2025

POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Analiza i Wizualizacja Ruchu Sieciowego z Wykorzystaniem Języka R

INŻYNIERIA I ANALIZA DANYCH

DEREŃ ADRIAN, DRAGUŁA BARTŁOMIEJ

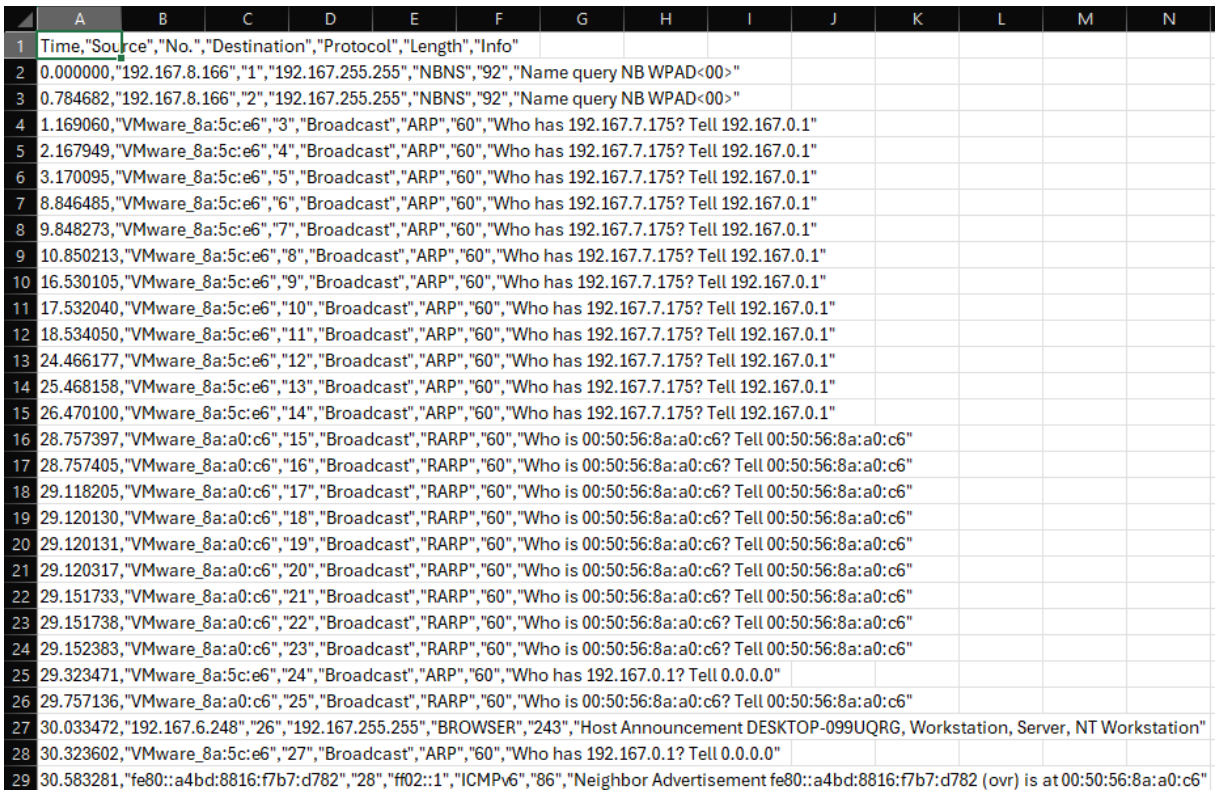
Spis treści

Zbieranie danych o ruchu sieciowym

Dane, które zostaną wprowadzone do skryptu oraz na których przeprowadzone zostaną poniższe operacje, zostały przechwycone przy pomocy programu Wireshark. Zawierają one informacje o pakietach przechwyconych w ciągu dwudziestu minut. Dane zostały wyeksportowane do pliku Microsoft Excel o formacie *.csv*. Zawierają blisko czterysta tysięcy rekordów oraz siedem kolumn. Kolumny te to odpowiednio:

- *Time* – czas przechwycenia pakietu w sekundach z sześcioma miejscami po przecinku,
- *Source* – źródło z którego wysyłany był pakiet,
- *No* – numer obserwacji,
- *Destination* – numer IP celu,
- *Protocol* – typ protokołu,
- *Length* – całkowita wielkość pakietu,
- *Info* – dodatkowe informacje.

Dane po wyeksportowaniu do pliku *.csv* zaprezentowane zostały poniżej.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Time,"Source","No.","Destination","Protocol","Length","Info"													
2	0.000000,"192.167.8.166","1","192.167.255.255","NBNS","92","Name query NB WPAD<00>"													
3	0.784682,"192.167.8.166","2","192.167.255.255","NBNS","92","Name query NB WPAD<00>"													
4	1.169060,"VMware_8a:5c:e6","3","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
5	2.167949,"VMware_8a:5c:e6","4","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
6	3.170095,"VMware_8a:5c:e6","5","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
7	8.846485,"VMware_8a:5c:e6","6","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
8	9.848273,"VMware_8a:5c:e6","7","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
9	10.850213,"VMware_8a:5c:e6","8","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
10	16.530105,"VMware_8a:5c:e6","9","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
11	17.532040,"VMware_8a:5c:e6","10","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
12	18.534050,"VMware_8a:5c:e6","11","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
13	24.466177,"VMware_8a:5c:e6","12","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
14	25.468158,"VMware_8a:5c:e6","13","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
15	26.470100,"VMware_8a:5c:e6","14","Broadcast","ARP","60","Who has 192.167.7.175? Tell 192.167.0.1"													
16	28.757397,"VMware_8a:a0:c6","15","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
17	28.757405,"VMware_8a:a0:c6","16","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
18	29.118205,"VMware_8a:a0:c6","17","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
19	29.120130,"VMware_8a:a0:c6","18","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
20	29.120131,"VMware_8a:a0:c6","19","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
21	29.120317,"VMware_8a:a0:c6","20","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
22	29.151733,"VMware_8a:a0:c6","21","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
23	29.151738,"VMware_8a:a0:c6","22","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
24	29.152383,"VMware_8a:a0:c6","23","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
25	29.323471,"VMware_8a:5c:e6","24","Broadcast","ARP","60","Who has 192.167.0.1? Tell 0.0.0.0"													
26	29.757136,"VMware_8a:a0:c6","25","Broadcast","RARP","60","Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6"													
27	30.033472,"192.167.6.248","26","192.167.255.255","BROWSER","243","Host Announcement DESKTOP-099UQRG, Workstation, Server, NT Workstation"													
28	30.323602,"VMware_8a:5c:e6","27","Broadcast","ARP","60","Who has 192.167.0.1? Tell 0.0.0.0"													
29	30.583281,"fe80::a4bd:8816:f7b7:d782","28","ff02::1","ICMPv6","86","Neighbor Advertisement fe80::a4bd:8816:f7b7:d782 (ovr) is at 00:50:56:8a:a0:c6"													

Importowanie danych do R

Zbiór danych wykorzystany do tego projektu to plik o formacie *.csv*. Z tego względu do zaimportowania go, zostanie wykorzystana komenda *read.csv*, która służy do wczytywania plików wartości oddzielanych przecinkami.

```
##############################################################################################
# Importowanie danych do R
##############################################################################################

dane <- read.csv("C:/Bartek/STUDIA/Magisterka/Semestr 2/Pracownia problemowa/Dane_wireshark.csv")

##############################################################################################
```

Dane po wprowadzeniu do środowiska *RStudio* zostały zaprezentowane poniżej.

	Time	Source	No.	Destination	Protocol	Length	Info
1	0.000000	192.167.8.166	1	192.167.255.255	NBNS	92	Name query NB WPAD<00>
2	0.784682	192.167.8.166	2	192.167.255.255	NBNS	92	Name query NB WPAD<00>
3	1.169060	VMware_8a:5c:e6	3	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
4	2.167949	VMware_8a:5c:e6	4	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
5	3.170095	VMware_8a:5c:e6	5	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
6	8.846485	VMware_8a:5c:e6	6	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
7	9.848273	VMware_8a:5c:e6	7	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
8	10.850213	VMware_8a:5c:e6	8	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
9	16.530105	VMware_8a:5c:e6	9	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
10	17.532040	VMware_8a:5c:e6	10	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
11	18.534050	VMware_8a:5c:e6	11	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
12	24.466177	VMware_8a:5c:e6	12	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
13	25.468158	VMware_8a:5c:e6	13	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
14	26.470100	VMware_8a:5c:e6	14	Broadcast	ARP	60	Who has 192.167.7.175? Tell 192.167.0.1
15	28.757397	VMware_8a:a0:c6	15	Broadcast	RARP	60	Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6
16	28.757405	VMware_8a:a0:c6	16	Broadcast	RARP	60	Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6
17	29.118205	VMware_8a:a0:c6	17	Broadcast	RARP	60	Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6
18	29.120130	VMware_8a:a0:c6	18	Broadcast	RARP	60	Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6
19	29.120131	VMware_8a:a0:c6	19	Broadcast	RARP	60	Who is 00:50:56:8a:a0:c6? Tell 00:50:56:8a:a0:c6

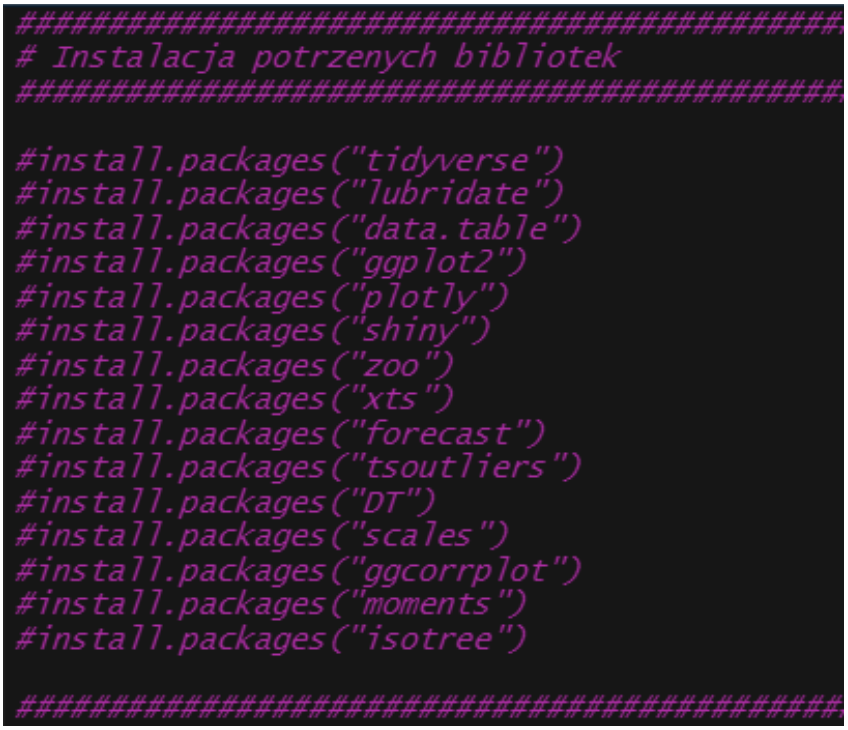
Jak można zauważyć dane zostały poprawnie wczytane oraz podzielone na odpowiednie kolumny. Dzięki temu można teraz przystąpić do pracy nad wczytanym zbiorem danych.

Przy użyciu polecenia *head()*, możemy wyświetlić pierwsze kilka rekordów oraz nazwy kolumn oraz wierszy.

```
> head(dane)
      Time          Source No.      Destination Protocol Length                                 Info
1 0.000000   192.167.8.166   1 192.167.255.255     NBNS     92                 Name query NB WPAD<00>
2 0.784682   192.167.8.166   2 192.167.255.255     NBNS     92                 Name query NB WPAD<00>
3 1.169060 VMware_8a:5c:e6   3        Broadcast      ARP     60 Who has 192.167.7.175? Tell 192.167.0.1
4 2.167949 VMware_8a:5c:e6   4        Broadcast      ARP     60 Who has 192.167.7.175? Tell 192.167.0.1
5 3.170095 VMware_8a:5c:e6   5        Broadcast      ARP     60 Who has 192.167.7.175? Tell 192.167.0.1
6 8.846485 VMware_8a:5c:e6   6        Broadcast      ARP     60 Who has 192.167.7.175? Tell 192.167.0.1
```

Instalacja oraz wczytanie wymaganych bibliotek

Do stworzenia tego projektu, wymagane jest użycie niektórych bibliotek w RStudio. W tym celu każdą z nich należy zainstalować, a następnie wczytać. Wykorzystane biblioteki zostały zaprezentowane poniżej.



```
##############################################
# Instalacja potrzenych bibliotek
##############################################

#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("data.table")
#install.packages("ggplot2")
#install.packages("plotly")
#install.packages("shiny")
#install.packages("zoo")
#install.packages("xts")
#install.packages("forecast")
#install.packages("tsoutliers")
#install.packages("DT")
#install.packages("scales")
#install.packages("ggcorrplot")
#install.packages("moments")
#install.packages("isotree")

##############################################
```

W celu lepszego zrozumienia, dlaczego wybrane zostały właśnie te biblioteki, krótko opiszemy ich zastosowanie oraz sposób wykorzystania:

- *Tidyverse* – biblioteka do wizualizacji danych,
- *Lubridate* – pomaga w prowadzeniu operacji na datach oraz godzinach,
- *Data.table* – pomaga w pracy przy dużych zbiorach danych,
- *Ggplot2* – umożliwia tworzenie zaawansowanych wizualizacji danych,
- *Plotly* – tworzenie interaktywnych wykresów,
- *Shiny* – służy do budowania interaktywnych aplikacji,
- *Zoo* – pomocny w przypadku pracy przy szeregach czasowych
- *Xts* – służy do wsparcia w pracy z szeregami czasowymi,
- *Forecast* – służy do prognozowania szeregów czasowych,
- *Tsoutliers* – pozwala wykrywać wartości odstające w szeregu,
- *DT* – służy do prezentacji danych w postaci tabel HTML
- *Scales* – daje możliwość formatowania osi wykresów,
- *Ggcorrplot* – pozwala wizualizować macierze korelacji w przyjemny dla oka sposób,
- *Moments* – biblioteka pomocna w obliczaniu momentów statystycznych,
- *Isotree* – służy do wykrywania anomalii przy użyciu algorytmu drzew izolacji.

Analiza statystyczna ruchu sieciowego

W tym rozdziale zaprezentowane zostanie obliczanie podstawowych parametrów opisujących wykorzystane dane. Ponadto sprawdzone zostaną również bardziej skomplikowane parametry analityczne oraz posłużymy się narzędziami do wizualizacji uzyskanych wyników.

Podstawowe parametry

Przy pomocy polecenia *summary()*, uzyskujemy podstawowe informacje o wczytanych danych takie jak między innymi wartości minimalne oraz maksymalne, średnią oraz medianę. Dane te dotyczą każdej z występujących kolumn.

```
> summary(dane)
      Time                              Source             No.                    Destination          Protocol           Length              Info
 Min.   :2025-01-01 01:00:00.00   192.167.7.162   :116067   Min.   :     1   192.167.7.162   :276849   TCP    :314873   Min.   :   42.0   Length:394136
 1st Qu.:2025-01-01 01:09:18.18   104.91.166.75   : 57238   1st Qu.: 98535   104.91.166.75   : 15565   TLSv1.3: 71625   1st Qu.:   60.0   Class :character
 Median :2025-01-01 01:13:01.93   74.125.9.169    : 37085   Median :197069   23.33.29.79     : 10538   ICMP   :  2690   Median : 1462.0   Mode  :character
 Mean   :2025-01-01 01:13:01.59   23.33.29.79     : 25918   Mean   :197069   74.125.9.169    :  6800   DNS    :  1999   Mean   :  985.6
 3rd Qu.:2025-01-01 01:16:26.27   173.194.133.202: 22832   3rd Qu.:295602   173.194.133.202:  6451   TLSv1.2:  1763   3rd Qu.: 1514.0
 Max.   :2025-01-01 01:20:56.92   104.91.166.113  : 18222   Max.   :394136   146.75.78.73    :  6116   ARP    :   447   Max.   :49745.0
                                  (Other)         :116774                    (Other)         : 71817   (Other):   739
```

Polecenie *str()* pokazuje w jakim formacie jest każda z kolumn.

```
> str(dane)
Classes 'data.table' and 'data.frame':  394136 obs. of  7 variables:
 $ Time       : POSIXct, format: "2025-01-01 01:00:00" "2025-01-01 01:00:00" "2025-01-01 01:00:01" "2025-01-01 01:00:02" ...
 $ Source     : Factor w/ 372 levels "0.0.0.0","10.25.3.2",..: 184 184 356 356 356 356 356 356 356 356 ...
 $ No.        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Destination: Factor w/ 308 levels "10.25.3.2","10.27.3.2",..: 153 153 304 304 304 304 304 304 304 304 ...
 $ Protocol   : Factor w/ 16 levels "ARP","BROWSER",..: 8 8 1 1 1 1 1 1 1 1 ...
 $ Length     : int  92 92 60 60 60 60 60 60 60 60 ...
 $ Info       : chr  "Name query NB WPAD<00>" "Name query NB WPAD<00>" "Who has 192.167.7.175? Tell 192.167.0.1" "Who has 192.167.7.175? Tell 192.167.0.1" ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Możemy również sprawdzić ilość występowania każdego z protokołów.

```
> table(dane$Protocol)

    ARP BROWSER    DHCP     DNS    HTTP    ICMP  ICMPv6    NBNS    OCSP    RARP   SSLv2    STUN     TCP   TLSv1 TLSv1.2 TLSv1.3
    447       8      17    1999       3    2690       9     231     258     178      11       4  314873      20    1763   71625
```

Transformacja danych do szeregu czasowego

W kolejnym kroku, dane zostały przetransformowane do szeregu czasowego co pozwoli na dodatkowe możliwości analizy.

```
###########################################################################
# Transformacja danych do szeregu czasowego
###########################################################################

dane$Time <- as.POSIXct(dane$Time, origin = "2025-01-01")

colSums(is.na(dane))

dane <- dane %>%
  mutate(
    Source = as.factor(Source),
    Destination = as.factor(Destination),
    Protocol = as.factor(Protocol)
  )

###########################################################################
```

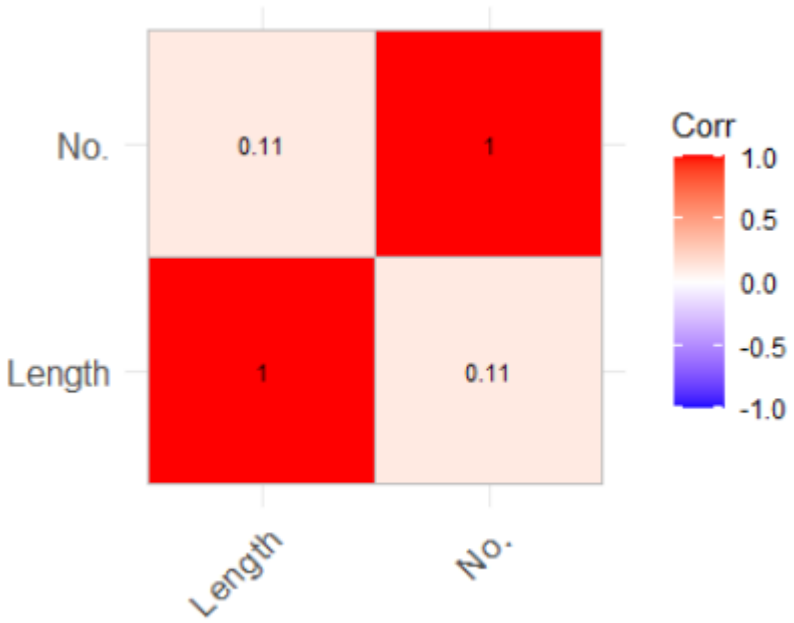
Przy pomocy polecenia *as.POSIXct()*, dane zostały przekonwertowane do szeregu czasowego. Sprawdzone zostały również brakujące wartości, a ich ilość w każdej z kolumn równa jest zero.

Macierz korelacji

Teraz utworzona zostanie macierze korelacji. Ze względu na to, że w badanych danych nie występuje wiele kolumn liczbowych, sprawdzona zostanie wyłącznie zależność liczby obserwacji w stosunku do długości pakietu.

```
##########################################################################
# Macierz korelacji
##########################################################################

korelacje <- cor(dane[, c('Length','No.')], use = "complete.obs")

# Wizualizacja macierzy korelacji
ggcorrplot(korelacje, lab = TRUE, lab_size = 3)

##########################################################################
```

Dzięki powyższym poleceniom, otrzymana zostaje wizualizacja macierzy korelacji, która została zaprezentowana poniżej.

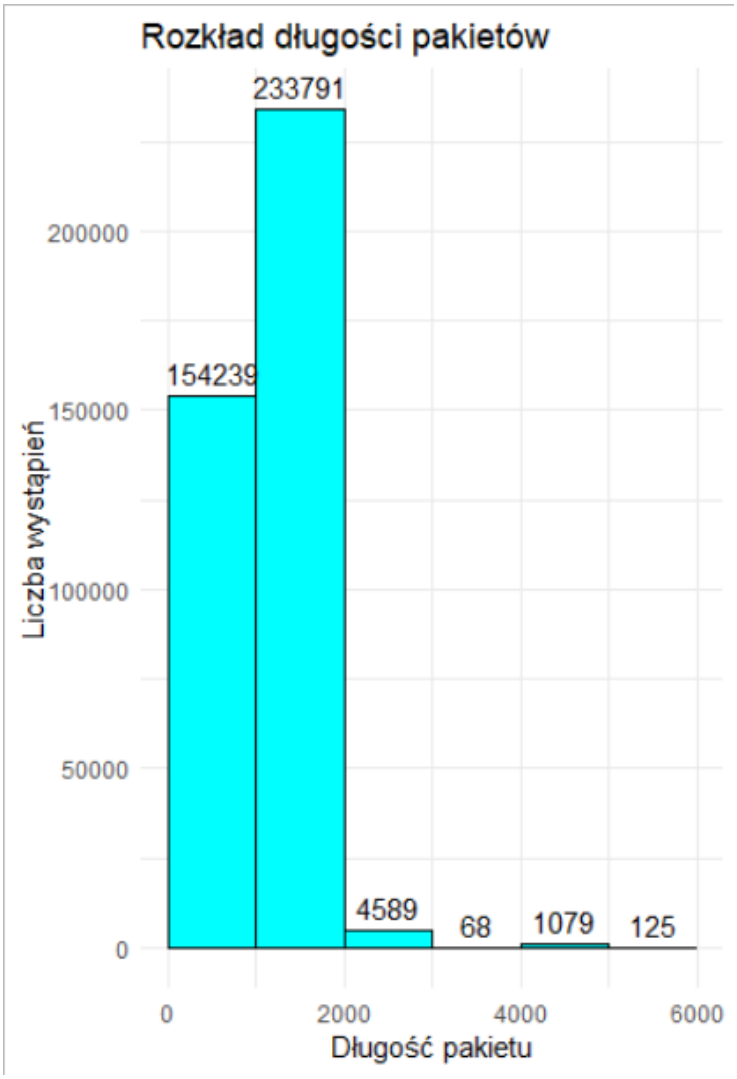


Po utworzeniu oraz zwizualizowaniu powyższej macierzy, dostajemy informację, że występuje bardzo niska korelacja dodatnia między tymi dwoma kolumnami. Oznacza to, że wraz ze wzrostem liczby obserwacji, tylko nieznacznie wzrasta również długość pakietu – a co za tym idzie, zmienne te nie są ze sobą mocno powiązane.

Rozkład długości pakietów

Utworzona została również graficzna wizualizacja rozkładu długości pakietów. Korzystając z biblioteki *ggplot2*, utworzony został histogram – na osi X znajduje się długość pakietu, w kolumnie Y liczba wystąpień.

```
##############################################################################################################
# Rozkład długości pakietów
##############################################################################################################

ggplot(dane, aes(x = Length)) +
  geom_histogram(binwidth = 1000, boundary = 0, fill = "cyan", color = "black") +
  stat_bin(binwidth = 1000, geom = "text", aes(label = ..count..), vjust = -0.5, color = "black", boundary = 0) +
  labs(title = "Rozkład długości pakietów", x = "Długość pakietu", y = "Liczba wystąpień") +
  theme_minimal() +
  xlim(0, 6000)

##############################################################################################################
```

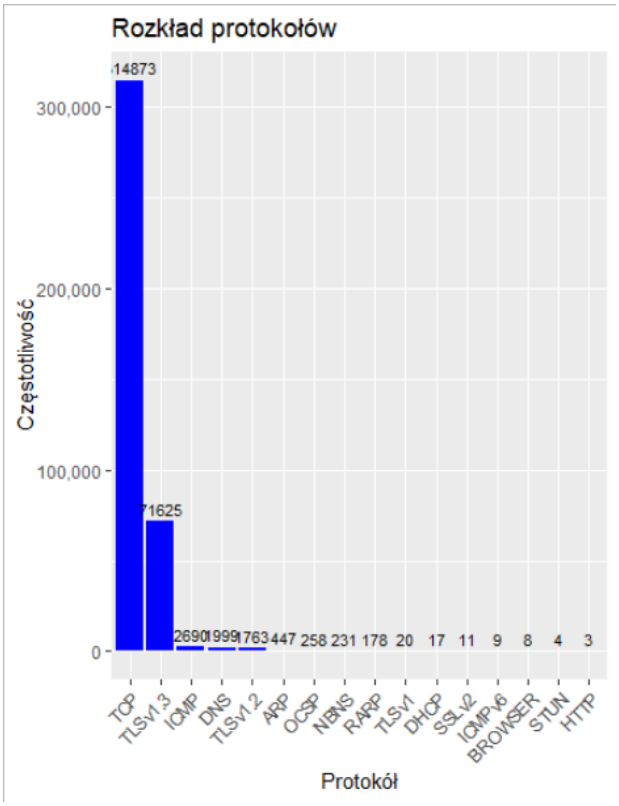


Zauważyć można, że najwięcej wystąpień mają pakiety o długości od 0 do 1000 oraz od 1000 do 2000. Pakiety o większej długości występują bardzo sporadycznie.

Rozkład protokołów

Stworzony został również rozkład protokołów. W tym celu ponownie została wykorzystana biblioteka *ggplot2*.

```
##########################################################################################
# Rozkład protokołów
##########################################################################################

protocol_count <- dane %>%
  group_by(Protocol) %>%
  summarise(Frequency = n()) %>%
  arrange(desc(Frequency))

ggplot(protocol_count, aes(x = reorder(Protocol, -Frequency), y = Frequency)) +
  geom_bar(stat = "identity", fill = "blue") +
  geom_text(aes(label = Frequency), vjust = -0.5, size = 3) +
  labs(title = "Rozkład protokołów", x = "Protokół", y = "Częstotliwość") +
  scale_y_continuous(labels = label_comma()) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

##########################################################################################
```

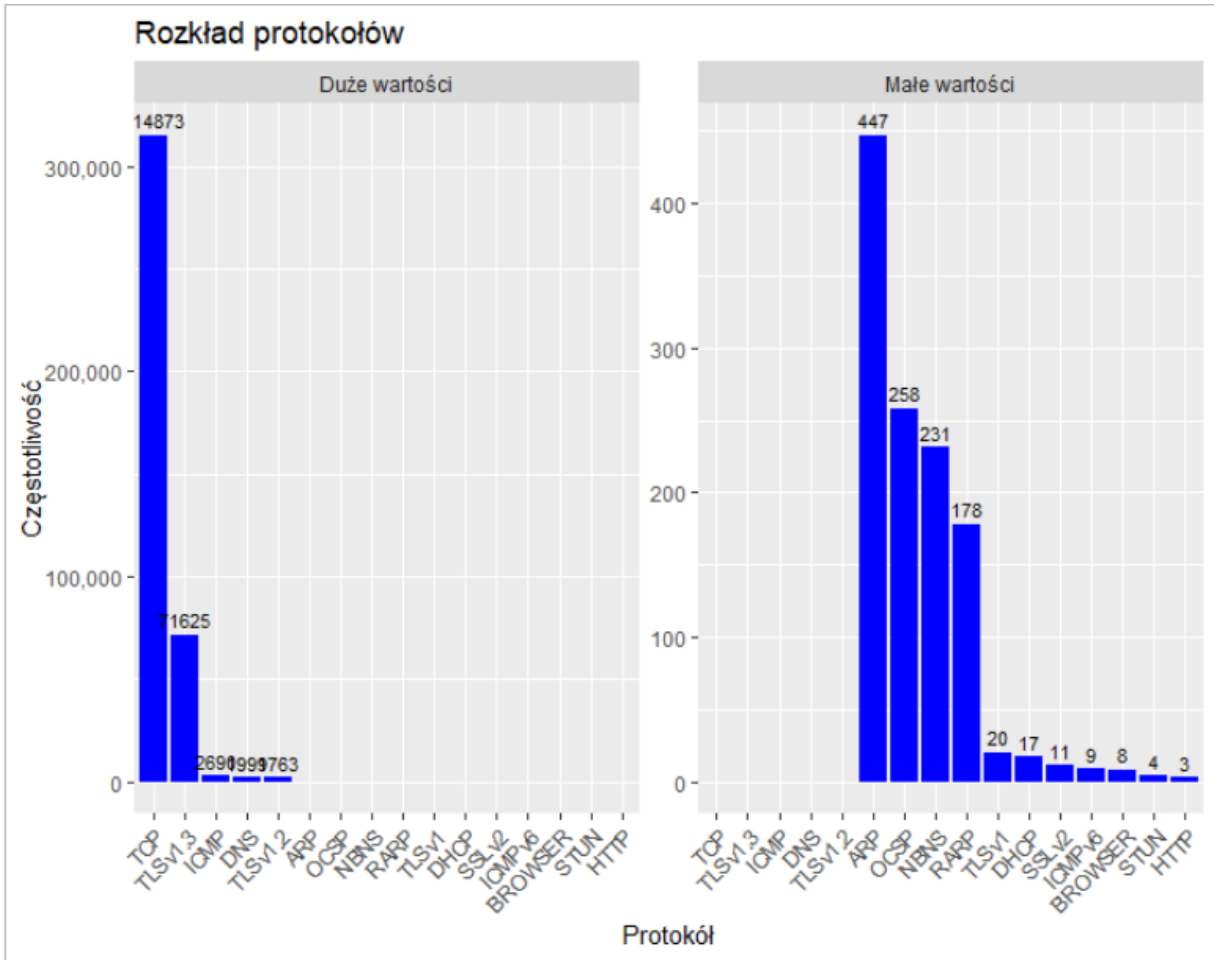


Dane pokazują, że TCP i TLSv1.3 są dominującymi protokołami w analizowanej sieci, co jest zgodne z ich rolą w zapewnianiu bezpieczeństwa w Internecie. Można z łatwością zaobserwować, że największe występowanie można przypisać protokołowi TCP. Jego liczba wystąpień to ponad 314 tysięcy.

Liczba protokołów na wykresach dla protokołów o dużej i małej liczbie pakietów

Ze względu na to, że protokoły o mniejszej liczbie wystąpień są słabo widoczne na utworzonym wykresie, dane zostaną podzielone na dwa wykresy. Na pierwszym z nich umieszczone zostaną słupki prezentujące protokoły o liczbie wystąpień większej niż 1000, na drugim zaś protokoły o mniejszej liczbie wystąpień.

```
##############################################################################################################
# Liczba protokołów na wykresach dla protokołów o dużej i małej liczbie pakietów
##############################################################################################################

# Ze względu na duży rozstrzał pomiędzy liczbą pakietów, tworzymy 2 wykresy dla lepszego zobrazowania
protocol_count <- protocol_count %>%
  mutate(Group = ifelse(Frequency > 1000, "Duże wartości", "Małe wartości"))

ggplot(protocol_count, aes(x = reorder(Protocol, -Frequency), y = Frequency)) +
  geom_bar(stat = "identity", fill = "blue") +
  geom_text(aes(label = Frequency), vjust = -0.5, size = 3) +
  labs(title = "Rozkład protokołów", x = "Protokół", y = "Częstotliwość") +
  scale_y_continuous(labels = label_comma()) +
  facet_wrap(~Group, scales = "free_y") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

##############################################################################################################
```



Podstawowe parametry danych sieciowych

W tym kroku utworzona zostaje nowa zmienna *time_analysis*. Sumuje ona łączną liczbę pakietów, które przechwycone zostały w każdej z minut.

```
##############################################################################################
# Podstawowe parametry
##############################################################################################

time_analysis <- dane %>%
  group_by(Time = floor_date(Time, "minute")) %>%
  summarise(Packets = n())

srednia <- mean(time_analysis$Packets)
mediana <- median(time_analysis$Packets)
wariancja <- var(time_analysis$Packets)
odchylenie_std <- sd(time_analysis$Packets)
skosnosc <- skewness(time_analysis$Packets)
kurt <- kurtosis(time_analysis$Packets)

cat("Średnia:", srednia, "\nMediana:", mediana, "\nWariancja:", wariancja,
    "\nOdchylenie_std:", odchylenie_std, "\nSkośność:", skosnosc, "\nKurtoza:", kurt, "\n")

##############################################################################################
```

Ponadto przy pomocy polecenia *cat()* wypisane zostały podstawowe parametry nowej zmiennej, takie jak średnia, mediana, wariancja, odchylenie standardowe, skośność oraz kurtoza.

```
> cat("Średnia:", srednia, "\nMediana:", mediana, "\nWariancja:", wariancja,"\nOdchylenie_std:",odchylenie_std,
+     "\nSkośność:", skosnosc, "\nKurtoza:", kurt, "\n")
Średnia: 18768.38
Mediana: 18608
Wariancja: 216952802
Odchylenie_std: 14729.32
Skośność: 0.5995784
Kurtoza: 2.920103
```

Średnia liczba pakietów wynosi 18768 pakietów na minutę, co jest porównywalne do typowego obciążenia sieci.

Mediana, która wynosi 18608 pakietów, wskazuje, że połowa z analizowanych minut miała liczbę pakietów poniżej tej wartości, a połowa powyżej. Jest ona bardzo stabilnym wskaźnikiem.

Wysoka wariancja oraz odchylenie standardowe oznaczają dużą zmienność w liczbie pakietów między kolejnymi minutami, co oznacza, że przepływ pakietów jest mocno zróżnicowany.

Dodatnia skośność informuje, że rozkład jest lekko przesunięty w stronę wyższych wartości, co oznacza więcej minut z mniejszymi przepływami.

Dodatnia wartość kurtozy - wynosząca 2,92 – wskazuje, że rozkład jest bardziej szpiczasty niż rozkład normalny, co sugeruje obecność skrajnych wartości.

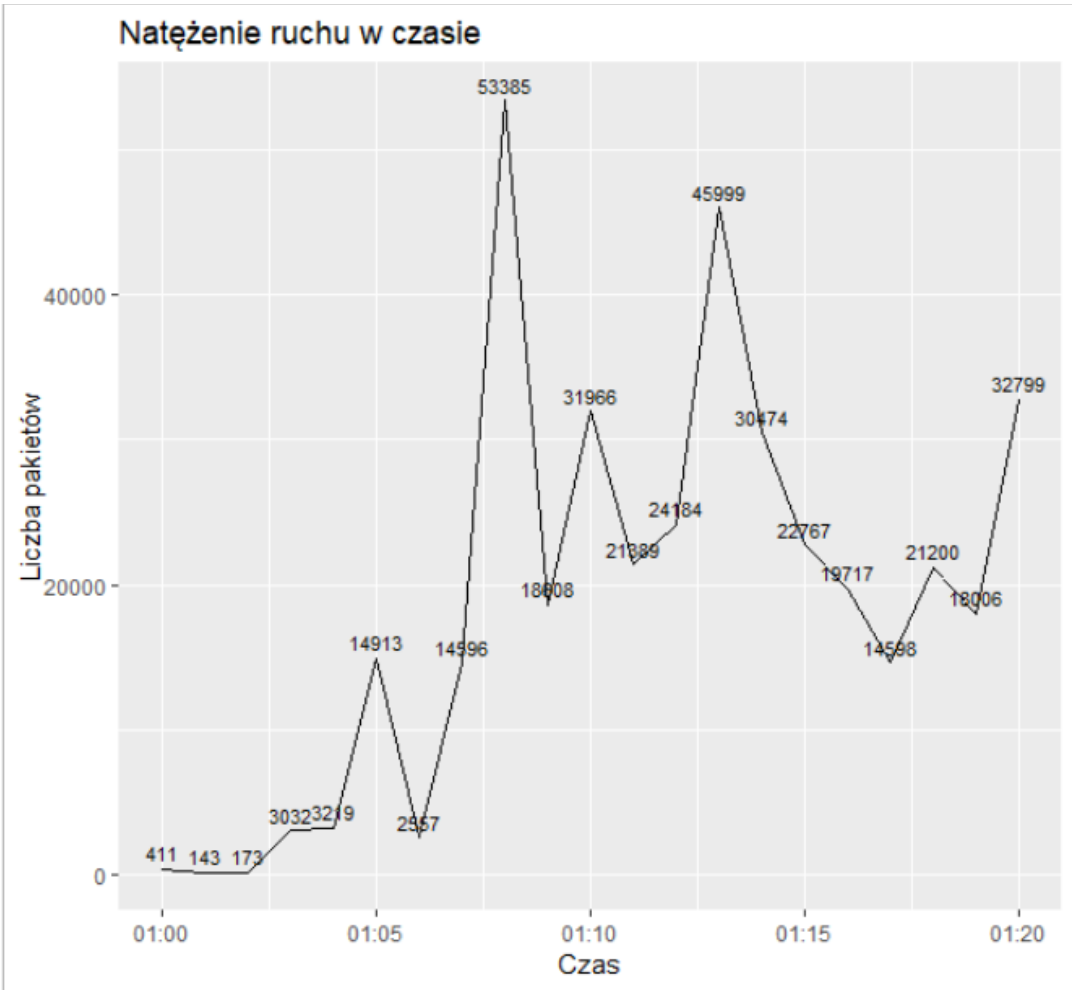
Natężenie ruchu w czasie

Stworzony został również wykres prezentujący natężenie ruchu w czasie. Dla każdej minuty została również dopisana łączna liczba pakietów przechwyconych w czasie jej trwania.

```
##############################################################################
# Natężenie ruchu w czasie
##############################################################################

ggplot(time_analysis, aes(x = Time, y = Packets)) +
  geom_line(color = "black") +
  labs(title = "Natężenie ruchu w czasie", x = "Czas", y = "Liczba pakietów") +
  geom_text(aes(label = Packets),
            vjust = -0.5,
            size = 3)

##############################################################################
```

Po wywołaniu powyższego polecenia, otrzymujemy wykres przedstawiający natężenie ruchu w czasie z podpisaną liczbą pakietów dla każdej z minut.

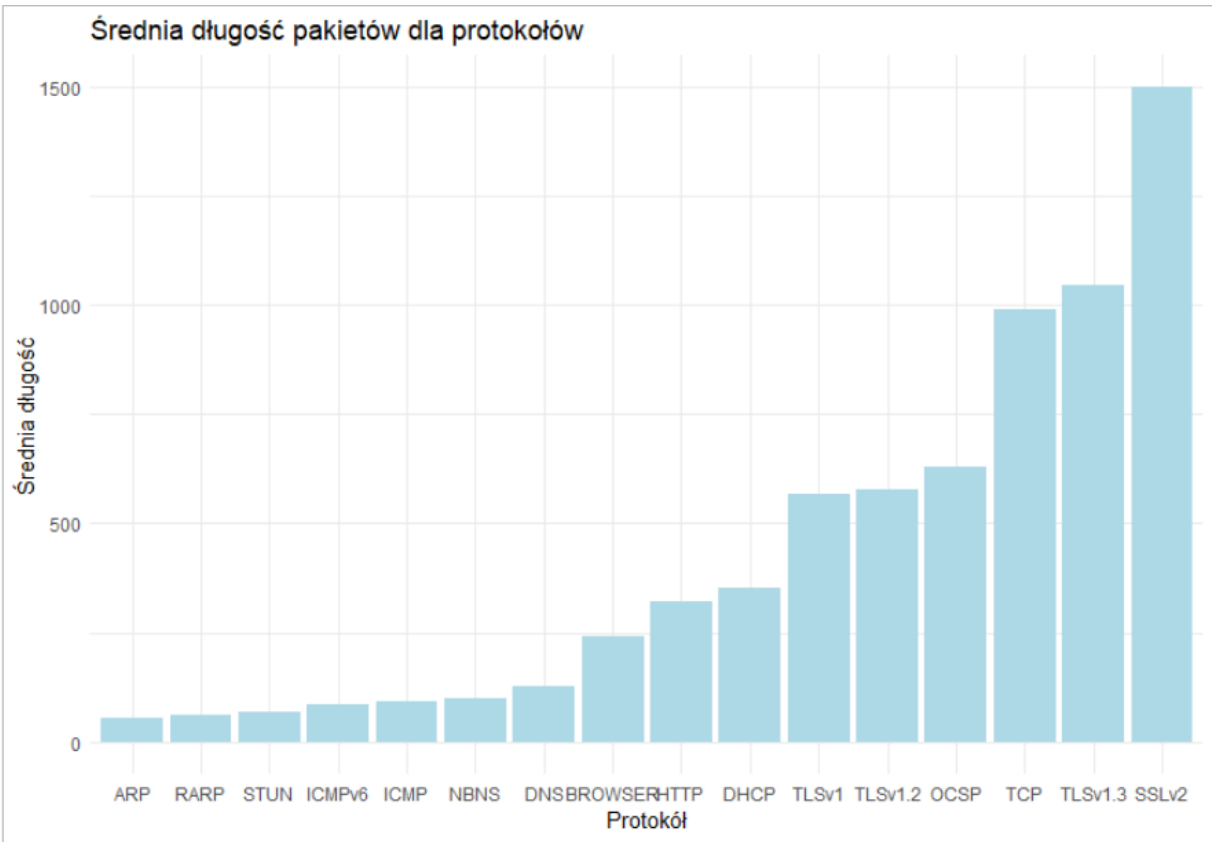
Dane pokazują, że liczba pakietów zmieniała się dynamicznie, z dwoma zauważalnymi szczytami aktywności. Tego rodzaju wahania w liczbie pakietów mogą być wynikiem różnych czynników, takich jak np. obciążenie serwerów.



Średnia długość pakietów dla protokołów

Utworzony został również wykres słupkowy, który przedstawia średnią długość pakietów dla każdego z protokołów.

```
##############################################################################################################
# Średnia długość pakietów dla protokołów
##############################################################################################################

protocol_summary <- dane %>%
  group_by(Protocol) %>%
  summarise(Avg_Length = mean(Length, na.rm = TRUE), .groups = "drop")

ggplot(protocol_summary, aes(x = reorder(Protocol, Avg_Length), y = Avg_Length)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(title = "Średnia długość pakietów dla protokołów", x = "Protokół", y = "Średnia długość") +
  theme_minimal()

##############################################################################################################
```



Różne protokoły mają różną średnią długość pakietu. Największą średnią długość pakietów możemy zaobserwować dla protokołu SSLv2. Protokoły, które zajmują się kontrolą sieci lub diagnostyką (takie jak DNS, ICMP, RARP) mają mniejsze średnie długości, podczas gdy protokoły związane z bezpieczeństwem (SSL, TLS) oraz transferem danych (HTTP, TCP) generują większe pakiety.

Wykrywanie anomalii

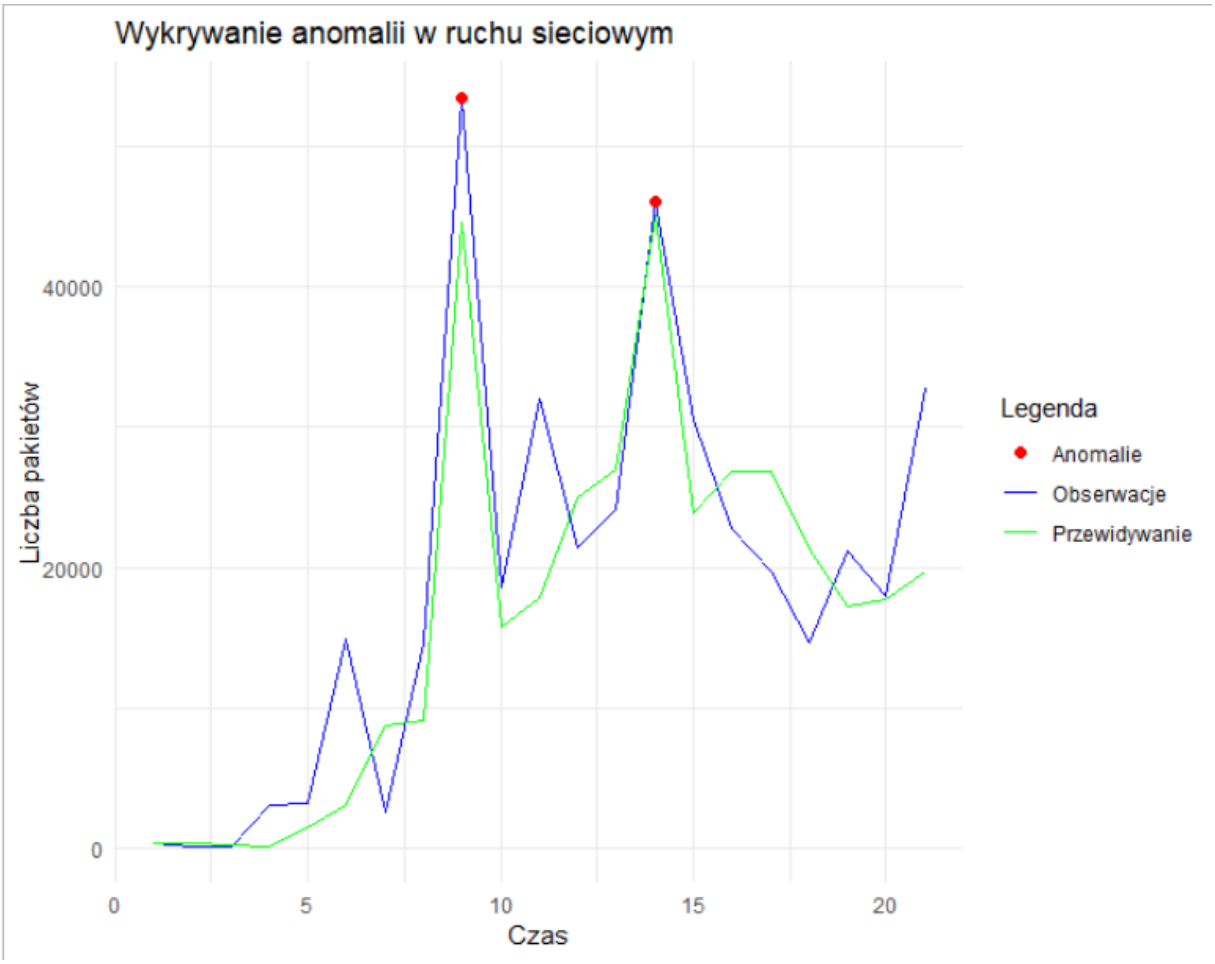
W tym rozdziale zaprezentowane zostanie cztery sposoby wykrywania anomalii dla analizowanych danych.

Sposób 1 - z użyciem pakietu do wykrywania wartości odstających w szeregach czasowych

Do pierwszego sposobu, wykorzystana zostanie biblioteka *tsoutliers*. Metoda ta wykorzystuje zaawansowane algorytmy statystyczne do wykrywania nietypowych wzorców w szeregach czasowych. Biblioteka *tsoutliers* jest szczególnie przydatna w analizie danych szeregów czasowych, gdzie może wychwycić subtelne anomalie.

```
##########################################################################################################

# Sposób 1 - z użyciem pakietu do wykrywania wartości odstających w szeregach czasowych
traffic_ts <- ts(time_analysis$Packets, frequency = 60)

anomalies <- tso(traffic_ts)


observed <- as.numeric(traffic_ts)
fitted <- as.numeric(fitted(anomalies$fit))
time_index <- seq_along(observed)

anomalies_indices <- anomalies$outliers$ind
anomaly_data <- data.frame(
  Time = time_index,
  Observed = observed,
  Fitted = c(fitted, rep(NA, length(observed) - length(fitted))),
  Anomalies = ifelse(time_index %in% anomalies_indices, "Anomaly", "Normal")
)


ggplot(anomaly_data, aes(x = Time)) +
  geom_line(aes(y = Observed, color = "Obserwacje")) +
  geom_line(aes(y = Fitted, color = "Przewidywanie")) +
  geom_point(
    data = anomaly_data %>% filter(Anomalies == "Anomaly"),
    aes(y = Observed, color = "Anomalie"), size = 2, shape = 16
  ) +
  scale_color_manual(
    values = c("Obserwacje" = "blue", "Przewidywanie" = "green", "Anomalie" = "red"),
    name = "Legenda"
  ) +
  labs(
    title = "Wykrywanie anomalii w ruchu sieciowym",
    x = "Czas",
    y = "Liczba pakietów"
  ) +
  theme_minimal()

##########################################################################################################
```

Dzięki wykonaniu powyższych poleceń, otrzymujemy poniższy wykres, który przestawia ruch sieciowy. Ponadto zaznaczone na zielono zostało również przewidywanie, a czerwonymi kropkami zaznaczono występujące anomalie.



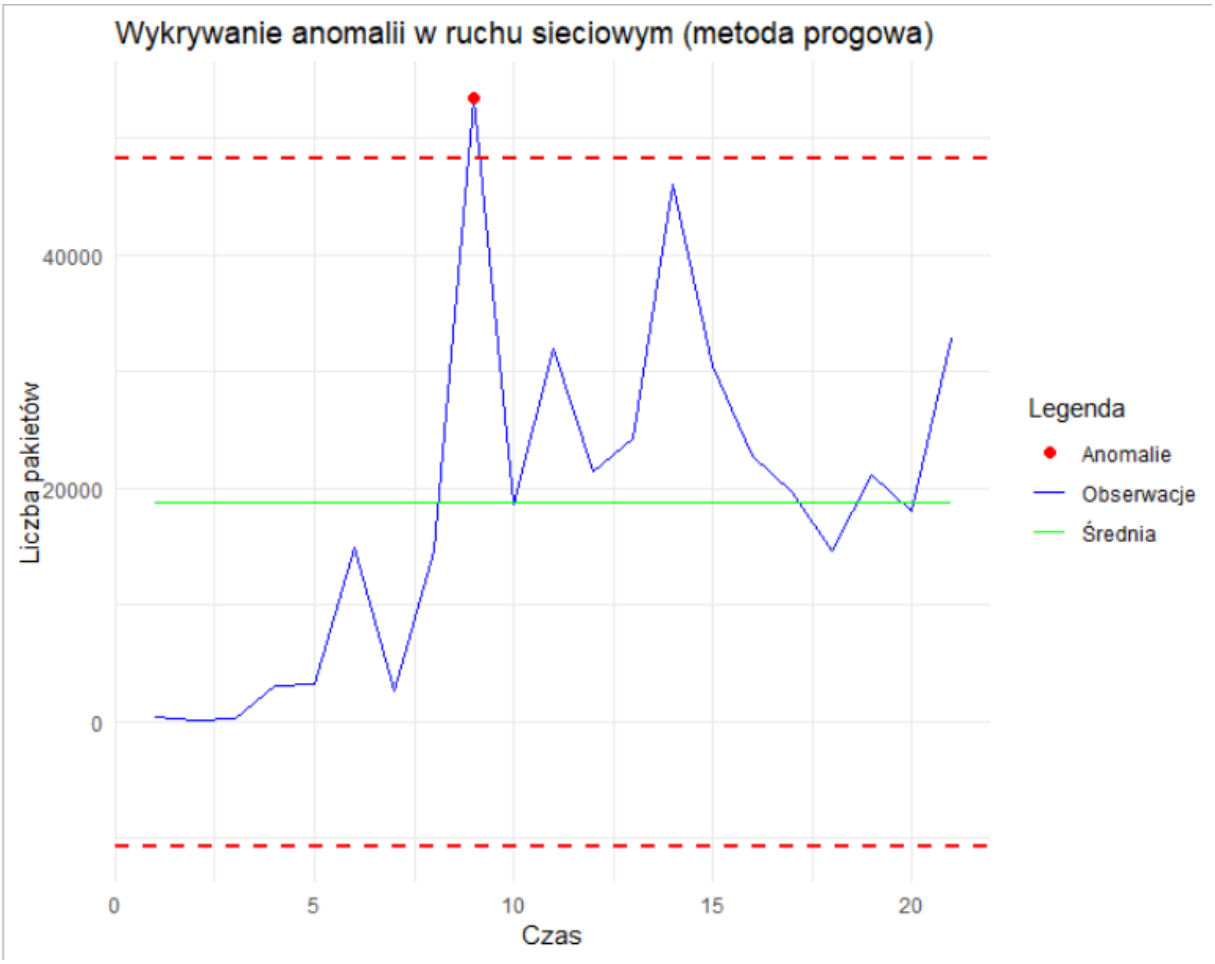
Metoda *tsoutliers* jest specjalnie zaprojektowana do analizy szeregów czasowych. Algorytmy *tsoutliers* identyfikują anomalie poprzez analizę wzorców w danych szeregów czasowych i wykrywanie nietypowych punktów. Jest ona skuteczna w identyfikacji subtelnych anomalii w szeregach czasowych, takich jak nagłe zmiany trendów, sezonowe odstępstwa lub niespodziewane skoki wartości.

Sposób 2 - metoda progowa

Drugim sposobem wykrywania anomalii jest metoda progowa. W przypadku metody progowej, anomalie są wykrywane jako odchylenia od ustalonych granic wartości średniej.

```r
# Sposób 2 - metoda progowa
mean_observed <- mean(observed, na.rm = TRUE)
sd_observed <- sd(observed, na.rm = TRUE)

upper_threshold <- mean_observed + 2*sd_observed
lower_threshold <- mean_observed - 2*sd_observed


anomaly_data <- anomaly_data %>%
  mutate(
    Threshold_Anomaly = ifelse(Observed > upper_threshold | Observed < lower_threshold, "Anomaly", "Normal")
  )

mean_pred <- mean(anomaly_data$Observed, na.rm = TRUE)

# Dodanie przewidywanych wartości (średnia)
anomaly_data <- anomaly_data %>%
  mutate(
    Predicted = mean_pred
  )
```

```r
# Wykres z metodą progową i przewidywaniem
ggplot(anomaly_data, aes(x = Time)) +
  # Linia dla obserwacji
  geom_line(aes(y = Observed, color = "Obserwacje"), size = 0.5) +

  # Linia dla przewidywań (średnia)
  geom_line(aes(y = Predicted, color = "Średnia"), linetype = 1, size = 0.5) +

  # Linia progowa - górny próg
  geom_hline(yintercept = upper_threshold, linetype = "dashed", color = "red", size = 0.8) +

  # Linia progowa - dolny próg
  geom_hline(yintercept = lower_threshold, linetype = "dashed", color = "red", size = 0.8) +

  # Punkty dla anomalii
  geom_point(
    data = anomaly_data %>% filter(Threshold_Anomaly == "Anomaly"),
    aes(y = Observed, color = "Anomalie"), size = 2, shape = 16
  ) +

  # Ustalenie kolorów w legendzie
  scale_color_manual(
    values = c("Obserwacje" = "blue", "Anomalie" = "red", "Średnia" = "green"),
    name = "Legenda"
  ) +

  # Opis wykresu
  labs(
    title = "Wykrywanie anomalii w ruchu sieciowym (metoda progowa)",
    x = "Czas",
    y = "Liczba pakietów"
  ) +
  theme_minimal()
```

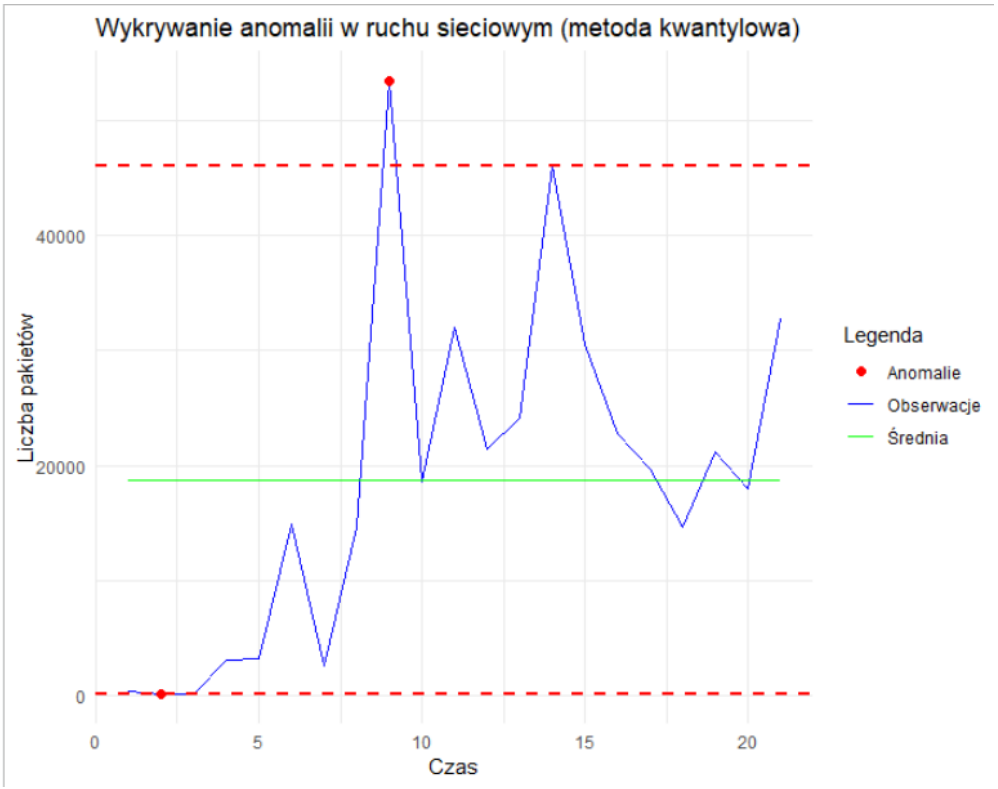


Jedyna anomalia występuje w minucie 9, co sugeruje, że w tym czasie wystąpił znaczny wzrost liczby pakietów, który przekroczył przyjęty próg. Pozostałe minuty nie wykazują żadnych istotnych odchyleń od normy. Metoda progowa okazała się skuteczna w wykrywaniu tej anomalii, ponieważ przeszła przez klasyfikację, w której obserwacje normalne były porównywane z ustalonymi granicami.

Sposób 3 - metoda kwantylowa

Trzecim sposobem wykrywania anomalii jest metoda kwantylowa. Anomalie są identyfikowane jako wartości wychodzące poza ustalone kwantyle rozkładu obserwacji. Metoda kwantylowa jest bardziej adaptacyjna niż metoda progowa, dzięki czemu może lepiej radzić sobie z dynamicznie zmieniającymi się danymi.

```
################################################################################################################

# Sposób 3 - metoda kwantylowa
prog_min <- quantile(time_analysis$Packets, 0.05, na.rm = TRUE)  # Dolny próg (10%)
prog_max <- quantile(time_analysis$Packets, 0.95, na.rm = TRUE)  # Górny próg (90%)

anomaly_data <- anomaly_data %>%
  mutate(
    Quantile_Anomaly = ifelse(Observed < prog_min | Observed > prog_max, "Anomaly", "Normal")
  )


ggplot(anomaly_data, aes(x = Time)) +
  geom_line(aes(y = Observed, color = "Obserwacje")) +
  geom_line(aes(y = Predicted, color = "Średnia"), linetype = 1, size = 0.5) +
  geom_hline(yintercept = prog_min, linetype = "dashed", color = "red", size = 0.8) +
  geom_hline(yintercept = prog_max, linetype = "dashed", color = "red", size = 0.8) +
  geom_point(
    data = anomaly_data %>% filter(Quantile_Anomaly == "Anomaly"),
    aes(y = Observed, color = "Anomalie"), size = 2, shape = 16
  ) +
  scale_color_manual(
    values = c("Obserwacje" = "blue", "Anomalie" = "red", "Średnia" = "green"),
    name = "Legenda"
  ) +
  labs(
    title = "Wykrywanie anomalii w ruchu sieciowym (metoda kwantylowa)",
    x = "Czas",
    y = "Liczba pakietów"
  ) +
  theme_minimal()

################################################################################################################
```



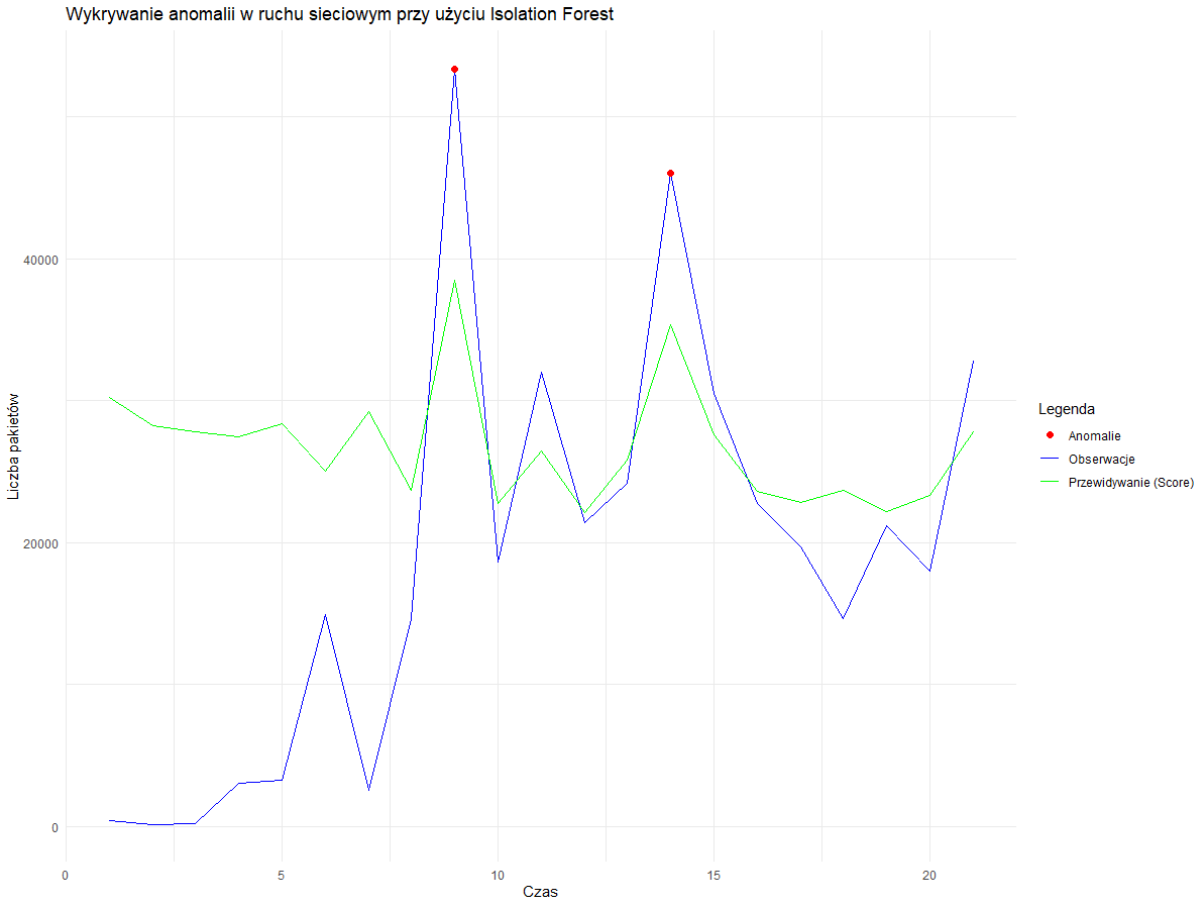
Tylko 2 minuty - druga i czternasta, uznane zostały za anomalie, co świadczy o tym, że reszta obserwacji w analizowanej próbce wykazuje stabilny przepływ pakietów.

Sposób 4 - metoda isolation forest

Ostatnim z opisanych sposobów jest metoda *isolation* forest. Metoda ta polega na izolowaniu anomalii poprzez tworzenie losowych podziałów w danych. *Isolation Forest* jest skuteczna w wykrywaniu odizolowanych punktów anomalii i jest odporna na duże zbiory danych.

```
#########################################################################################################

#Sposób 4 - metoda isolation forest

model_data <- data.frame(Packets = time_analysis$Packets)

model <- isolation.forest(model_data, ndim = 1)

scores <- predict(model, model_data, type = "score")

anomaly_threshold <- 0.6
anomaly_data <- data.frame(
  Time = seq_along(time_analysis$Packets),
  Observed = time_analysis$Packets,
  Anomalies = ifelse(scores > anomaly_threshold, "Anomaly", "Normal"),
  Score = scores
)

ggplot(anomaly_data, aes(x = Time)) +
  geom_line(aes(y = Observed, color = "Obserwacje"), size = 1) +
  geom_line(aes(y = Score * max(Observed), color = "Przewidywanie (Score)"), linetype = 1, size = 1) +
  geom_point(
    data = anomaly_data %>% filter(Anomalies == "Anomaly"),
    aes(y = Observed, color = "Anomalie"), size = 2, shape = 16
  ) +
  scale_color_manual(
    values = c("Obserwacje" = "blue", "Anomalie" = "red", "Przewidywanie (Score)" = "green"),
    name = "Legenda"
  ) +
  labs(
    title = "Wykrywanie anomalii w ruchu sieciowym przy użyciu Isolation Forest",
    x = "Czas",
    y = "Liczba pakietów"
  ) +
  theme_minimal()

#########################################################################################################
```

Isolation Forest jest algorytmem, który izoluje dane przez tworzenie losowych podziałów. W przewidywaniu anomalii, algorytm ten jest trenowany na danych historycznych, aby nauczyć się wzorców normalnego zachowania. Następnie, nowe dane są analizowane pod kątem odstępstw od tych wzorców. Anomalie są identyfikowane na podstawie tego, jak szybko dane punkty są izolowane w strukturze drzewa. Algorytm ten jest efektywny w wykrywaniu pojedynczych, odizolowanych anomalii.



Minuty dziewiąta oraz czternasta zostały zakwalifikowane jako anomalie ze względu na wysokie liczby pakietów o wartościach powyżej 45 oraz 50 tysięcy. Pozostałe wartości są raczej normalne, z drobnymi niestabilnościami w aktywności sieciowej, ale nie na tyle dużymi, by wskazywać na problemy.

Heatmapy

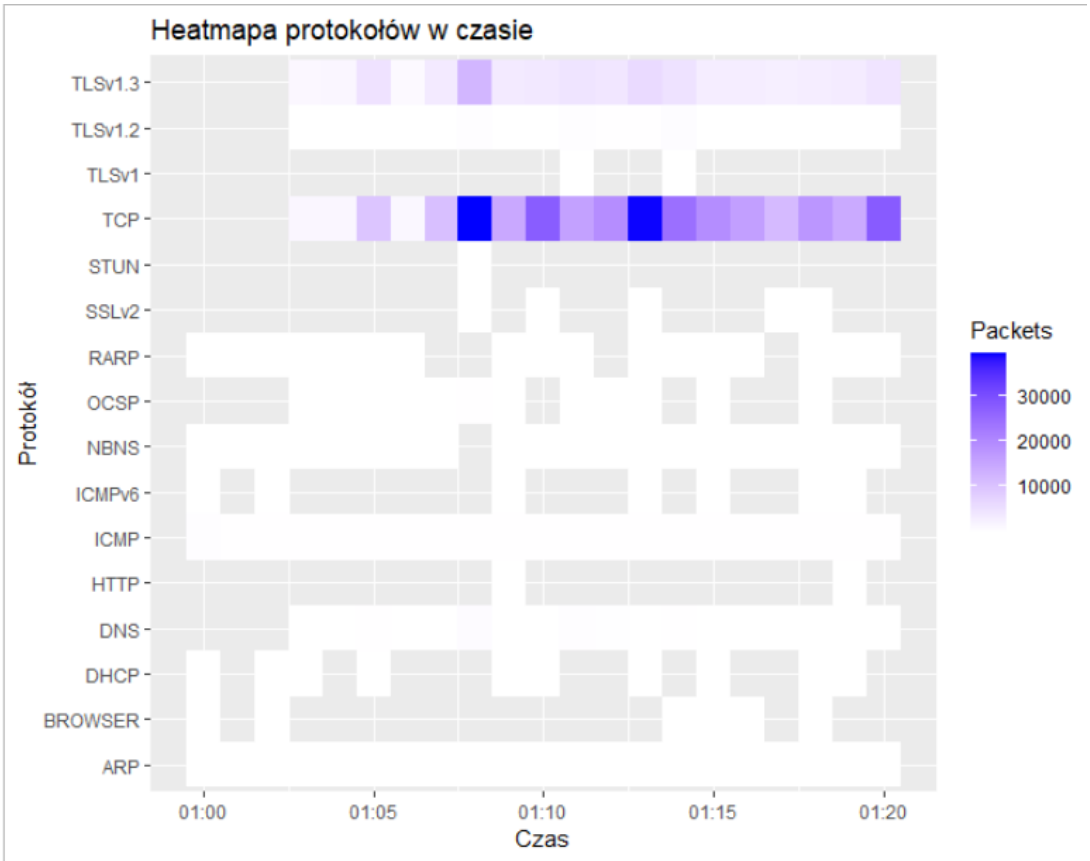
Kolejnym krokiem będzie wygenerowanie heatmap – map ciepła. Heatmapa to graficzna forma służąca wizualizowaniu danych, przy pomocy kolorów na grafie. Kolory te reprezentują w tym przypadku częstość występowania elementów w odniesieniu do czasu, bądź do innej kolumny.

Heatmapa protokołów w czasie

Pierwszą z wygenerowanych heatmap, jest heatmapa protokołów w czasie. Dzięki niej dowiadujemy się jakie protokoły były najczęściej przechwytywane w danej minucie.

```
##############################################################################
# Heatmapa
##############################################################################

#Heatmapa protokołów w czasie
heatmap_data <- dane %>%
  group_by(Time = floor_date(Time, "minute"), Protocol) %>%
  summarise(Packets = n())

ggplot(heatmap_data, aes(x = Time, y = Protocol, fill = Packets)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Heatmapa protokołów w czasie", x = "Czas", y = "Protokół")

##############################################################################
```

Na wygenerowanej heatmapie zauważyć można kolor szary, biały oraz odcienie niebieskiego. Kolor szary oznacza brak przechwyconych protokołów w danej minucie, biały oznacza bardzo małą liczbę, natomiast im kolor jest bliższy niebieskiemu, tym więcej protokołów zostało w tym czasie zaobserwowanych.

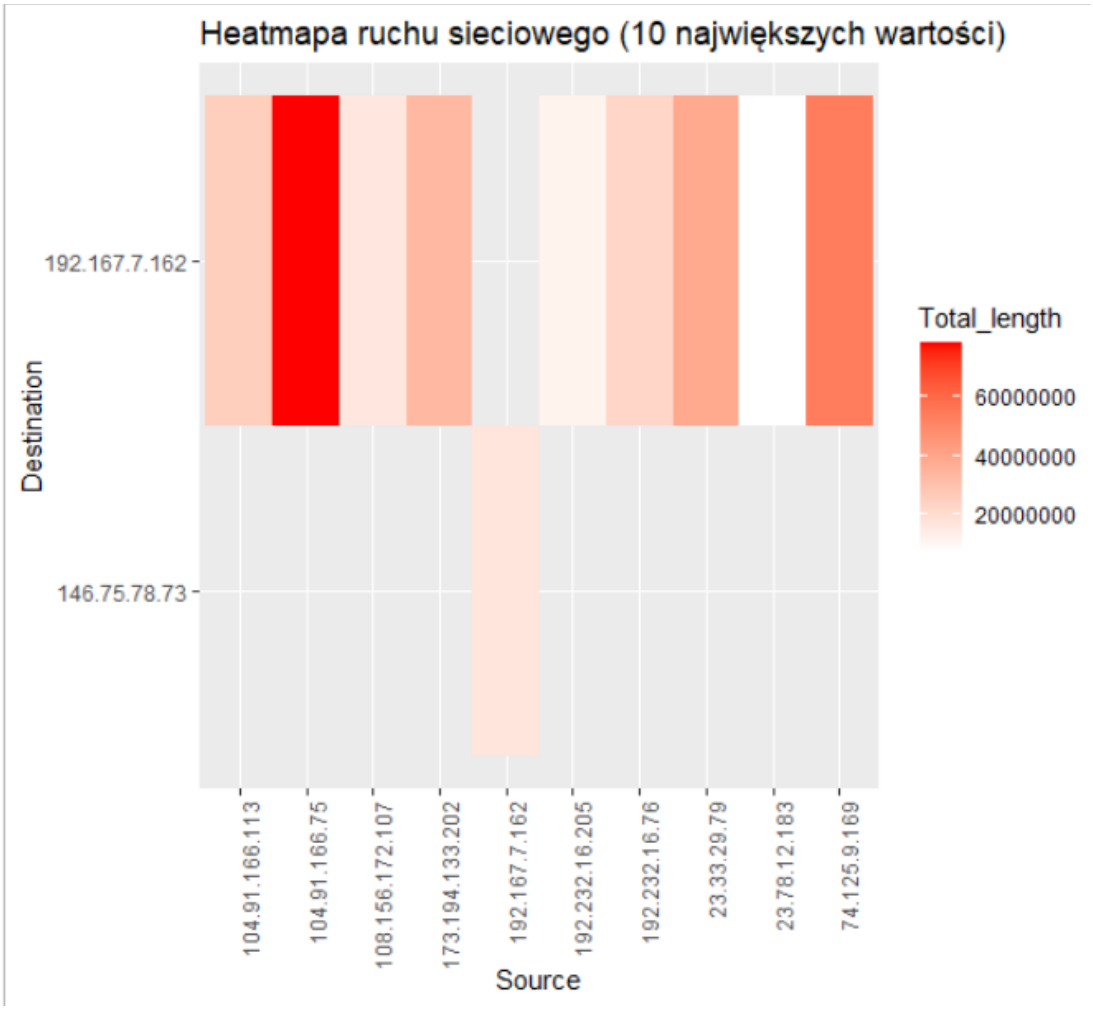


Heatmapa ruchu sieciowego

Druga heatmapa skupiać się będzie na analizie ruchu sieciowego, a dokładniej na źródłach oraz celach przesyłanych pakietów. Ze względu na to, że w danych występuje bardzo dużo unikalnych wartości IP, mapa ciepła została okrojona do dziesięciu najczęściej występujących,

```
##############################################################################

# Heatmapa ruchu sieciowego
heatmap_data <- dane %>%
  group_by(Source, Destination) %>%
  summarise(Total_Length = sum(Length)) %>%
  ungroup() %>%
  arrange(desc(Total_Length)) %>%
  slice_head(n = 10)

ggplot(heatmap_data, aes(x = Source, y = Destination, fill = Total_Length)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  labs(title = "Heatmapa ruchu sieciowego (10 największych wartości)",
       x = "Source",
       y = "Destination",
       fill = "Total_length") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

##############################################################################
```

Na poniższej mapie ciepła możemy zauważyć, że najczęstszym celem pakietów był adres *192.167.7.162*, który otrzymał najwięcej pakietów z adresu *104.91.166.75*.

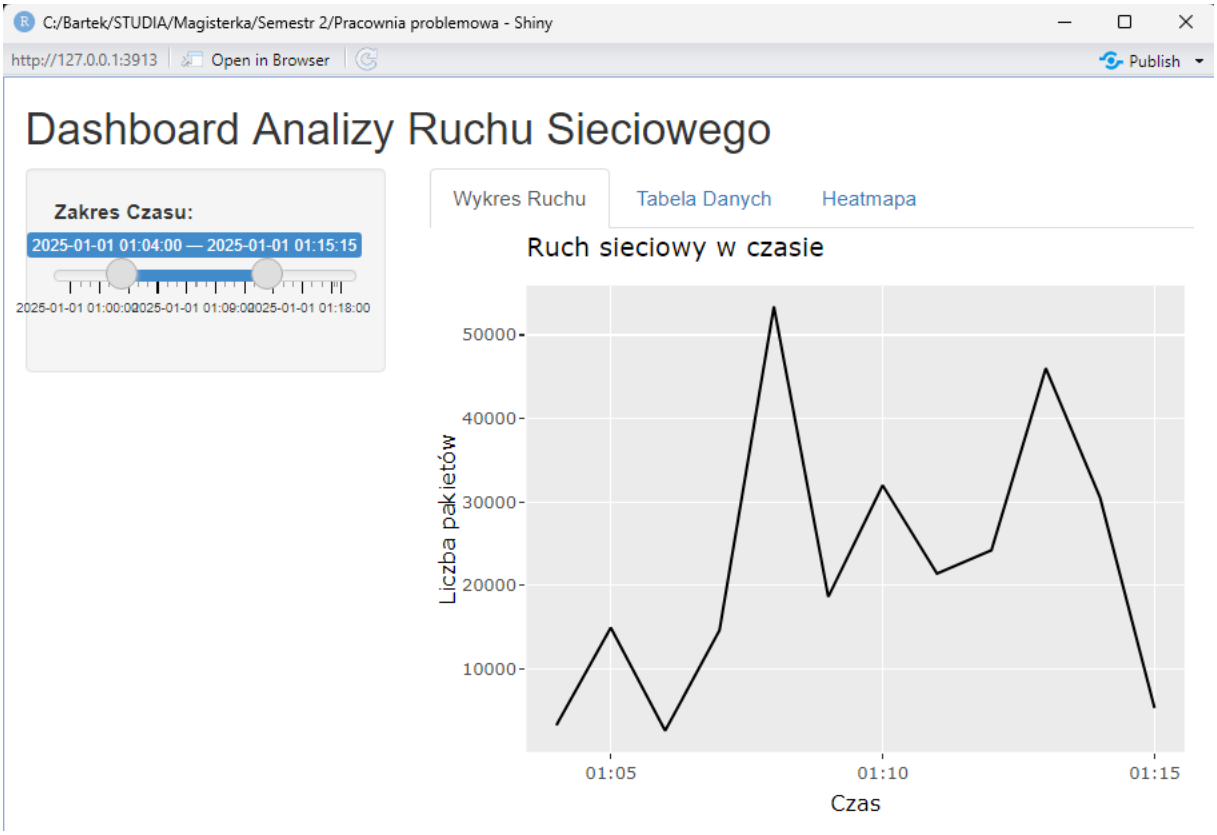


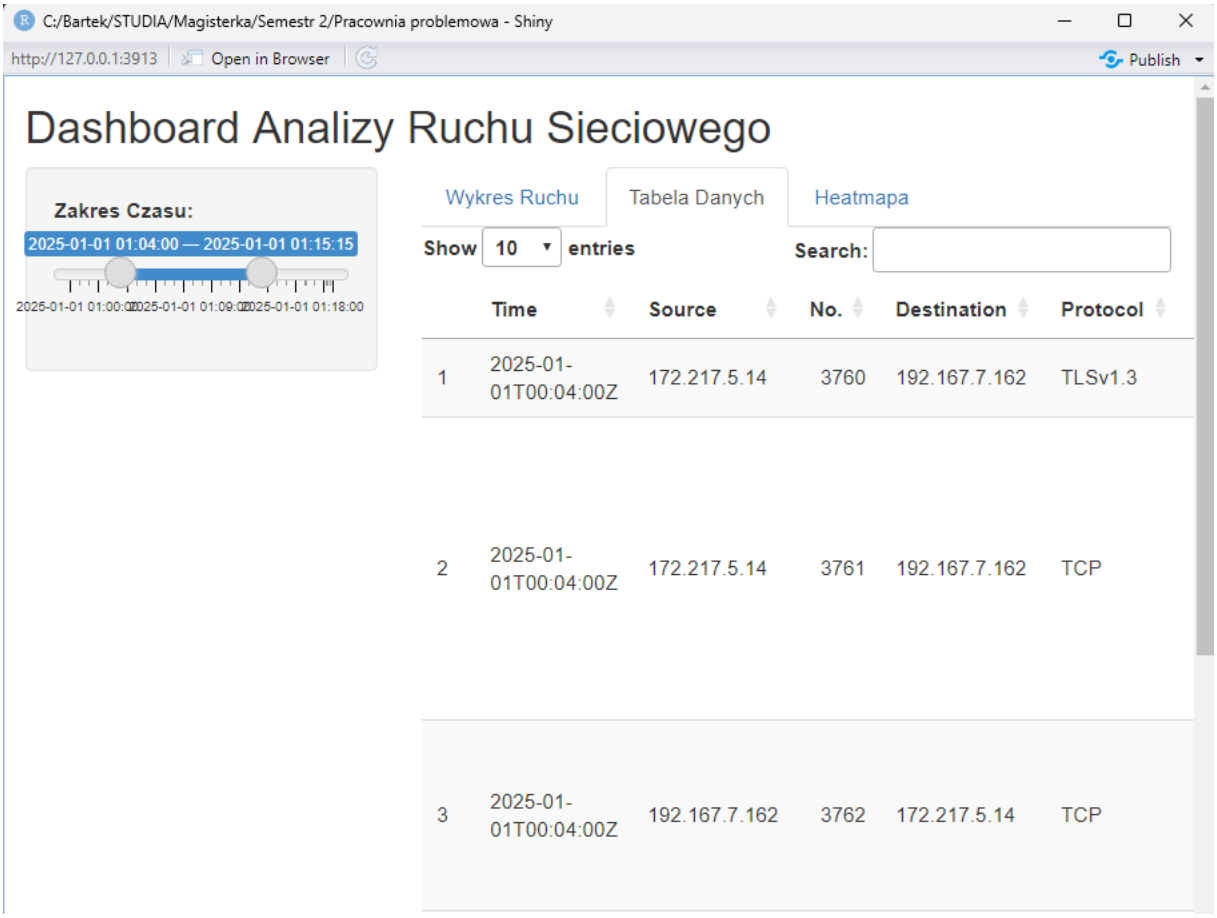
Dashboard

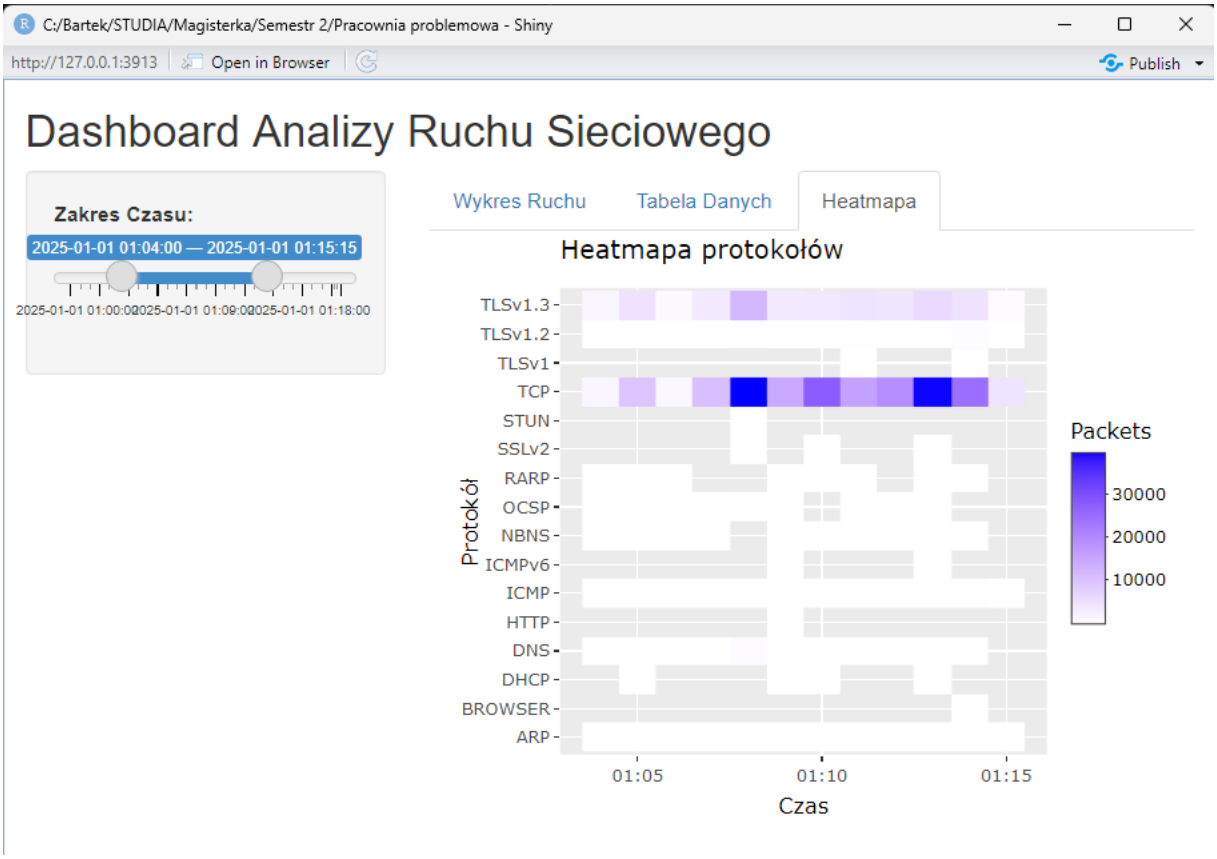
Ostatnim krokiem, który zostanie zrealizowany w tym projekcie jest utworzenie dashboardu. Pozwoli on na analizę ruchu sieciowego, w której skład wejdzie wykres ruchu, tabela danych oraz heatmapa. Powyższe informacje będą określane na podstawie określonego manualnie czasu. W tym celu napisany został poniższy skrypt.

```
##########################################################################################################

# Dashboard
ui <- fluidPage(
  titlePanel("Dashboard Analizy Ruchu Sieciowego"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("timeRange", "Zakres Czasu:",
                  min = min(dane$Time), max = max(dane$Time),
                  value = c(min(dane$Time), max(dane$Time)), timeFormat = "%Y-%m-%d %H:%M:%S")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Wykres Ruchu", plotlyOutput("timePlot")),
        tabPanel("Tabela Danych", dataTableOutput("dataTable")),
        tabPanel("Heatmapa", plotlyOutput("heatmap"))
      )
    )
  )
)

server <- function(input, output) {

  filteredData <- reactive({
    dane %>%
      filter(Time >= input$timeRange[1], Time <= input$timeRange[2])
  })

  output$timePlot <- renderPlotly({
    data <- filteredData() %>%
      group_by(Time = floor_date(Time, "minute")) %>%
      summarise(Packets = n(), .groups = "drop")

    ggplotly(
      ggplot(data, aes(x = Time, y = Packets)) +
        geom_line(color = "black") +
        labs(title = "Ruch sieciowy w czasie", x = "Czas", y = "Liczba pakietów")
    )
  })

  output$dataTable <- DT::renderDT({
    DT::datatable(filteredData())
  })

  output$heatmap <- renderPlotly({
    data <- filteredData() %>%
      group_by(Time = floor_date(Time, "minute"), Protocol) %>%
      summarise(Packets = n(), .groups = "drop")

    ggplotly(
      ggplot(data, aes(x = Time, y = Protocol, fill = Packets)) +
        geom_tile() +
        scale_fill_gradient(low = "white", high = "blue") +
        labs(title = "Heatmapa protokołów", x = "Czas", y = "Protokół")
    )
  })
}

shinyApp(ui = ui, server = server)

##########################################################################################################
```

W rezultacie otrzymany został poniższy dashboard, którego działanie zostanie zobrazowane w formie zrzutów ekranu.







Na powyższych zrzutach ekranu zaprezentowana została analiza pakietów przechwyconych od czwartej do piętnastej minuty.

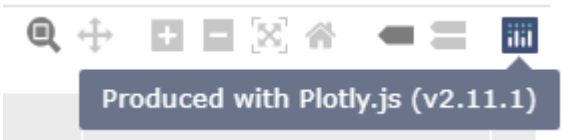
Dzięki powyższemu dashboardowi możliwe jest wybranie dowolnych ram czasowych, które chcemy przeanalizować. Po wybraniu momentu, który nas interesuje, otrzymujemy gotowe dane na jego temat.

W pierwszej zakładce otrzymujemy wykres ruchu sieciowego w czasie z podziałem na minuty. Obrazuje on dokładnie ile pakietów zostało przechwyconych w danej minucie.

W następnej sekcji widoczna jest tabela, która przedstawia zbiór wszystkich pakietów, których ruchy zostały zarejestrowane w określonym okresie. Umieszczone zostały tam najważniejsze kolumny z początkowego zbioru danych.

Ostatnia sekcja przedstawia heatmapę podobną do tej, utworzonej w poprzednim rozdziale. Określa ona więc natężenie protokołów w danej minucie pomiaru.

Otrzymane w dashboardzie podglądy możemy również modyfikować według uznania. Pozwala nam na to umieszczone w rogu menu, dzięki któremu wykresy możemy skalować.



Podsumowanie

W ramach tego projektu, zrealizowane zostały pierwotne założenia oraz osiągnięty zamierzony cel. Dane zostały zebrane, wyeksportowane, a następnie wczytane do środowiska, na którym pracowaliśmy – RStudio. Obliczone zostały podstawowe parametry statystyczne, a następnie dane zostały przetransformowane do szeregu czasowego, co dało większe możliwości w dalszej analizie.

Dzięki wizualizacji, przedstawione zostało wiele wykresów, które w estetyczny i prosty sposób obrazują obliczone wartości. Powstały wykresy takie jak:

- Macierz korelacji,
- Rozkład długości pakietów,
- Rozkład protokołów,
- Natężenie ruchu w czasie,
- Średnia długość pakietów dla protokołów.

Następnie, głównym celem projektu było wykrycie występujących anomalii. W tym celu skorzystaliśmy z czterech różnych sposobów:

- Sposób 1 - z użyciem pakietu do wykrywania wartości odstających w szeregach czasowych,
- Sposób 2 - metoda progowa,
- Sposób 3 - metoda kwantylowa,
- Sposób 4 - metoda isolation forest.

Dzięki temu wykryte oraz zobrazowane w formie wykresów zostały występujące anomalie.

W kolejnym rozdziale wygenerowane zostały również dwie mapy ciepła:

- Heatmapa protokołów w czasie,
- Heatmapa ruchu sieciowego.

Wygenerowane heatmapy pomogły zwizualizować natężenie konkretnych typów protokołów w czasie oraz najczęściej występujące źródła oraz cele przechwytywanych pakietów.

Ostatnim krokiem było wygenerowanie interaktywnego dashboardu, który pozwolił na szybki i intuicyjny podgląd analiz w ustalonym obrębie czasu.

Projekt objął wiele różnych operacji w środowisku programistycznym RStudio. Zrealizowane zostało wiele wizualizacji oraz obliczonych wiele parametrów statystycznych. Znajomość języka była bardzo przydatna, natomiast część realizowanych celów wymagała odpowiedniego przygotowania oraz opanowania nowych umiejętności.