wykres_zmiennych*", która zaprezentowana jest w kodzie 3.21.

```
create or replace function wykres_zmiennych()
returns float8[] as '
sql<-paste("SELECT cases, deaths FROM covid
where countriesandterritories = ''Poland''
order by cases", sep="")
rs <- pg.spi.exec(sql)
prco<-prcomp(rs, scale = FALSE)
library(factoextra)
png("wykres_zmiennych.png")
fviz_pca_var(prco, col.var = "contrib",
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE)
dev.off()
' language 'plr';
select * from wykres_zmiennych();
```

Rys. 3.21: Kod funkcji wykres_zmiennych do wizualizacji wykresu zmiennych. Źródło: Własne

W wyniku wizualizacji otrzymamy wykres zaprezentowany na rysunku 3.22.



Rysunek 3.22: Wizualizacja wykresu zmiennych. Źródło: Własne

Zakończenie

Język proceduralny PL/R w bazie danych ma szerokie zastosowanie do przeprowadzenia analizy statystycznej i wizualizacji danych. Przy jego pomocy można w prosty sposób i praktycznie bez żadnych ograniczeń przeprowadzić analizę danych jak i ich wizualizcję. Analiza statystyczna jest niezwykle ważna w celu uzyskania odpowiednich informacji na temat badanych danych. Dzięki wizualizacji danych możemy lepiej je zrozumieć oraz zobaczyć jak rozkładają się w zależności od zmiennych. Principal Component Analysis jest przydatnym narzędziem w rzutowaniu wykresów wielowymiarowych do 2D. Głównym celem pracy inżynierskiej było stworzenie przykładowej implementacji zastosowania języka proceduralnego do przeprowadzenia analizy statystycznej i wizualizacji danych za pomocą PL/R w bazie danych. Aktualna sytuacja na świecie przyczyniła się do wybrania bazy danych covid-19. Wartym uwagi jest również test Shapiro-Wilka. Przy jego pomocy możemy przeprowadzić prosty test informujący o tym czy rozkład naszych danych jest podobny do rozkładu normalnego.

Literatura

[1] A. Z. Artur Suchwałko, Agnieszka Suchwałko. Wprowadzenie do r. https://cran.r-project.org/doc/contrib/wprowadzenie_do_R.pdf. Dostęp: 2021-01-10.

[2] J. Conway. Chapter 3. functions and arguments. http://www.joeconway.com/doc/plr-funcs.html. Dostęp: 2021-01-10.

[3] J. Conway. Chapter 4. passing data values. http://www.joeconway.com/doc/plr-data.html. Dostęp: 2021-01-10.

[4] J. Conway. Chapter 7. postgresql support functions. http://www.joeconway.com/doc/plr-pgsql-support-funcs.html. Dostęp: 2021-01-10.

[5] J. Conway. Pl/r user's guide - r procedural language, chapter 1. overview. http://joeconway.com/plr/doc/plr-overview.html. Dostęp: 2021-01-10.

[6] D. Cramer. Rel8 4. https://github.com/postgres-plr/plr/releases/tag/REL8_4. Dostęp: 2021-01-10.

[7] E. C. ds. Zapobiegania i Kontroli Chorób. Dane — portal otwartych danych unii europejskiej. https://data.europa.eu/euodp/pl/data/. Dostęp: 2021-01-10.

[8] A. Goroncy. Wykresy w środowisku r. https://www-users.mat.umk.pl//~iggypop/wykresy_w_R.pdf. Dostęp: 2021-01-10.

[9] Z. Jaadi. A step-by-step explanation of principal component analysis. https://builtin.com/data-science/step-step-explanation-principal-component-analysis. Dostęp: 2021-01-10.

[10] I. T. Jolliffe, J. Cadima. Principal component analysis: a review and recent developments. https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202. Dostęp: 2021-01-10.

[11] A. Wlizło. Zastosowanieplr. https://github.com/adrian-wlizlo/ZastosowaniePLR. Dostęp: 2021-01-10.

[12] 5 najważniejszych metod statystycznych w analizie danych. https://www.statystyka.az.pl/5-najwazniejszych-metod-statystycznych-w-analizie-danych.php. Dostęp: 2021-01-10.

[13] The comprehensive r archive network. https://cran.r-project.org/. Dostęp: 2021-01-10.

[14] Czym jest uczenie maszynowe (machine learning)? https://www.sap.com/poland/insights/what-is-machine-learning.html. Dostęp: 2021-01-10.

[15] Download the rstudio ide. https://rstudio.com/products/rstudio/download/. Dostęp: 2021-01-10.

[16] Histogram, wykres rozkładu zmiennej. https://www.naukowiec.org/wiedza/statystyka/histogram_764.html. Dostęp: 2021-01-10.

[17] Kwartyle, miary statystyczne. https://www.naukowiec.org/wiedza/statystyka/kwartyle_699.html. Dostęp: 2021-01-10.

[18] Mediana, wartość środkowa zbioru. https://www.naukowiec.org/wiedza/statystyka/mediana_704.html. Dostęp: 2021-01-10.

[19] Odchylenie standardowe, pomiar zróżnicowania. https://www.naukowiec.org/wiedza/statystyka/odchylenie-standardowe_703.html. Dostęp: 2021-01-10.

[20] pakiet. https://sjp.pwn.pl/slowniki/pakiet.html. Dostęp: 2021-01-10.

[21] Postgresql. https://vavatech.pl/technologie/bazy-danych/postgresql. Dostęp: 2021-01-10.

[22] Postgresql database download. https://www.enterprisedb.com/downloads/postgres-postgresql-downloads. Dostęp: 2021-01-10.

[23] R-3.6.0 for windows (32/64 bit). https://cran.r-project.org/bin/windows/base/old/3.6.0/. Dostęp: 2021-01-10.

[24] Regresja liniowa, analiza regresji. https://www.naukowiec.org/wiedza/statystyka/regresja-liniowa_765.html. Dostęp: 2021-01-10.

[25] Tablice rozkładu w (test shapiro-wilka). https://www.naukowiec.org/tablice/statystyka/rozklad-w-test-shapiro-wilka-_335.html. Dostęp: 2021-01-10.

[26] Test shapiro-wilka. https://www.naukowiec.org/wiedza/statystyka/test-shapiro-wilka_758.html. Dostęp: 2021-01-10.

[27] Wariancja, miara zmienności. https://www.naukowiec.org/wiedza/statystyka/wariancja_719.html. Dostęp: 2021-01-10.

[28] Wolne oprogramowanie. https://pl.wikipedia.org/wiki/Wolne_oprogramowanie. Dostęp: 2021-01-10.

[29] Wzór na odchylenie standardowe. https://www.naukowiec.org/wzory/statystyka/odchylenie-standardowe_5.html. Dostęp: 2021-01-10.

[30] Wzór na regresję liniową. https://www.naukowiec.org/wzory/statystyka/regresja-liniowa_80.html. Dostęp: 2021-01-10.

[31] Wzór na test normalności rozkładu shapiro-wilka. https://www.naukowiec.org/wzory/statystyka/test-shapiro-wilka_333.html. Dostęp: 2021-01-10.

[32] Wzór na wariancję. https://www.naukowiec.org/wzory/statystyka/wariancja_6.html. Dostęp: 2021-01-10.

[33] Średnia arytmetyczna. https://www.matemaks.pl/srednia-arytmetyczna.html. Dostęp: 2021-01-10.

Załączniki

Załączniki zawierają wszystkie kody źródłowe, plik README, który zawiera opis uruchomienia kodów źródłowych oraz plik zawierający dane wirusa COVID-19 do bazy danych. Pliki dostępne są również w repozytorium umieszczonym na platformie Github[11]. Do pracy załączone zostały następujące elementy.
Pliki:

- README.txt
- covid

Kody źródłowe:

- konfiguracja.sql
- tabele.sql
- testy.sql
- funkcja summary.sql
- funkcje agregujące.sql
- pca procent wariancji.sql
- wektor cech.sql
- wizualizacja danych.sql
- wykres zmiennych.sql

Spis rysunków

Spis tablic

Spis kodów źródłowych