

Statystyka matematyczna i ekonometria

Projekt - Auto Sprzedam

Autorzy

Anna Kalitka (255445)
Antoni Bezdziety (249327)

Projekt zaliczeniowy
Laboratorium, lato 2023/24

I - Opis bazy danych

Wybrana została baza danych **AutoSprzedam**. Baza pochodzi z zasobów ePortalu.

Zawartość bazy danych

Baza danych składa się z 41034 rekordów, zawierających szczegółowe dane dotyczące sprzedaży samochodów m.in.

Kolumna	Typ danych	Opis
NrOferty	int	Numer oferty sprzedaży samochodu.
CenaPLN	string	Cena samochodu wyrażona w polskich złotych (PLN).
KM	int	Liczba koni mechanicznych samochodu.
Marka	string	Marka samochodu.
Model	string	Model samochodu.
LiczbaDrzwi	string	Liczba drzwi w samochodzie.
PojemnoscSkokowa	int	Pojemność skokowa silnika wyrażona w centymetrach sześciennych (cm ³).
PrzebiegKm	int	Przebieg samochodu wyrażony w kilometrach.
RodzajPaliwa	string	Rodzaj paliwa używanego przez samochód (benzyna, diesel, hybryda, elektryczny itp.).
RokProdukcji	int	Rok produkcji samochodu.
Kolor	string	Kolor samochodu.
KrajPochodzenia	string	Kraj pochodzenia samochodu.
PojazdUszkodzony	string	Informacja czy pojazd jest uszkodzony (Tak/Nie).
SkrzyniaBiegow	string	Typ skrzyni biegów w samochodzie (manualna, automatyczna).

Prezentacja bazy danych

```
# Wczytanie bazy danych
database = read.csv("../database/AutoSprzedam.dat", sep = "\t", dec = ',');

cat("Liczba rekordów:", nrow(database), ", liczba kolumn:", ncol(database), "\n")

## Liczba rekordów: 41034 , liczba kolumn: 14
N = 5;
cat("-", "Przykład ", N, "pierwszych rekordów.\n")
```

- Przykład 5 pierwszych rekordów.

```
head(database, N)
```

```
##   NrOferty CenaPLN  KM      Marka      Model LiczbaDrzwi PojemnoscSkokowa
## 1         1  27900 150      Opel      Vectra         4/5             1900
## 2         2  28000 116    Toyota Corolla Verso         4/5             2000
## 3         3  25500 150      Skoda      Superb         4/5             1781
## 4         4  29900 109 Mercedes-Benz      A 180         2/3             1991
## 5         5  29800 207    Peugeot      607          4/5             2946
##   PrzebiegKm      RodzajPaliwa RokProdukcji      Kolor
## 1      80840 olej napędowy (diesel)      2005 czarny-metallic
## 2     166000 olej napędowy (diesel)      2004      bialy
## 3     112000      benzyna+LPG      2002 bordowy-metallic
## 4      42000 olej napędowy (diesel)      2005      czerwony
## 5     169000      benzyna      2004 granatowy-metallic
##   KrajPochodzenia PojazdUszkodzony SkrzyniaBiegow
## 1      Niemcy      Nie      manualna
## 2      Polska      Nie      manualna
## 3      Polska      Nie      manualna
## 4      Polska      Nie      manualna
## 5     Francja      Nie      manualna
```

```
cat("-", "Przykład ", N, "ostatnich rekordów.\n")
```

- Przykład 5 ostatnich rekordów.

```
tail(database, N)
```

```
##   NrOferty CenaPLN  KM      Marka      Model LiczbaDrzwi PojemnoscSkokowa
## 41030    41030 98000.00 220      Opel Insignia         4/5             1998
## 41031    41031 34924.50 184 Mercedes-Benz      S 400         4/5             3996
## 41032    41032 41175.09  70    Peugeot      308          2/3             1397
## 41033    41033 47900.00 115      Ford      C-MAX         4/5             1560
## 41034    41034 14200.00  90      Ford      Mondeo         4/5             1998
##   PrzebiegKm      RodzajPaliwa RokProdukcji      Kolor
## 41030     25500      benzyna      2010 grafitowy-metallic
## 41031    162000 olej napędowy (diesel)      2001 srebrny-metallic
## 41032      9289      benzyna      2010 srebrny-metallic
## 41033     45000 olej napędowy (diesel)      2010 czarny-metallic
## 41034    191024 olej napędowy (diesel)      2003 czarny-metallic
##   KrajPochodzenia PojazdUszkodzony SkrzyniaBiegow
## 41030      Polska      Nie      manualna
## 41031      Czechy      Nie      automatyczna
## 41032      Niemcy      Nie      manualna
```

## 41033	Belgia	Nie	manualna
## 41034	Niemcy	Nie	manualna

Przygotowanie bazy danych

Przed przejściem do dalszej pracy z bazą danych postanowiono wprowadzenie zmian w jej strukturze, których celem jest uproszczenie przyszłych prac z danymi.

Pole - *SkrzyniaBiegow*

W bazie występują 3 rodzaje typów skrzyni biegów: *półautomatyczna/sekwencyjna*, *manualna* oraz *automatyczna*. Udział typu *półautomatyczna/sekwencyjna* w całej bazie wynosi 1,5%.

W związku z niewielkim udziałem ze skrzynią *półautomatyczna/sekwencyjna*, typ ten został usunięty z bazy. Dzięki czemu możliwe jest przekształcenie kolumny *SkrzyniaBiegow* (char), na *SkrzyniaBiegowManualna* (bool).

```
gear_box_summary = database %>% group_by(database$SkrzyniaBiegow) %>% summarise(liczba = n());
gear_box_summary
```

```
## # A tibble: 3 x 2
##   `database$SkrzyniaBiegow`   liczba
##   <chr>                     <int>
## 1 automatyczna              6941
## 2 manualna                  34030
## 3 półautomatyczna/sekwencyjna    63
```

```
cat("Udział pojazdów ze skrzynią biegów typu 'półautomatyczna/sekwencyjna'",
    sum(gear_box_summary[3,2])/sum(gear_box_summary[,2])*100, "%.");
```

```
## Udział pojazdów ze skrzynią biegów typu 'półautomatyczna/sekwencyjna' 0.1535312 %.
```

```
# Remove cars with database$SkrzyniaBiegow == 'półautomatyczna/sekwencyjna'
database = database[database$SkrzyniaBiegow != "półautomatyczna/sekwencyjna",];
```

```
# Cast to logical value
```

```
database$SkrzyniaBiegow = database$SkrzyniaBiegow == "manualna";
colnames(database)[14] = "SkrzyniaBiegowManualna";
```

Pole *PojazdUszkodzony*

Domyślnym typem danych dla pola *PojazdUszkodzony* jest (char). Zmienna jest typem logicznym (TAK/NIE). W związku z powyższym kolumna *PojazdUszkodzony* została przekształcona to typu *bool*.

```
database$PojazdUszkodzony = database$PojazdUszkodzony == "Tak";
```

Pole *Liczba drzwi*

W bazie występują pojazdy tylko i wyłącznie z dwoma liczbami drzwi 2/3 oraz 4/5. W związku z powyższym można analogicznie przekształcić typ danych na typ *bool*.

```
database$LiczbaDrzwi = database$LiczbaDrzwi == "4/5";
colnames(database)[6] = "LiczbaDrzwi4/5"
```

Prezentacja bazy danych - po wprowadzonych zmianach

```
N = 5;
cat("-", "Przykład ", N, "pierwszych rekordów.\n")
```

- Przykład 5 pierwszych rekordów.

```
head(database, N)
```

```
##      NrOferty CenaPLN  KM      Marka      Model LiczbaDrzwi4/5
## 1          1   27900 150      Opel      Vectra             TRUE
## 2          2   28000 116    Toyota Corolla Verso             TRUE
## 3          3   25500 150      Skoda      Superb             TRUE
## 4          4   29900 109 Mercedes-Benz      A 180             FALSE
## 5          5   29800 207    Peugeot      607             TRUE
##      PojemnoscSkokowa PrzebiegKm      RodzajPaliwa RokProdukcji
## 1          1900      80840 olej napędowy (diesel)      2005
## 2          2000     166000 olej napędowy (diesel)      2004
## 3          1781     112000      benzyna+LPG      2002
## 4          1991     42000 olej napędowy (diesel)      2005
## 5          2946     169000      benzyna      2004
##      Kolor KrajPochodzenia PojazdUszkodzony SkrzyniaBiegowManualna
## 1   czarny-metallic      Niemcy      FALSE             TRUE
## 2          biały      Polska      FALSE             TRUE
## 3   bordowy-metallic      Polska      FALSE             TRUE
## 4          czerwony      Polska      FALSE             TRUE
## 5   granatowy-metallic      Francja      FALSE             TRUE
```

```
cat("-", "Przykład ", N, "ostatnich rekordów.\n")
```

- Przykład 5 ostatnich rekordów.

```
tail(database, N)
```

```
##      NrOferty CenaPLN  KM      Marka      Model LiczbaDrzwi4/5
## 41030      41030 98000.00 220      Opel      Insignia             TRUE
## 41031      41031 34924.50 184 Mercedes-Benz      S 400             TRUE
## 41032      41032 41175.09  70    Peugeot      308             FALSE
## 41033      41033 47900.00 115      Ford      C-MAX             TRUE
## 41034      41034 14200.00  90      Ford      Mondeo             TRUE
##      PojemnoscSkokowa PrzebiegKm      RodzajPaliwa RokProdukcji
## 41030          1998      25500      benzyna      2010
## 41031          3996     162000 olej napędowy (diesel)      2001
## 41032          1397      9289      benzyna      2010
## 41033          1560     45000 olej napędowy (diesel)      2010
## 41034          1998     191024 olej napędowy (diesel)      2003
##      Kolor KrajPochodzenia PojazdUszkodzony
## 41030   grafitowy-metallic      Polska      FALSE
## 41031   srebrny-metallic      Czechy      FALSE
## 41032   srebrny-metallic      Niemcy      FALSE
## 41033   czarny-metallic      Belgia      FALSE
## 41034   czarny-metallic      Niemcy      FALSE
##      SkrzyniaBiegowManualna
## 41030             TRUE
## 41031             FALSE
## 41032             TRUE
```

```
## 41033 TRUE
## 41034 TRUE
```

II - Wyznaczenie podstawowych statystyk

W tej sekcji przedstawione zostaną podstawowe statystyki dla wybranych zmiennych.

Statystyki dla zmiennych numerycznych

```
numerical_database = data.frame(database$CenaPLN,
                                database$KM,
                                database$PojemnoscSkokowa,
                                database$PrzebiegKm,
                                database$RokProdukcji);
numerical_statistics = data.frame(do.call(cbind, lapply(numerical_database, summary)));
numerical_statistics[nrow(numerical_statistics)+1,] = sapply(numerical_database, var);
numerical_statistics[nrow(numerical_statistics)+1,] = sapply(numerical_database, sd);
row.names(numerical_statistics)[7:8] <- c("Var", "Sd");
numerical_statistics
```

```
##      database.CenaPLN database.KM database.PojemnoscSkokowa
## Min.      1.000000e+03      1.00000      13.0000
## 1st Qu.    1.720000e+04     83.00000     1560.0000
## Median    2.600000e+04    110.00000     1896.0000
## Mean      4.084835e+04    121.00517     1916.8589
## 3rd Qu.    4.190000e+04    140.00000     1998.0000
## Max.      1.788000e+06   1400.00000     7300.0000
## Var       2.636581e+09   3589.98079    415825.4410
## Sd        5.134764e+04    59.91645     644.8453
##      database.PrzebiegKm database.RokProdukcji
## Min.              1.0      2001.000000
## 1st Qu.           80000.0      2003.000000
## Median          131000.0      2005.000000
## Mean            125329.7      2005.289864
## 3rd Qu.          170361.0      2007.000000
## Max.            2600000.0      2011.000000
## Var            5110148687.7        7.805945
## Sd              71485.3        2.793912
```

Statystyki dla zmiennych dwumianowych

```
logical_database = data.frame(database$`LiczbaDrzwi4/5`,
                              database$PojazdUszkodzony,
                              database$SkrzyniaBiegowManualna);

logical_statistics = data.frame(do.call(cbind, lapply(logical_database, summary)))[2:3,];
logical_statistics
```

```
##      database..LiczbaDrzwi4.5. database.PojazdUszkodzony
## FALSE              5301              39497
## TRUE               35670              1474
##      database.SkrzyniaBiegowManualna
## FALSE              6941
```

```
## TRUE
```

```
34030
```

Statystyki dla pozostałych zmiennych (jakościowych)

```
quality_database = data.frame(database$Marka,  
                               database$RodzajPaliwa,  
                               database$Kolor,  
                               database$KrajPochodzenia);  
  
sapply(quality_database, function (x){  
  round(sort(table(x), decreasing = TRUE) / nrow(quality_database) , 3)})
```

```
## $database.Marka
```

```
## x
```

##	Volkswagen	Ford	Opel	Renault	Audi
##	0.151	0.132	0.122	0.115	0.091
##	Peugeot	Skoda	Toyota	Mercedes-Benz	BMW
##	0.078	0.072	0.066	0.064	0.062
##	Fiat				
##	0.047				

```
##
```

```
## $database.RodzajPaliwa
```

```
## x
```

##	olej napędowy (diesel)	benzyna	benzyna+LPG
##	0.680	0.297	0.022
##	hybryda	benzyna+CNG	napęd elektryczny
##	0.001	0.001	0.000

```
##
```

```
## $database.Kolor
```

```
## x
```

##	srebrny-metallic	czarny-metallic	niebieski-metallic	szary-metallic
##	0.252	0.191	0.070	0.068
##	grafitowy-metallic	granatowy-metallic	biały	zielony-metallic
##	0.057	0.056	0.053	0.033
##	czerwony	czarny	niebieski	granatowy
##	0.031	0.029	0.027	0.025
##	złoty-metallic	srebrny	bordowy-metallic	czerwony-metallic
##	0.017	0.012	0.012	0.011
##	beżowy-metallic	brazowy-metallic	biały-metallic	złoty
##	0.009	0.009	0.007	0.005
##	zielony	szary	wisniowy-metallic	fioletowy-metallic
##	0.005	0.004	0.004	0.003
##	grafitowy	pomarańczowy-metal	bordowy	beżowy
##	0.003	0.002	0.002	0.001
##	złoty	złoty-metallic	fioletowy	brazowy
##	0.001	0.001	0.000	0.000
##	pomarańczowy	wisniowy	rozowy-metallic	rozowy
##	0.000	0.000	0.000	0.000

```
##
```

```
## $database.KrajPochodzenia
```

```
## x
```

##	Niemcy	Polska	Francja	Czechy
##	0.398	0.298	0.092	0.063
##	Belgia	Włochy	Holandia	Austria

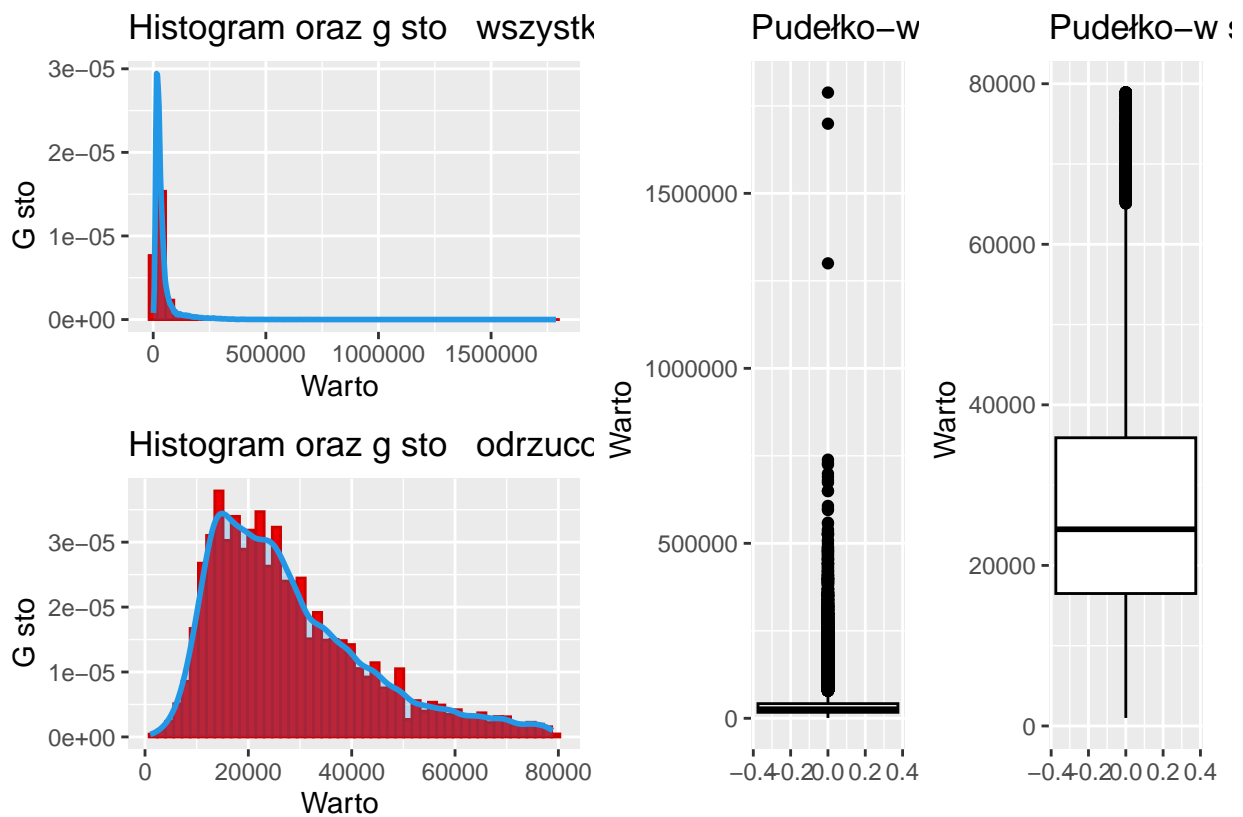
```
##          0.052          0.024          0.020          0.014
## Stany Zjednoczone    Szwajcaria    Wielka Brytania    Luksemburg
##          0.014          0.008          0.005          0.004
##          Dania        Hiszpania        Szwecja        Kanada
##          0.003          0.003          0.001          0.001
##          Słowacja     Irlandia        Norwegia        Estonia
##          0.000          0.000          0.000          0.000
##          Rumunia      Węgry          Grecja        Islandia
##          0.000          0.000          0.000          0.000
##          Rosja
##          0.000
```

III - Graficzna prezentacja danych

Wykresy dla zmiennych numerycznych

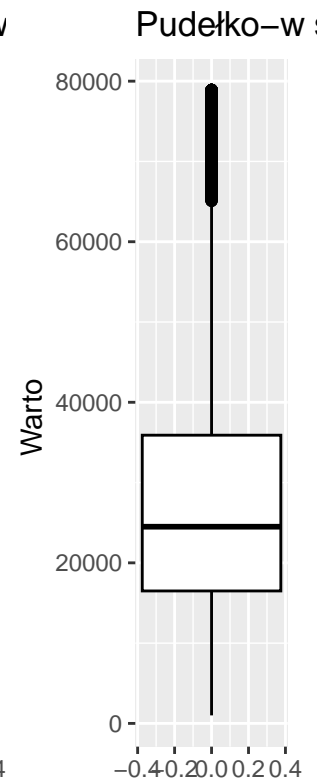
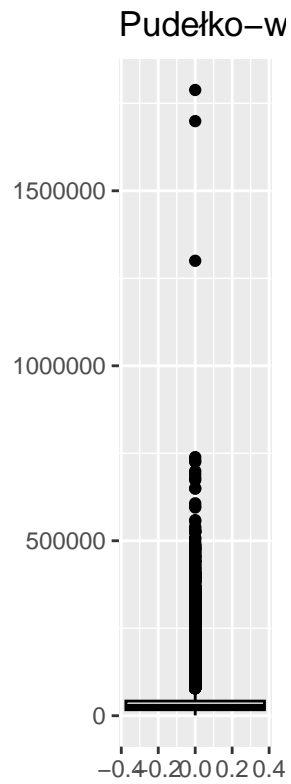
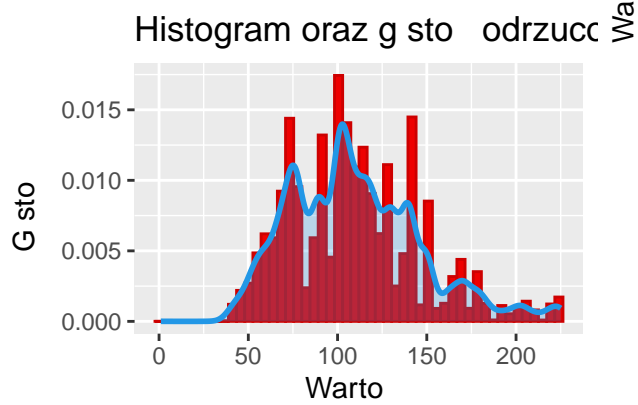
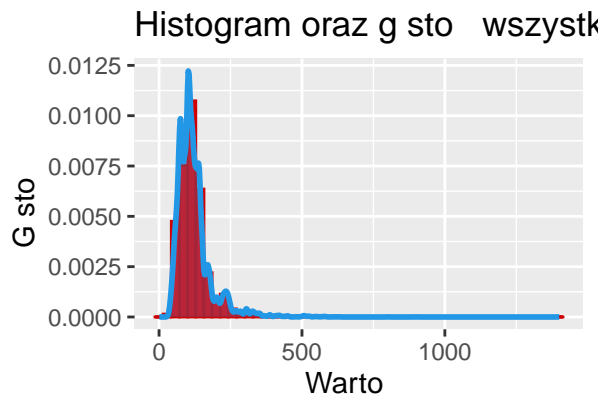
```
sapply(numerical_database, draw_numerical)
```

```
## NULL
## NULL
## NULL
## NULL
## NULL
## $database.CenaPLN
```



```
##
```

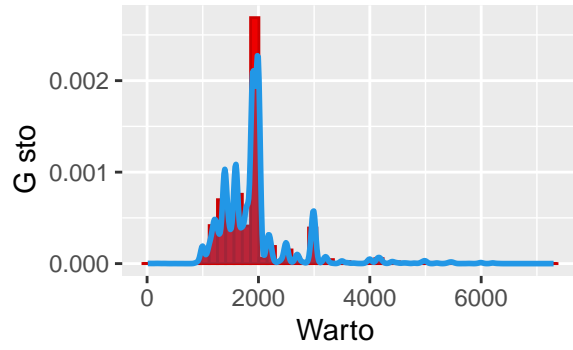
```
## $database.KM
```



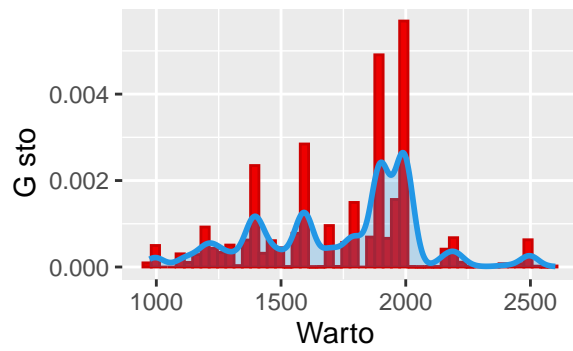
```
##
```

```
## $database.PojemnoscSkokowa
```

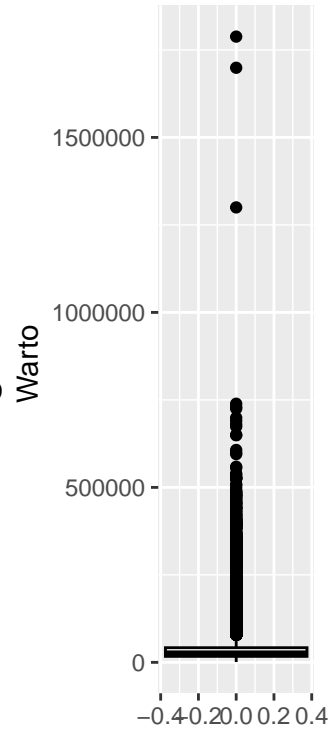

Histogram oraz g sto wszystkich



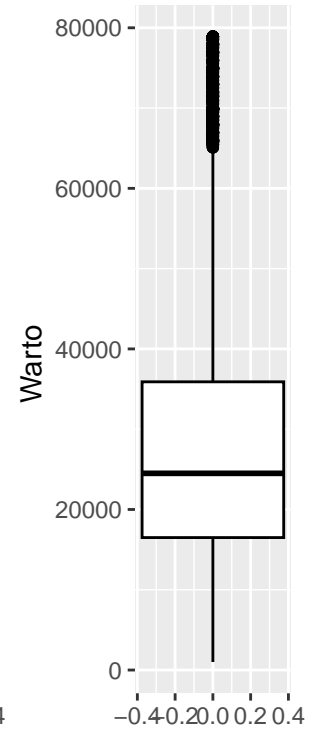
Histogram oraz g sto odrzuco



Pudełko-w

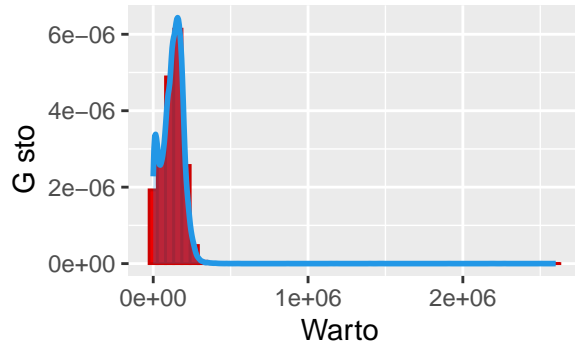


Pudełko-w s

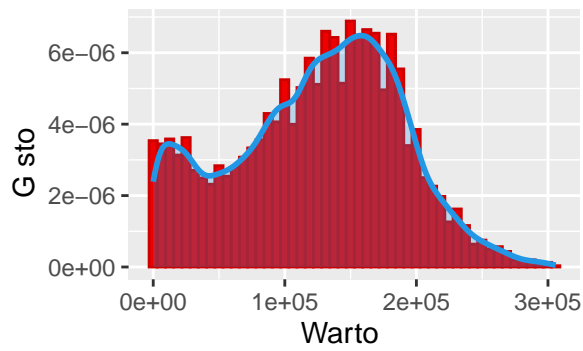


```
##
## $database.PrzebiegKm
```

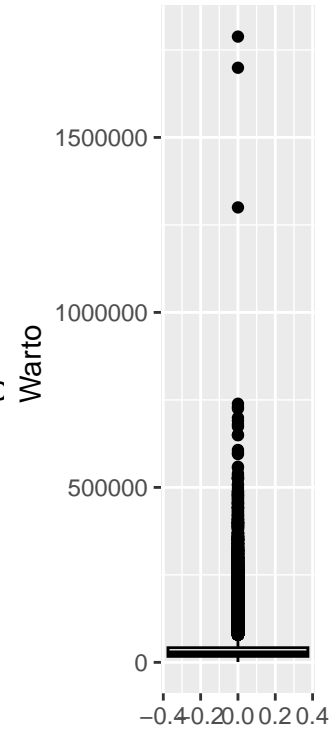
Histogram oraz g sto wszystkich



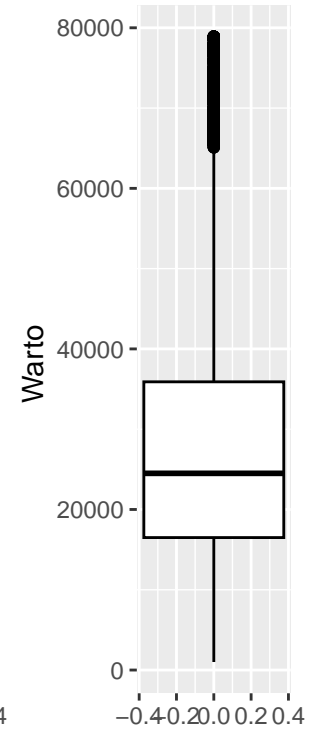
Histogram oraz g sto odrzucc



Pudełko-w

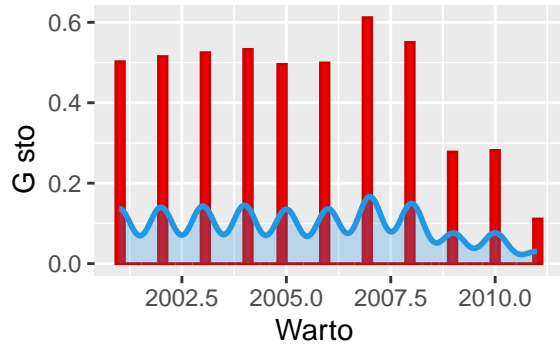


Pudełko-w :

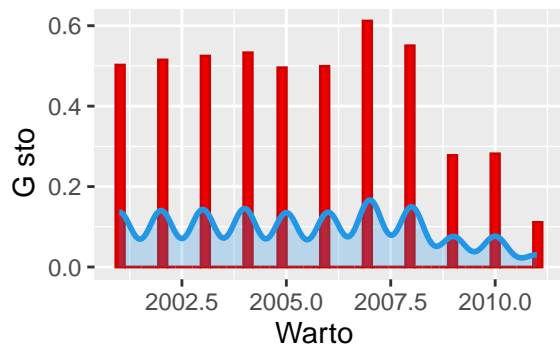


```
##
## $database.RokProdukcji
```

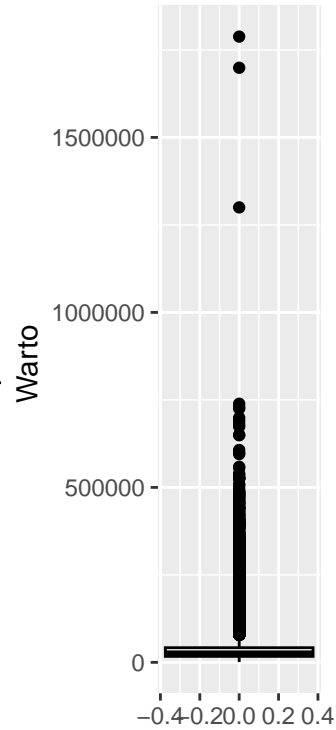
Histogram oraz g sto wszystkie



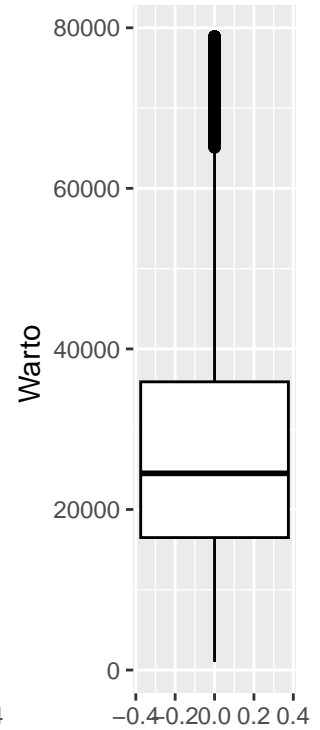
Histogram oraz g sto odrzucor



Pudełko-w



Pudełko-w s

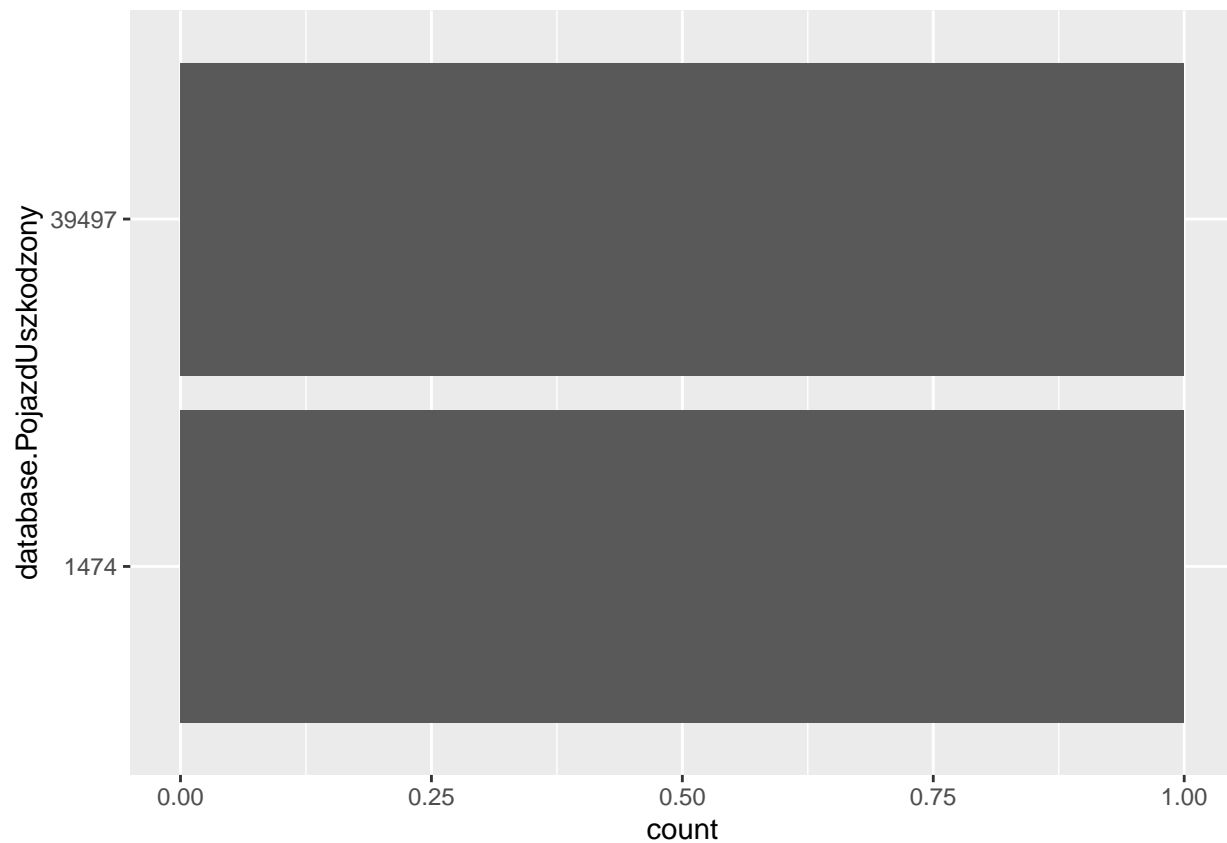


Statystyki dla zmiennych dwumianowych

logical_statistics

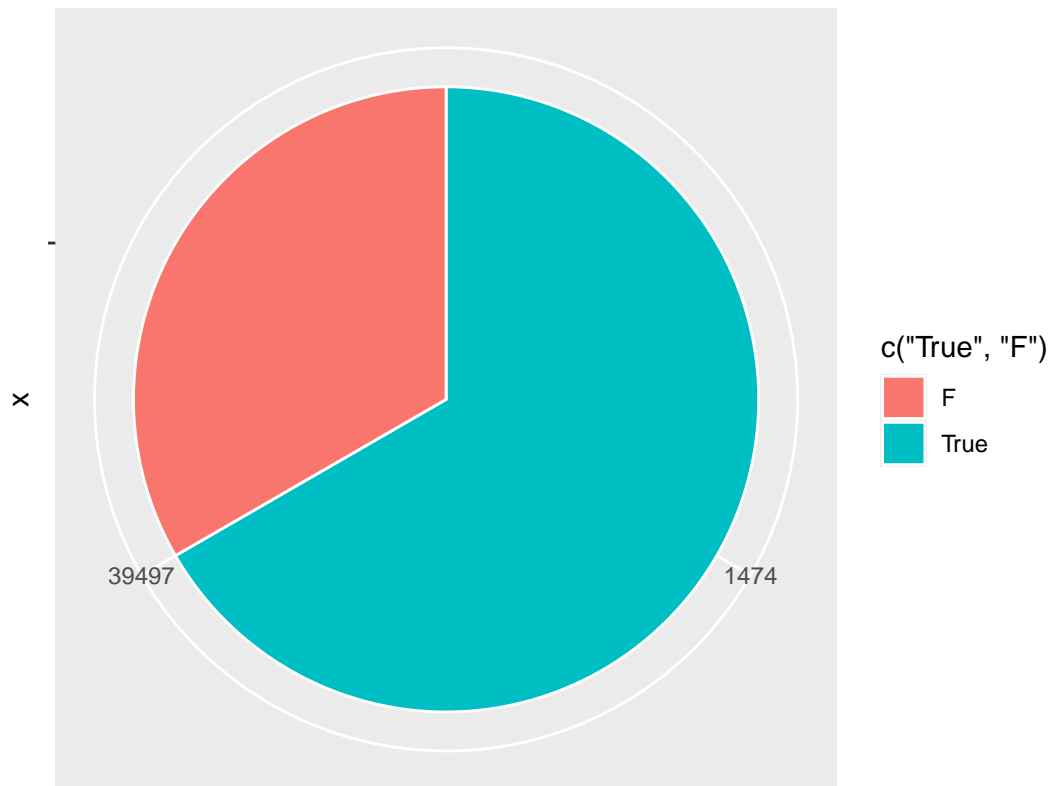
```
##      database..LiczbaDrzwi4.5. database.PojazdUszkodzony
## FALSE                    5301                    39497
## TRUE                     35670                    1474
##      database.SkrzyniaBiegowManualna
## FALSE                    6941
## TRUE                     34030
```

```
ggplot(logical_statistics) + geom_bar(aes(y=database.PojazdUszkodzony))
```



```
ggplot(logical_statistics, aes(x="", y=logical_statistics$database.PojazdUszkodzony, fill=c('True', 'False'))) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0)
```

```
## Warning: Use of `logical_statistics$database.PojazdUszkodzony` is discouraged.
## i Use `database.PojazdUszkodzony` instead.
```



logical_statistics\$database.PojazdUszkodzony

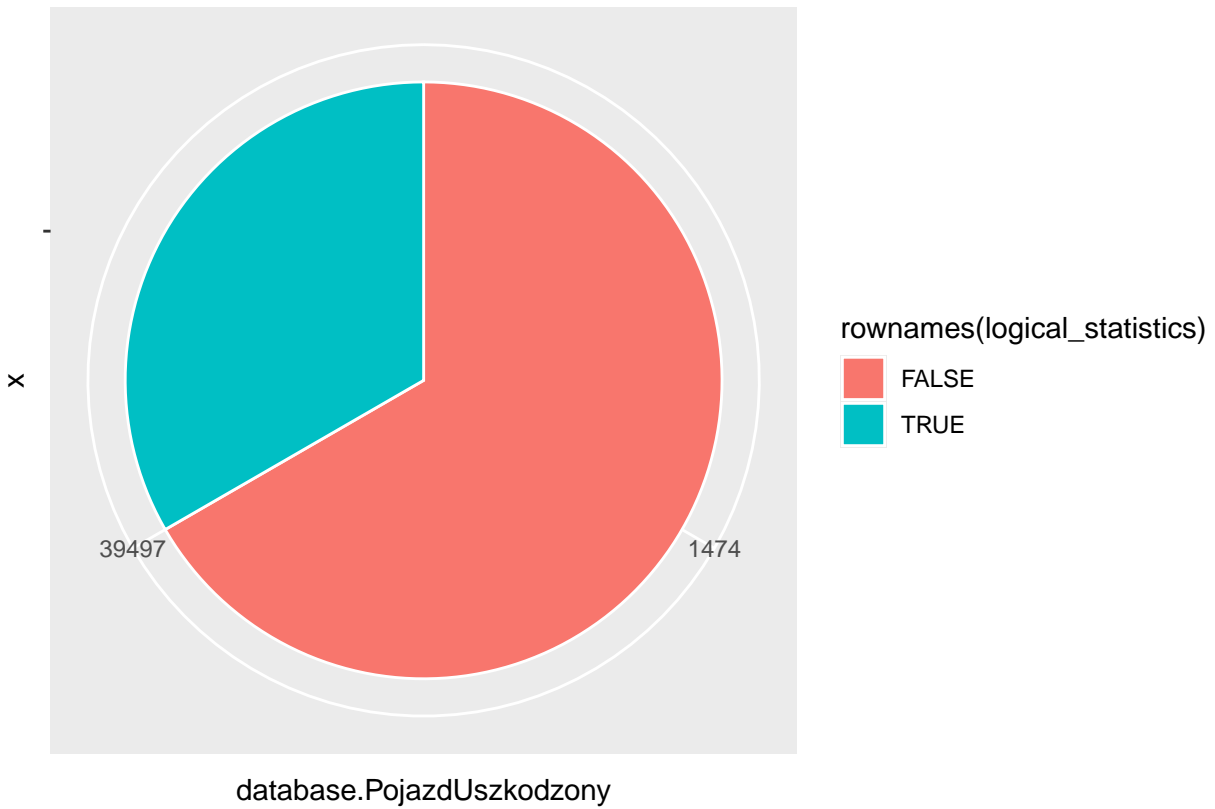
```
names(logical_statistics)
```

```
## [1] "database..LiczbaDrzwi4.5."      "database.PojazdUszkodzony"
## [3] "database.SkrzyniaBiegowManualna"
```

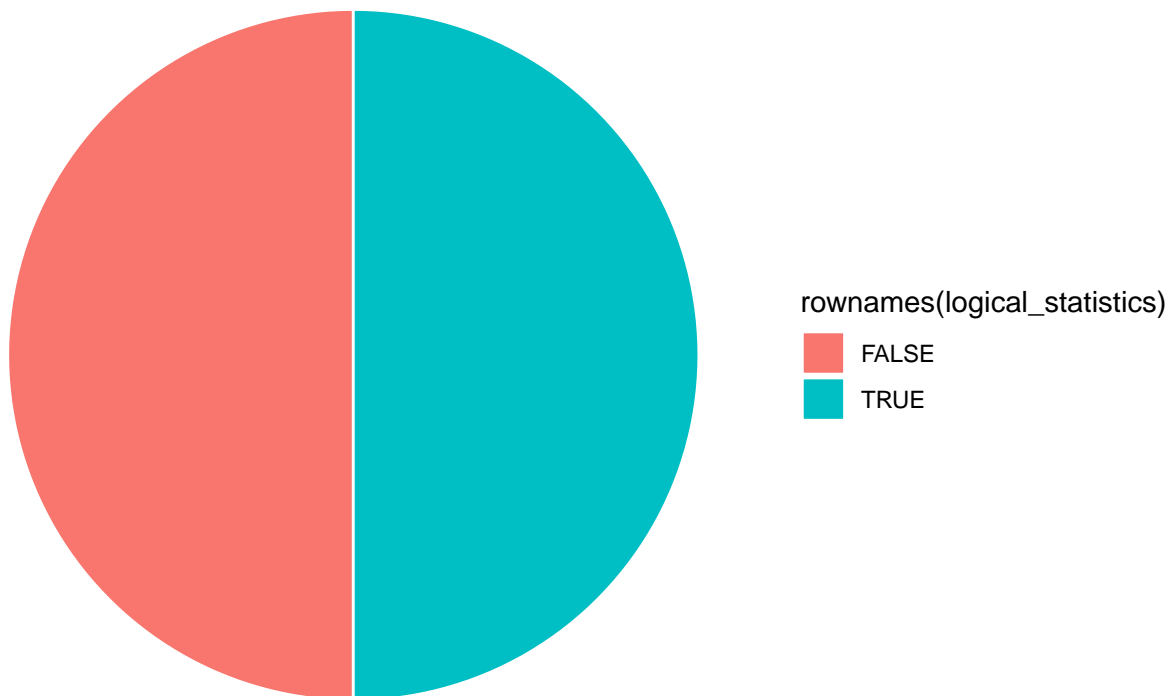
```
rownames(logical_statistics)
```

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

```
ggplot(logical_statistics, aes(x="", y=database.PojazdUszkodzony, fill=rownames(logical_statistics))) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0)
```



```
ggplot(logical_statistics, aes(x="", y="database.SkrzyniaBiegowManualna", fill=rownames(logical_statistics))) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) + theme_void();
```



```
database %>% group_by(Marka) %>% summarise(liczbaSprzedanych = n(),
                                             sredniaCena = mean(CenaPLN),
                                             medianaCena = median(CenaPLN))
```

```
## # A tibble: 11 x 4
##   Marka      liczbaSprzedanych sredniaCena medianaCena
##   <chr>          <int>          <dbl>      <dbl>
## 1 Audi           3717          86498.    52800
## 2 BMW            2543          87356.    56900
## 3 Fiat           1927          19343.    14800
## 4 Ford           5412          27515.    22900
## 5 Mercedes-Benz  2618          84583.    49900
## 6 Opel           5001          24280.    22500
## 7 Peugeot        3202          25203.    22000
## 8 Renault        4723          21089.    17800
## 9 Skoda          2935          29129.    26328
## 10 Toyota        2709          39099.    30268
## 11 Volkswagen    6184          37060.    30000
```

Dane po eliminacją danych odstających

```
summarise(database,
  srednia = mean(CenaPLN),
  mediana = median(CenaPLN),
  wariancja = var(CenaPLN),
  odchylenieStd = sd(CenaPLN),
```

```

max = max(CenaPLN),
min = min(CenaPLN),
)

```

```

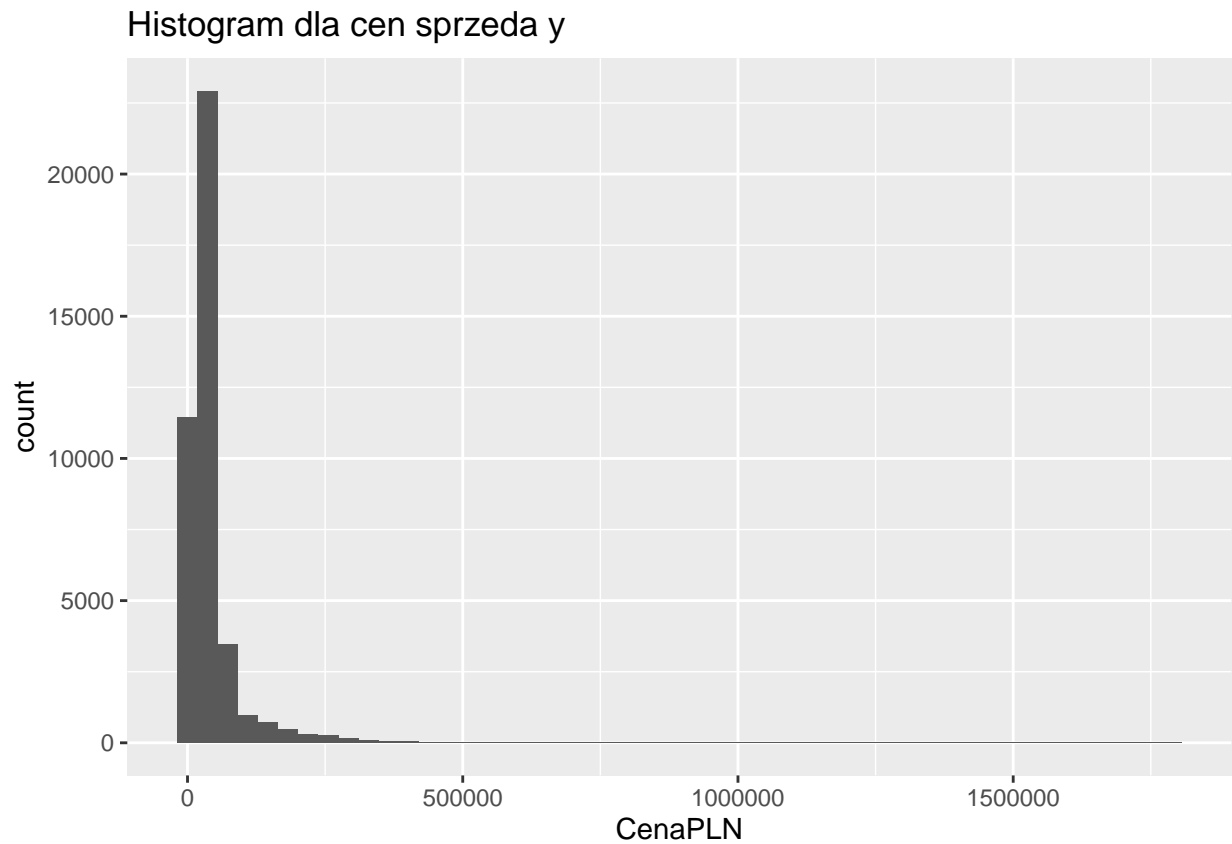
##      srednia  mediana  wariancja  odchylenieStd      max  min
## 1 40848.35    26000 2636580628      51347.64 1788000 1000

```

```

ggplot(data=database) + geom_histogram(aes(x=CenaPLN), bins = 50) + ggtitle("Histogram dla cen sprzedaży")

```

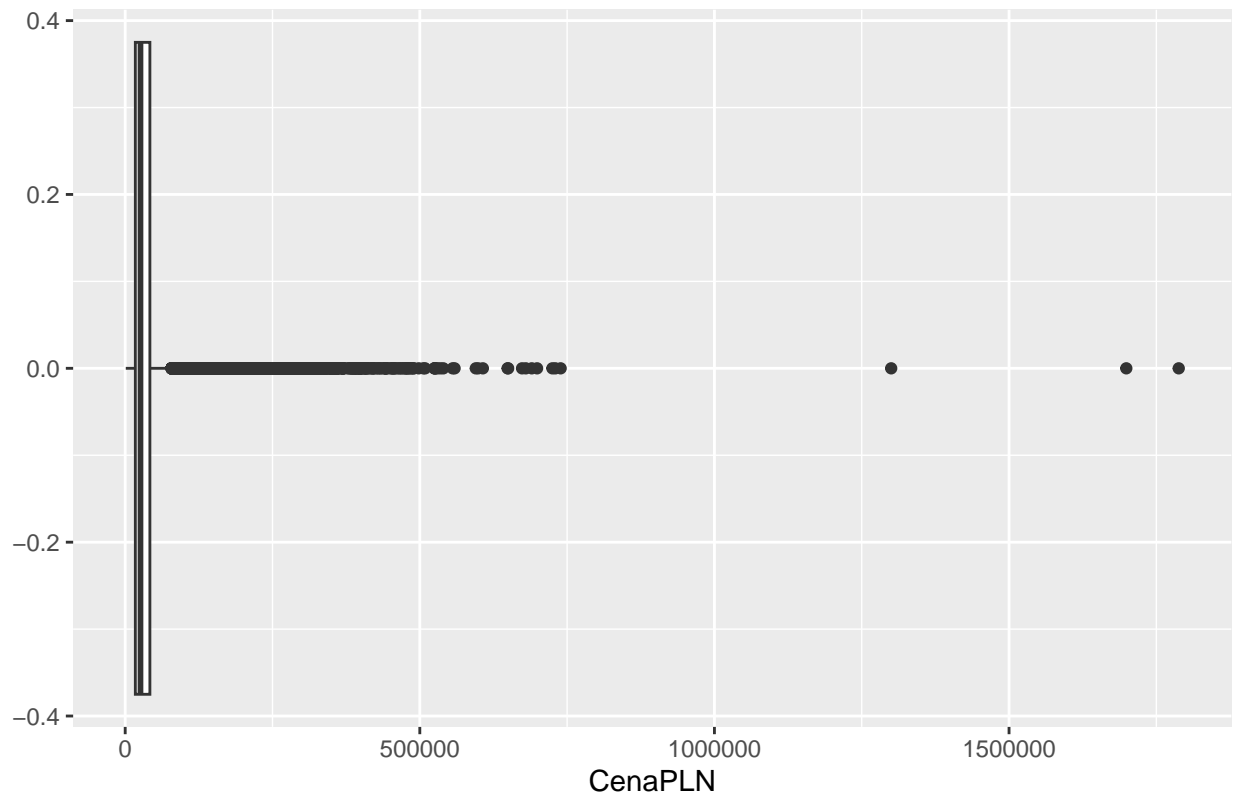


```

ggplot(data=database) + geom_boxplot(aes(x=CenaPLN)) + ggtitle("Wykres pudełko-wąsy dla cen sprzedaży")

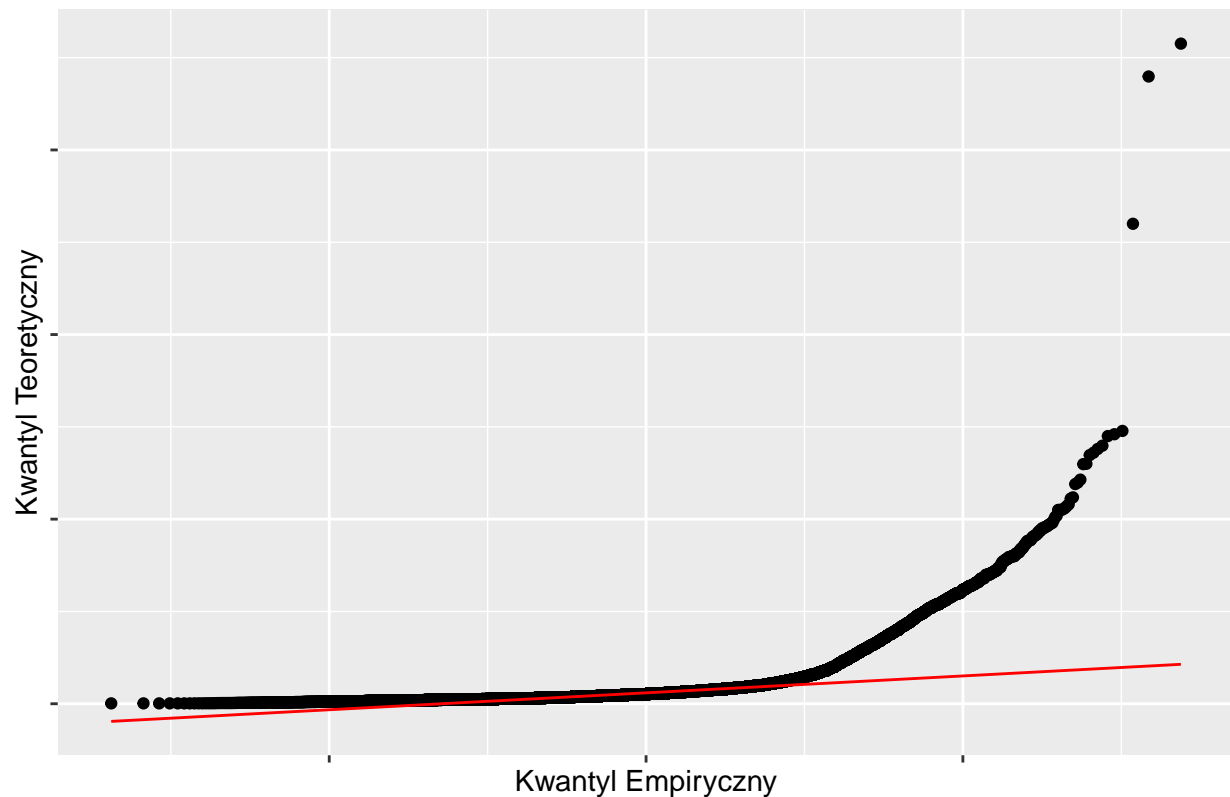
```


Wykres pudełko-w sy dla cen sprzedaży



```
ggplot(data=database, aes(sample = CenaPLN)) + stat_qq() + stat_qq_line(colour="red") + ylab("Kwantyl T")
ggtitle("Wykres QQ dla cen sprzedaży") + theme(axis.text.x = element_blank(), axis.text.y = element_b
```

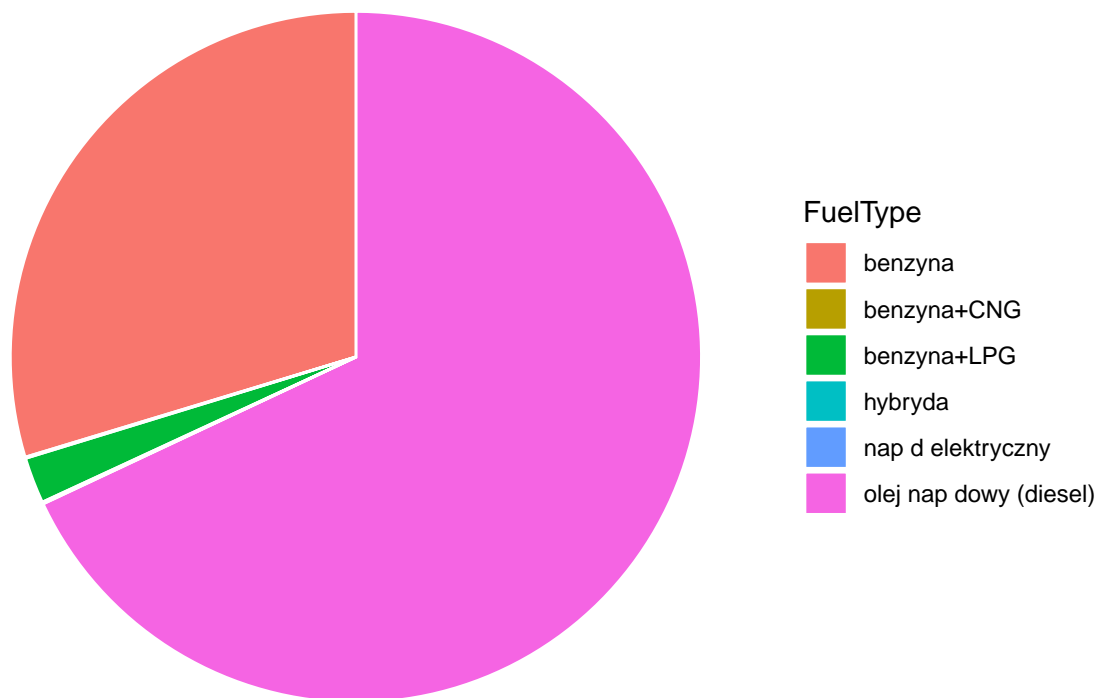
Wykres QQ dla cen sprzedaży



```
fuelType = database %>% group_by(database$RodzajPaliwa) %>% summarise(liczba = n());
colnames(fuelType) = c("FuelType", "Count");
fuelType = fuelType[order(fuelType$Count, decreasing = TRUE),]
fuelType
```

```
## # A tibble: 6 x 2
##   FuelType      Count
##   <chr>      <int>
## 1 olej napędowy (diesel) 27865
## 2 benzyna      12161
## 3 benzyna+LPG    886
## 4 hybryda       25
## 5 benzyna+CNG    23
## 6 napęd elektryczny    11
```

```
ggplot(fuelType, aes(x="", y=Count, fill=FuelType)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) + theme_void();
```



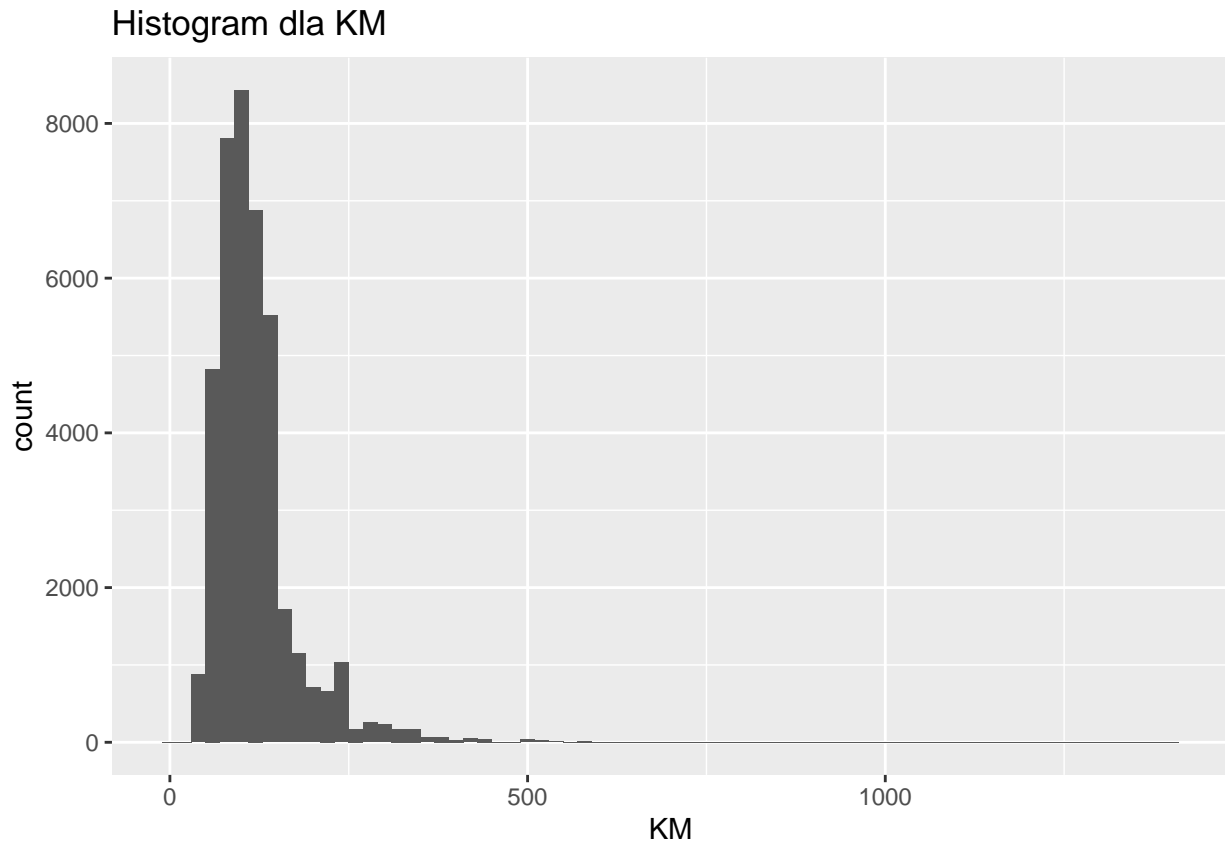
Liczba koni mechanicznych samochodu

```
summarise(database,
  srednia = mean(KM),
  mediana = median(KM),
  wariancja = var(KM),
  odchylenieStd = sd(KM),
  max = max(KM),
  min = min(KM),
)
```

```
##      srednia mediana wariancja odchylenieStd  max min
## 1 121.0052      110  3589.981      59.91645 1400  1
```

```
#upper_price_example = database[which(database$CenaPLN >= upper_bound),];
#head(upper_price_example[order(upper_price_example$CenaPLN, decreasing = TRUE),])
```

```
ggplot(data=database) + geom_histogram(aes(x=KM), binwidth = 20) + ggtitle("Histogram dla KM")
```



IV - Obserwacje odstające

Obserwacja odstająca, element odstający – obserwacja relatywnie odległa od pozostałych elementów próby. Innymi słowy, posiadająca nietypową wartość zmiennej niezależnej (objaśniającej) lub nietypowe wartości obydwu zmiennych – zależnej (objaśnianej) i objaśniającej (objaśniających w analizie regresji wielokrotnej).

Redukcja danych odstających zgodnie z regułą 3 sigma (*CenaPLN*)

Redukcja danych zgodnie z regułą 3 sigma.

Identyfikacja danych odstających za pomocą reguły trzech sigm:

1. Obliczenie średniej (μ): Najpierw oblicza się średnią dla danego zbioru danych.
2. Obliczenie odchylenia standardowego (σ): Następnie oblicza się odchylenie standardowe, które mierzy, jak bardzo dane rozpraszają się wokół średniej.
3. Ustalenie zakresu trzech sigm: Wartości, które znajdują się poza zakresem trzech sigm ($\mu \pm 3\sigma$), są uznawane za dane odstające.

```
iv_sigma_mean_price = mean(database$CenaPLN);
iv_sigma_sd_price   = sd(database$CenaPLN);

iv_sigma_lower_bound <- iv_sigma_mean_price - 3 * iv_sigma_sd_price;
iv_sigma_upper_bound <- iv_sigma_mean_price + 3 * iv_sigma_sd_price;

cat("Wyznacznowe granice metodą 3 sigma: \n")
```

```
## Wyznacznowe granice metodą 3 sigma:
cat("Dolna granica ", iv_sigma_lower_bound, ", górna granica ", iv_sigma_upper_bound, "\n");

## Dolna granica -113194.6 , górna granica 194891.3
iv_n_sigma_outlier = sum(database$CenaPLN >= iv_sigma_upper_bound |
                        database$CenaPLN <= iv_sigma_lower_bound)

cat("Ilość elementów odstających zgodnie z regułą 3 sigma dla CenaPLN n =", iv_n_sigma_outlier, "\n");

## Ilość elementów odstających zgodnie z regułą 3 sigma dla CenaPLN n = 1032
```

Redukcja danych odstających regułą odstępu międzykwartylowego (*PrzebiegKm*)

Metoda redukcji, polega na wykrywaniu i usuwaniu wartości odstających z zestawu danych. Metoda ta opiera się na kwartylach i rozstępie międzykwartylowym.

1. Wyznaczenie $Q1$ (pierwszy kwartył) oraz $Q3$ (trzeci kwartył)
2. Wyznaczenie odstępu międzykwartylowego $IQR = Q3 - Q1$
3. Wyznaczenie dolnej $Q1 - 1.5 \times IQR$ oraz górnej $Q1 + 1.5 \times IQR$ granicy.

```
iv_Q1 = quantile(database$PrzebiegKm, 0.25);
iv_Q3 = quantile(database$PrzebiegKm, 0.75);
iv_IQR = iv_Q3-iv_Q1;
iv_lower_bound = iv_Q1 - 1.5*iv_IQR #mniejsze niż
iv_upper_bound = iv_Q3 + 1.5*iv_IQR #większe niż

cat("Wyznacznowe granice metodą odstępu międzykwartylowego \n")

## Wyznacznowe granice metodą odstępu międzykwartylowego
cat("Dolna granica ", iv_lower_bound, ", górna granica ", iv_upper_bound, "\n");

## Dolna granica -55541.5 , górna granica 305902.5
iv_n_outlier = sum(database$PrzebiegKm >= iv_upper_bound |
                  database$PrzebiegKm <= iv_lower_bound)

cat("Ilość elementów odstających zgodnie z regułą odstępu międzykwartylowego PrzebiegKm n =", iv_n_outlier, "\n");

## Ilość elementów odstających zgodnie z regułą odstępu międzykwartylowego PrzebiegKm n = 137
```

V - Wyznaczanie prawdopodobieństw dla zmiennej

Rozkłady dyskretne

Dwumianowy

```
#Generowanie próbki
set.seed(42)
N <- 1000
n <- 20
p <- 0.4
X <- rbinom(N, n, p)
```

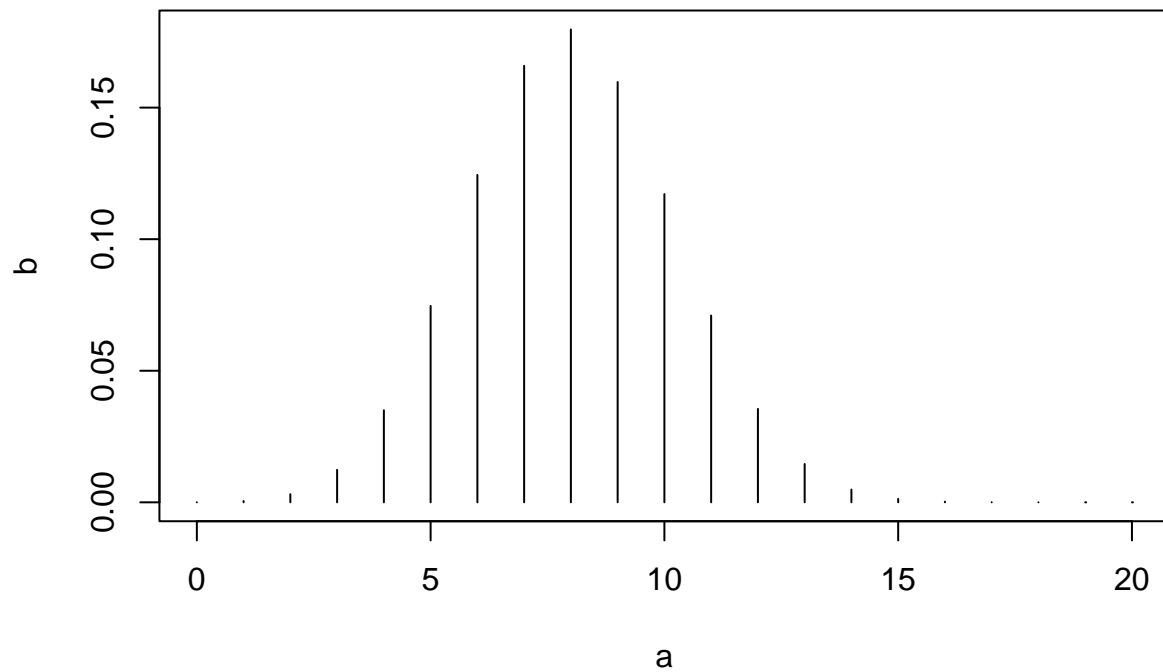
```
#Prawdopodobieństwa
pbinom(8, n, p) - pbinom(7, n, p) #  $P(X=8)$  p. punktowe
```

```
## [1] 0.1797058
```

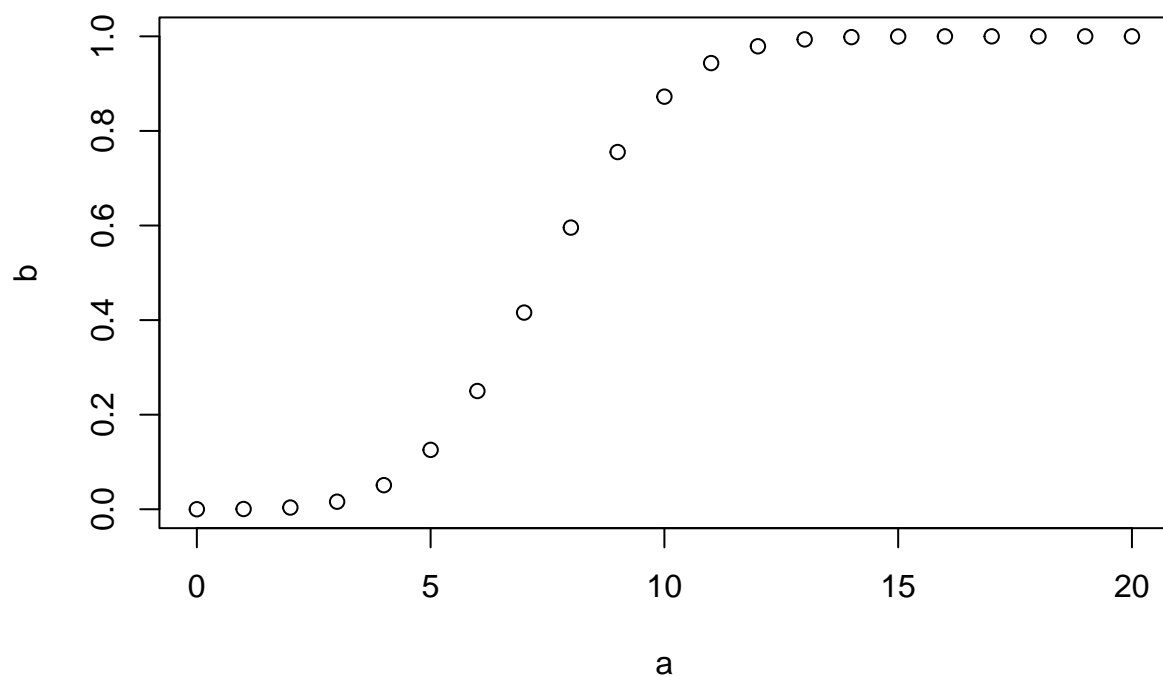
```
pbinom(10, n, p) #  $P(X \leq 10)$  p. przedziałowe
```

```
## [1] 0.8724788
```

```
#Wykres gęstości
a <- seq(0, n, by = 1)
b <- dbinom(a, n, p)
plot(a,b, type="h")
```



```
#Wykres dystrybucyjny
a <- seq(0, n, by = 1)
b <- pbinom(a, n, p)
plot(a,b)
```



```
### Rozkład Poissona
```

```
#Generowanie próbek
```

```
N <- 1000
```

```
lambda <- 200
```

```
X <- rpois(N, lambda)
```

```
#Prawdopodobieństwa
```

```
ppois(190, lambda) - ppois(189, lambda) # P(X=190)
```

```
## [1] 0.02243432
```

```
ppois(190, lambda) #(P<=190)
```

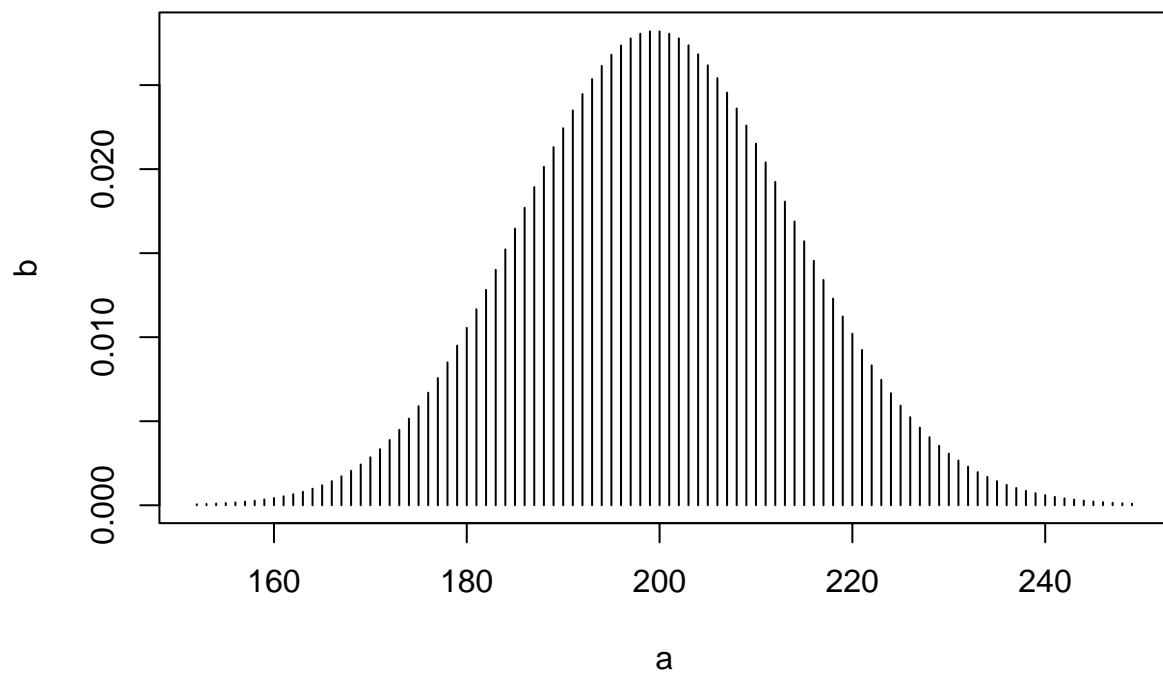
```
## [1] 0.2529326
```

```
#Wykres gęstości
```

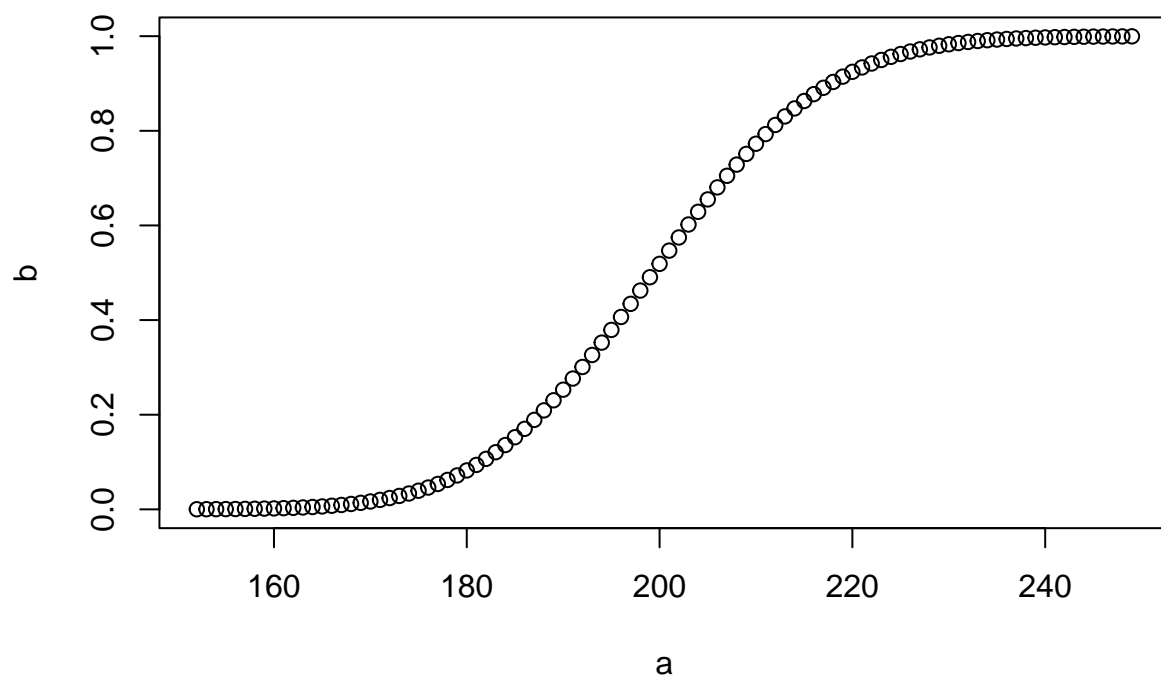
```
a <- seq(min(X), max(X), by = 1)
```

```
b <- dpois(a, lambda)
```

```
plot(a,b, type="h")
```



```
#Wykres dystrybucyjny  
a <- seq(min(X), max(X), by = 1)  
b <- ppois(a, lambda)  
plot(a,b)
```

Rozkłady ciągłe

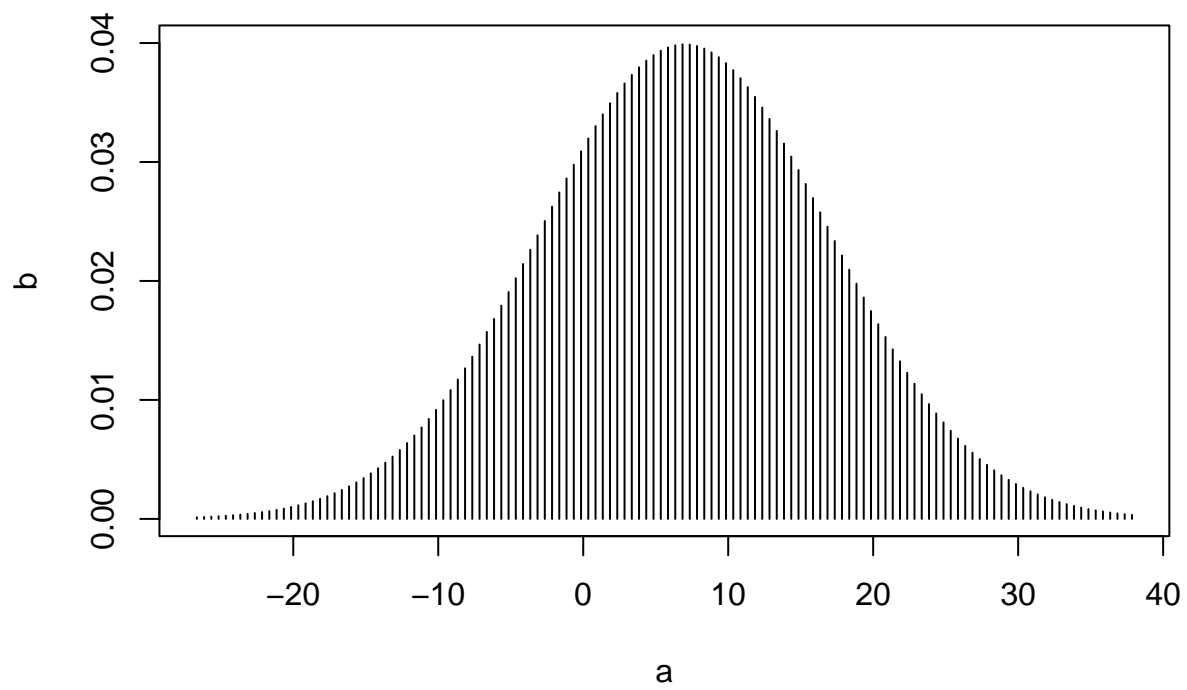
Rozkład normalny

```
#Generowanie próbek
N <- 1000
mu <- 7 # średnia
sigma <- 10 # odchylenie standardowe
X <- rnorm(N, mu, sigma)

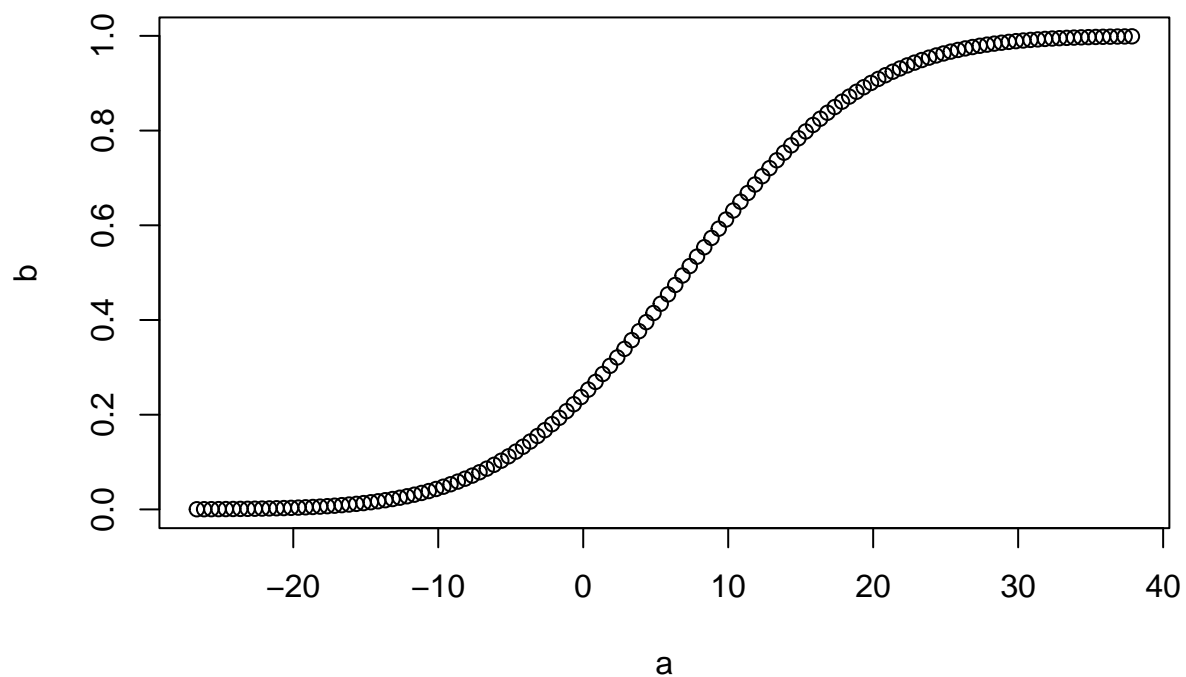
#Prawdopodobieństwa
#Prawd. punktowe w rozkładzie ciągłym = 0
pnorm(10, mu, sigma) # P(X<=10) p. przedziałowe

## [1] 0.6179114

#Wykres gęstości
a <- seq(min(X), max(X), by = 0.5)
b <- dnorm(a, mu, sigma)
plot(a,b, type="h")
```



```
#Wykres dystrybucyjny  
a <- seq(min(X), max(X), by = 0.5)  
b <- pnorm(a, mu, sigma)  
plot(a,b)
```



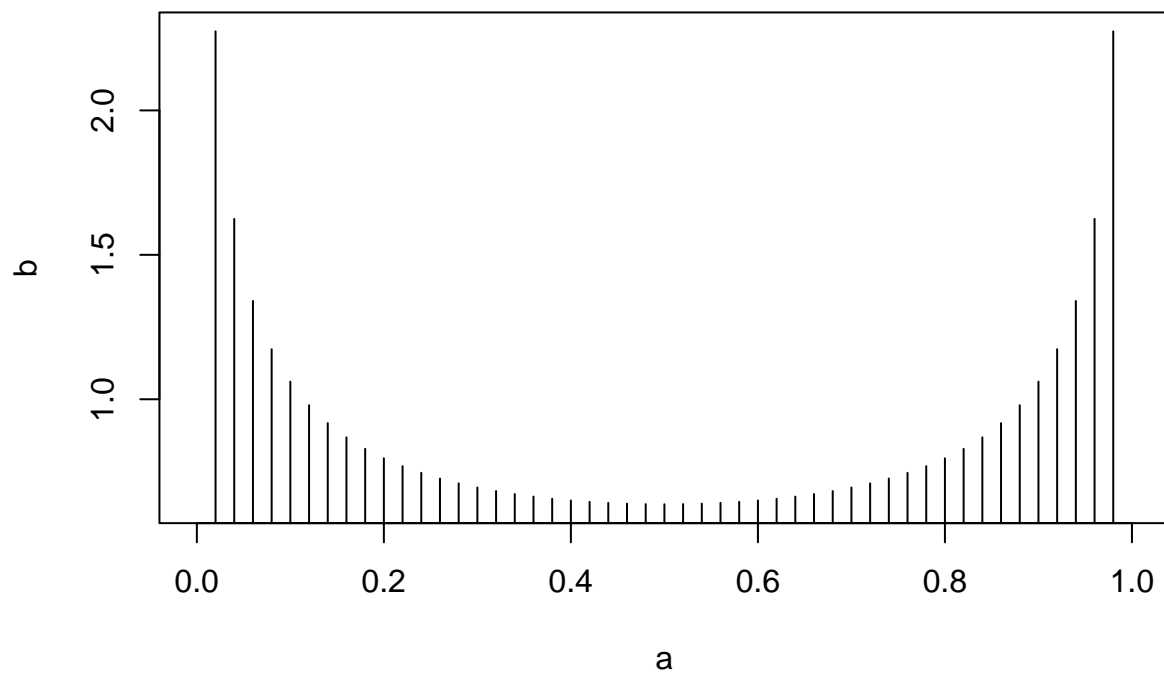
Rozkład Beta

```
#Generowanie próbki
N <- 1000
alpha <- 0.5
beta <- 0.5
X <- rbeta(N, alpha, beta)

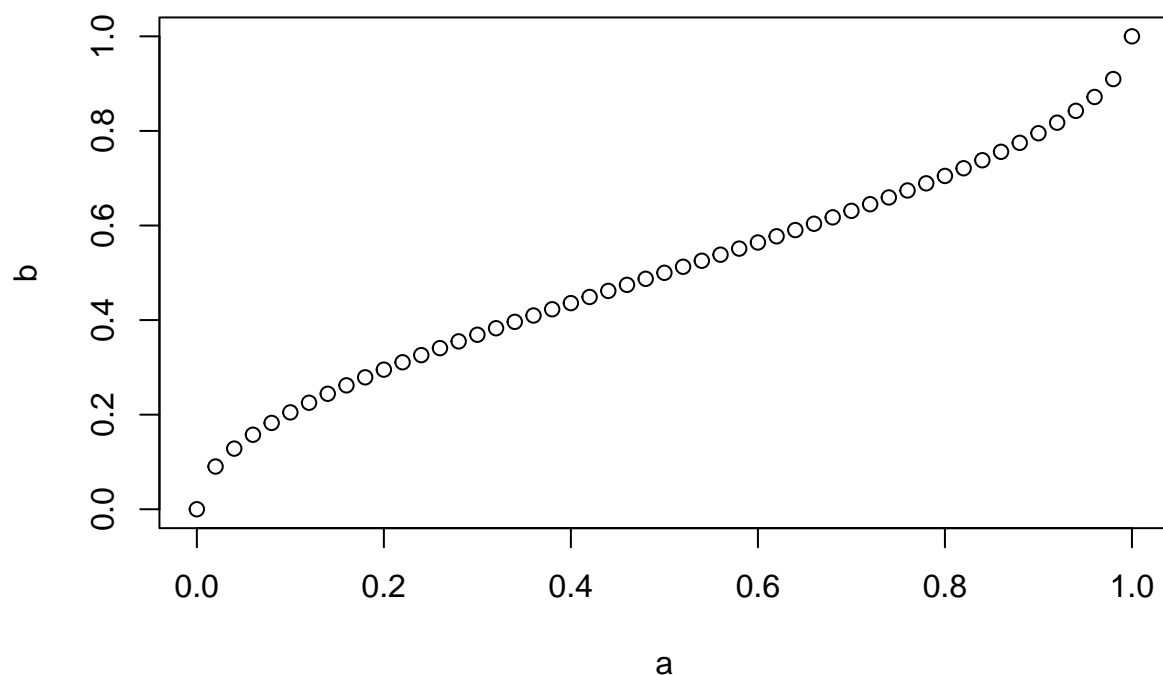
#Prawdopodobieństwa
#Prawd. punktowe w rozkładzie ciągłym = 0
pbeta(0.7, alpha, beta) # P(X<=0.7) p. przedziałowe

## [1] 0.6309899

#Wykres gęstości
a <- seq(0, 1, by = 0.02)
b <- dbeta(a, alpha, beta)
plot(a,b, type="h")
```



```
#Wykres dystrybucyjny  
a <- seq(0, 1, by = 0.02)  
b <- pbeta(a, alpha, beta)  
plot(a,b)
```



VI - Budowa macierzy z bazy danych

Do zbudowanie macierzy wykorzystano wszystkie dostępne numeryczne dane tj. *CenaPLN*, *KM*, *PrzebiegKm*, *RokProdukcji*, *PojemnoscSkokowa*. Dzięki temu możliwe jest wyznaczenie macierzy korelacji pomiędzy danymi.

```
# Select data
vi_database = data.matrix(select(filter(database, PojazdUszkodzony==FALSE & SkrzyniaBiegowManualna == T),
                                c("CenaPLN", "KM", "PrzebiegKm", "RokProdukcji", "PojemnoscSkokowa")))
```

```
#Is matrix
is.matrix(vi_database)
```

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

```
#Display head
head(vi_database)
```

```
##      CenaPLN  KM PrzebiegKm RokProdukcji PojemnoscSkokowa
## [1,]  27900 150     80840         2005           1900
## [2,]  28000 116    166000         2004           2000
## [3,]  25500 150    112000         2002           1781
## [4,]  29900 109     42000         2005           1991
## [5,]  29800 207    169000         2004           2946
## [6,]  21400 122    160000         2003           1800
```

```
# Dimension
vi_dim <- dim(vi_database);
```

```

cat("Rozmiar macierzy:", vi_dim[1], "x", vi_dim[2]);

## Rozmiar macierzy: 32776 x 5

# Columns mean
vi_means = round(colMeans(vi_database), 3);
cat(paste(names(vi_means), vi_means, sep = " : ", collapse = ",\n"))

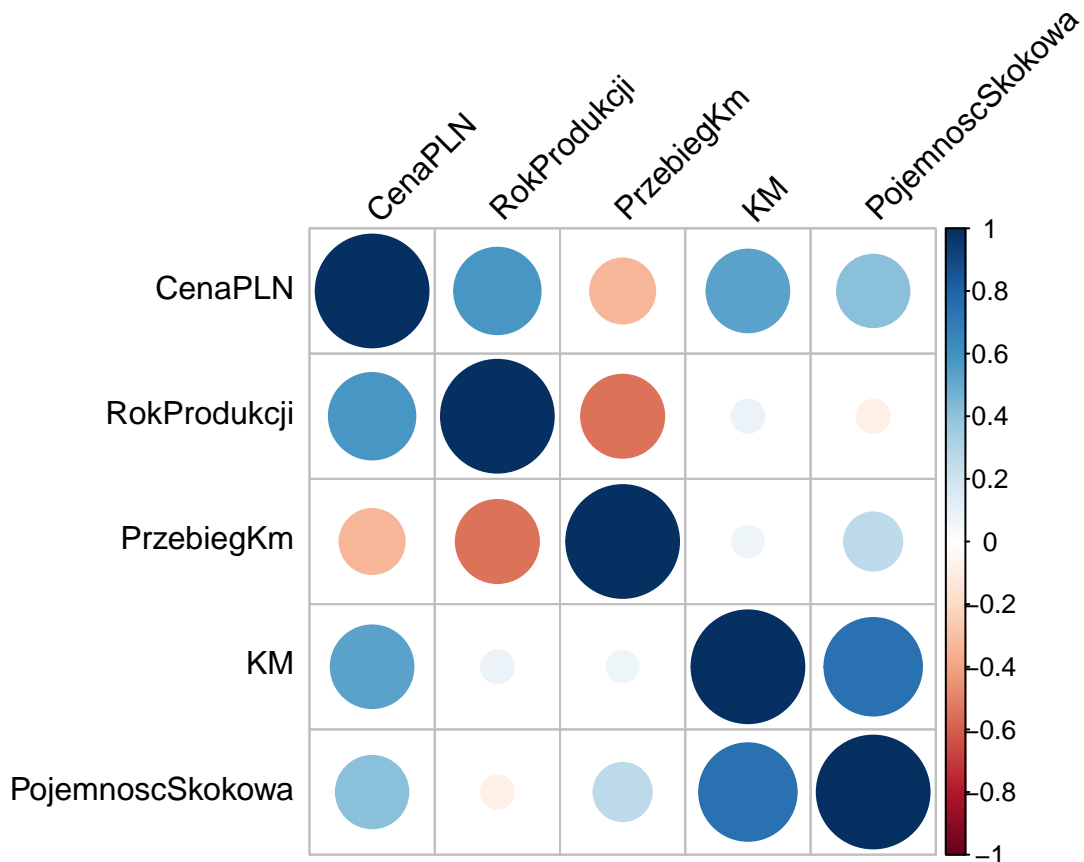
## CenaPLN : 29568.104,
## KM : 104.715,
## PrzebiegKm : 127770.544,
## RokProdukcji : 2005.09,
## PojemnoscSkokowa : 1742.825

# Correlation
vi_corr_matrix = cor(vi_database);
print(vi_corr_matrix)

##
##          CenaPLN          KM  PrzebiegKm RokProdukcji
## CenaPLN      1.0000000  0.53784100 -0.33277193   0.58752036
## KM           0.5378410  1.00000000  0.07887219   0.08579668
## PrzebiegKm   -0.3327719  0.07887219  1.00000000  -0.54240181
## RokProdukcji  0.5875204  0.08579668 -0.54240181  1.00000000
## PojemnoscSkokowa 0.4122382  0.74804407  0.26670668 -0.08499544
##
##          PojemnoscSkokowa
## CenaPLN      0.41223816
## KM           0.74804407
## PrzebiegKm   0.26670668
## RokProdukcji -0.08499544
## PojemnoscSkokowa 1.00000000

corrplot(vi_corr_matrix, order = "hclust",
          tl.col = "black", tl.srt = 45)

```



Jak można zaobserwować: - *CenaPLN* skorelowana jest z *RokProdukcji* (silnie), *KM*, *PojemnoscSkokowa* oraz odwrotnie z *PrzebiegKm*,
 - *PojemnoscSkokowa* jest silnie skorelowana z ilością *KM* (większa pojemność -> więcej KM), - *PrzebiegKm* jest odwrotnie skorelowany z *RokProdukcji* (starszy samochód -> większy przebieg).

VII - Przedziały ufności

W tej sekcji przedstawione zostaną badania określające przedziały ufności z różnym stopniem 'zaufania'. Oznacza to, że jeśli grupa badana była zgromadzona w sposób losowy to rzeczywisty parametr populacji z określonym stopniem 'zaufania' znajduje się w tym przedziale.

Zmienna numeryczna

W celu określenia przedziałów ufności zmiennej numerycznej wybrano cechę *CenaPLN*.

Przedział ufności dla średniej:

Ze względu na dużą liczebność próby $n > 30$, przedział ufności dla średniej zgodnie z wzorem:

$$P\left(\bar{X} - u_{1-\frac{\alpha}{2}} \frac{s}{\sqrt{n}}, \bar{X} + u_{1-\frac{\alpha}{2}} \frac{s}{\sqrt{n}}\right) = 1 - \alpha$$

gdzie:

- \bar{X} - średnia,
- n - to liczebność próby losowej,
- s - to odchylenie standardowe z próby,

- u_α - to wartość kwantyla $1 - \frac{\alpha}{2}$ rozkładu normalnego standaryzowanego dla poziomu istotności α .

```
# Meta data
# Confidence level 0.9, 0.95, 0.99
confidence_level = c(0.9, 0.95, 0.99);

# Basic data
vii_price_mean = mean(database$CenaPLN);
vii_price_sd = sd(database$CenaPLN);
n = length(database$CenaPLN);

# Mean confidence function
mean_confidence <- function(mean, sd, n, confidence_level) {
  alpha = 1 - confidence_level;
  offset = qnorm(1 - alpha / 2) * sd / sqrt(n);
  lower_bound = mean - offset;
  upper_bound = mean + offset;
  return(c(lower_bound, upper_bound));
};

# Calculate confidence
mean_confidence_intervals = sapply(confidence_level, function(conf_level) {
  mean_confidence(vii_price_mean, vii_price_sd, n, conf_level)
});

# Create matrix
mean_confidence_intervals = t(mean_confidence_intervals);
rownames(mean_confidence_intervals) = paste0("Poziom ufności: ", confidence_level);
colnames(mean_confidence_intervals) = c("Dolny przedział", "Górny przedział");

# Display
print("Przedziały ufności średniej dla różnych poziomów ufności:");

## [1] "Przedziały ufności średniej dla różnych poziomów ufności:"
print(mean_confidence_intervals);

##              Dolny przedział Górny przedział
## Poziom ufności: 0.9          40431.08      41265.61
## Poziom ufności: 0.95         40351.15      41345.54
## Poziom ufności: 0.99         40194.91      41501.78

print(paste("Wartość średnia z próby:", round(vii_price_mean,2)));

## [1] "Wartość średnia z próby: 40848.35"
```

Przedział ufności dla odchylenia standardowego:

Ze względu na dużą liczebność próby $n > 30$, przedział ufności dla odchylenia standardowego zgodnie z wzorem:

$$P\left(\frac{S}{1+\frac{u_\alpha}{\sqrt{2n}}} < \sigma < \frac{S}{1-\frac{u_\alpha}{\sqrt{2n}}}\right) = 1 - \alpha$$

gdzie:

- n - to liczebność próby losowej,
- S - to odchylenie standardowe z próby,

- u_α - to wartość kwantyla $1 - \frac{\alpha}{2}$ rozkładu normalnego standaryzowanego dla poziomu istotności α .

```
# Meta data
# Confidence level 0.9, 0.95, 0.99
confidence_level = c(0.9, 0.95, 0.99);

# Basic data
vii_price_mean = mean(database$CenaPLN);
vii_price_sd = sd(database$CenaPLN);
n = length(database$CenaPLN);

# SD confidence function
sd_confidence <- function(sd, n, confidence_level) {
  alpha = 1 - confidence_level;
  ua = qnorm(1 - alpha / 2);
  offset = ua / sqrt(2*n);
  lower_bound = sd / (1 + offset);
  upper_bound = sd / (1 - offset);
  return(c(lower_bound, upper_bound));
};

# Calculate confidence
sd_confidence_intervals = sapply(confidence_level, function(conf_level) {
  sd_confidence(vii_price_sd, n, conf_level);
});

# Create matrix
sd_confidence_intervals = t(sd_confidence_intervals);
rownames(sd_confidence_intervals) = paste0("Poziom ufności: ", confidence_level);
colnames(sd_confidence_intervals) = c("Dolny przedział", "Górny przedział");

# Display
print("Przedziały ufności odchylenia standardowego dla różnych poziomów ufności:");

## [1] "Przedziały ufności odchylenia standardowego dla różnych poziomów ufności:"
print(sd_confidence_intervals);

##              Dolny przedział Górny przedział
## Poziom ufności: 0.9          51054.28      51644.40
## Poziom ufności: 0.95         50998.46      51701.64
## Poziom ufności: 0.99         50889.72      51813.89

print(paste("Wartość odchylenia standardowego z próby:", round(vii_price_sd,2)));

## [1] "Wartość odchylenia standardowego z próby: 51347.64"
```

Przedział ufności dla zmiennej jakościowej (frakcyjna)

Do wyznaczenia przedziału ufności dla zmiennej *PojazdUszkodzony*.

$$p \in \left(\frac{m}{n} - u_{1-\frac{\alpha}{2}} \frac{\sqrt{\frac{m}{n} \left(1 - \frac{m}{n}\right)}}{\sqrt{n}}, \frac{m}{n} + u_{1-\frac{\alpha}{2}} \frac{\sqrt{\frac{m}{n} \left(1 - \frac{m}{n}\right)}}{\sqrt{n}} \right)$$

gdzie:

m - ilość zdarzeń sprzyjających,

n - ilość wszystkich zdarzeń,

- u_α - to wartość kwantyla $1 - \frac{\alpha}{2}$ rozkładu normalnego standaryzowanego dla poziomu istotności α .

```
# Meta data
# Confidence level 0.9, 0.95, 0.99
confidence_level = c(0.9, 0.95, 0.99);

# Basic data
vii_positive = sum(database$PojazdUszkodzony == TRUE);
vii_all      = length(database$PojazdUszkodzony);

mean_bool_confidence <- function(m, n, confidence_level) {
  alpha = 1 - confidence_level;
  mn = m / n;
  offset = qnorm(1 - alpha / 2) * sqrt(mn * (1-mn)) / sqrt(n);
  lower_bound = mn - offset;
  upper_bound = mn + offset;
  return(c(lower_bound, upper_bound));
};

# Calculate confidence
confidence_intervals = sapply(confidence_level, function(conf_level) {
  mean_bool_confidence(vii_positive, vii_all, conf_level);
});

# Create matrix
confidence_intervals = t(confidence_intervals);
rownames(confidence_intervals) = paste0("Poziom ufności: ", confidence_level);
colnames(confidence_intervals) = c("Dolny przedział", "Górny przedział");

# Display
print("Przedziały ufności dla frakcji dla różnych poziomów ufności (Udział pojazdów uszkodzonych):");

## [1] "Przedziały ufności dla frakcji dla różnych poziomów ufności (Udział pojazdów uszkodzonych):"
print(confidence_intervals);

##
##          Dolny przedział Górny przedział
## Poziom ufności: 0.9      0.03446330      0.03749003
## Poziom ufności: 0.95     0.03417338      0.03777995
## Poziom ufności: 0.99     0.03360675      0.03834658

print(paste("Udział pojazdów uszkodzonych w zbiorze ", round(vii_positive/length(database$PojazdUszkodzony), 2)));

## [1] "Udział pojazdów uszkodzonych w zbiorze  0.04"
```

Przedział ufności dla zmiennej jakościowej

Do analizy wybrano pole *Marka*.

```
vii marka = database %>% group_by(database$Marka) %>% summarise(percent = n()/length(database$Marka));
```

Przedział ufności dla średniej z udziału *Marki* w zbiorze

Ze względu na małą liczebność próby $n < 30$, przedział ufności dla odchylenia standardowego zgodnie z wzorem:

$$P\left(\bar{X} - t_{\alpha, n-1} \frac{s}{\sqrt{n}}, \bar{X} + t_{\alpha, n-1} \frac{s}{\sqrt{n}}\right) = 1 - \alpha$$

gdzie:

- \bar{X} - średnia,
- n - to liczebność próby losowej,
- s - to odchylenie standardowe z próby,
- $t_{\alpha, n-1}$ - dystrybuanta rozkładu t-studenta dla 1-a i n-1 stopni swobody.

```
# Meta data
# Confidence level 0.9, 0.95, 0.99
confidence_level = c(0.9, 0.95, 0.99);

# Basic data
vii_marka_mean = mean(vii_marka$percent);
vii_marka_sd = sd(vii_marka$percent);
n = length(vii_marka$percent);

# Mean confidence function
mean_confidence <- function(mean, sd, n, confidence_level) {
  alpha = 1 - confidence_level;
  offset = pt(alpha, n-1) * sd / sqrt(n);
  lower_bound = mean - offset;
  upper_bound = mean + offset;
  return(c(lower_bound, upper_bound));
};

# Calculate confidence
mean_confidence_intervals = sapply(confidence_level, function(conf_level) {
  mean_confidence(vii_marka_mean, vii_marka_sd, n, conf_level)
});

# Create matrix
mean_confidence_intervals = t(mean_confidence_intervals);
rownames(mean_confidence_intervals) = paste0("Poziom ufności: ", confidence_level);
colnames(mean_confidence_intervals) = c("Dolny przedział", "Górny przedział");

# Display
print("Przedziały ufności średniej z danych jakościowych dla różnych poziomów ufności:");

## [1] "Przedziały ufności średniej z danych jakościowych dla różnych poziomów ufności:"
print(mean_confidence_intervals);

##
##          Dolny przedział Górny przedział
## Poziom ufności: 0.9      0.08539988      0.09641830
## Poziom ufności: 0.95     0.08559816      0.09622002
## Poziom ufności: 0.99     0.08575720      0.09606098

print(paste("Wartość średnia z próby:", round(vii_marka_mean, 3)));

## [1] "Wartość średnia z próby: 0.091"
```

Przedział ufności dla średniej z udziału *Marki* w zbiorze

Ze względu na małą liczebność próby $n < 30$, przedział ufności dla odchylenia standardowego zgodnie z wzorem:

$$P\left(\frac{n \cdot S^2}{\chi^2_{1-\frac{\alpha}{2}, n-1}} < \sigma^2 < \frac{n \cdot S^2}{\chi^2_{\frac{\alpha}{2}, n-1}}\right) = 1 - \alpha$$

gdzie:

- n - to liczebność próby losowej,
- S - to odchylenie standardowe z próby,
- $\chi^2_{1-\frac{\alpha}{2}, n-1}$ - kwantyle rozkładu Chi-kwadrat 1-a i $n-1$ stopni swobody.

```
# Meta data
# Confidence level 0.9, 0.95, 0.99
confidence_level = c(0.9, 0.95, 0.99);

# Basic data
vii_marka_mean = mean(vii_marka$percent);
vii_marka_sd = sd(vii_marka$percent);
n = length(vii_marka$percent);

# Mean confidence function
sd_confidence <- function(sd, n, confidence_level) {
  alpha = 1 - confidence_level;
  lower_bound = sqrt((n * sd * sd) / (qchisq(1-alpha/2, n-1)));
  upper_bound = sqrt((n * sd * sd) / (qchisq(alpha/2, n-1)));
  return(c(lower_bound, upper_bound));
};

# Calculate confidence
sd_confidence_intervals = sapply(confidence_level, function(conf_level) {
  sd_confidence(vii_marka_sd, n, conf_level);
});

# Create matrix
sd_confidence_intervals = t(sd_confidence_intervals);
rownames(sd_confidence_intervals) = paste0("Poziom ufności: ", confidence_level);
colnames(sd_confidence_intervals) = c("Dolny przedział", "Górny przedział");

# Display
print("Przedziały ufności odchylenia standardowego dla różnych poziomów ufności:");

## [1] "Przedziały ufności odchylenia standardowego dla różnych poziomów ufności:"
print(sd_confidence_intervals);

##
##          Dolny przedział Górny przedział
## Poziom ufności: 0.9      0.02628534      0.05665759
## Poziom ufności: 0.95     0.02484986      0.06241419
## Poziom ufności: 0.99     0.02240909      0.07659717
print(paste("Wartość odchylenia standardowego z próby:", round(vii_marka_sd,2)));

## [1] "Wartość odchylenia standardowego z próby: 0.03"
```

VIII - Testowanie hipotez

Zmienne jakościowe

Test niezależności

Za pomocą testu niezależności χ^2 sprawdzona zostanie niezależność pomiędzy Marką a udziałem w nich samochodów ze skrzynią mechaniczną.

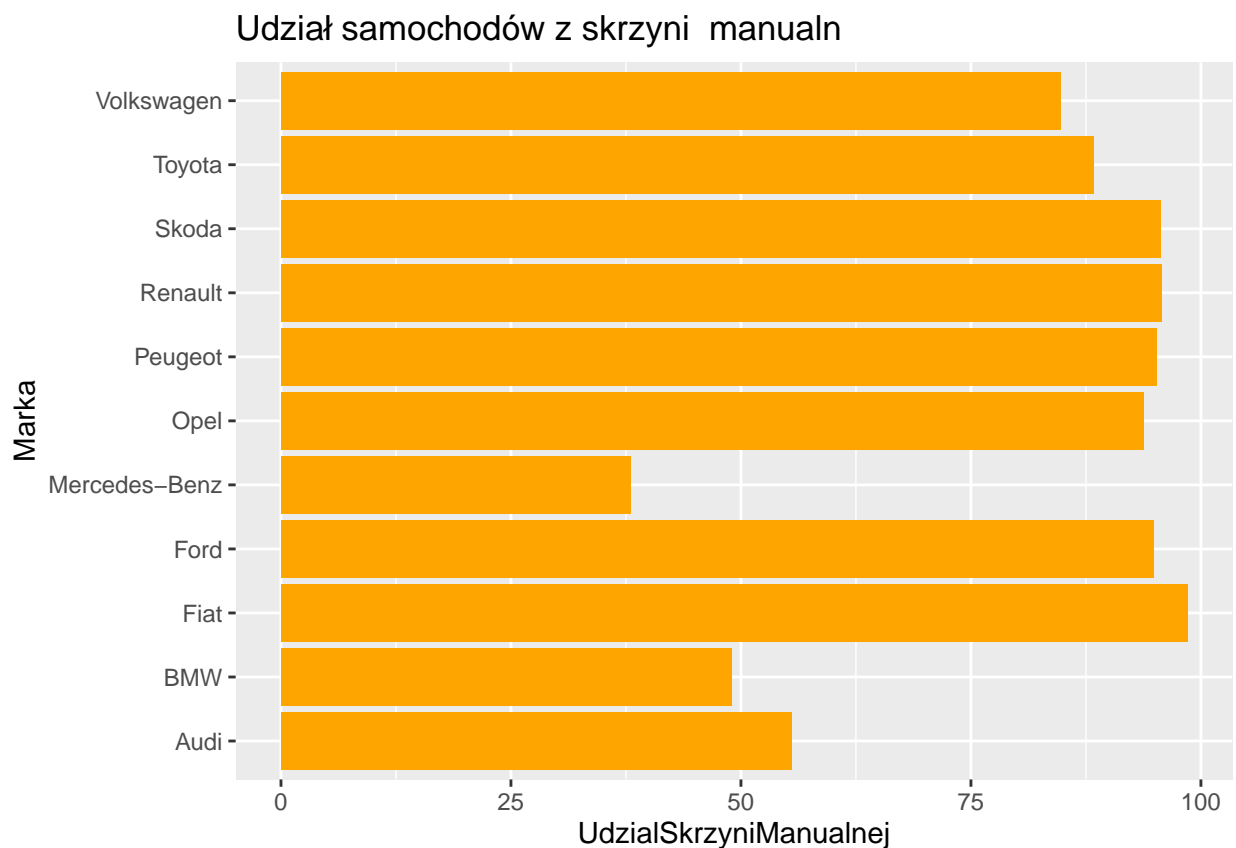
Hipoteza:

H_0 - udział skrzyni mechanicznej jest niezależny od marki,

H_1 - udział skrzyni mechanicznej jest zależny od marki,

Poziom istotności $\alpha = 0.05$.

```
vii_cross = table(database$Marka, database$SkrzyniaBiegowManualna)
vii_table = data.frame(Marka=rownames(vii_cross), UdzialSkrzyniManualnej = (vii_cross[,2]/(vii_cross[,1]
ggplot(vii_table, aes(y=Marka, x=UdzialSkrzyniManualnej)) +
  geom_bar(position="dodge", stat="identity", fill="orange")+
  ggtitle("Udział samochodów z skrzynią manualną")
```



```
chisq.test(vii_cross)
```

```
##
## Pearson's Chi-squared test
##
## data:  vii_cross
```

```
## X-squared = 10428, df = 10, p-value < 2.2e-16
```

Zgodnie z wynikiem testu mamy podstawy do odrzucenia hipotezy zerowej na korzyść hipotezy alternatywnej
- udział skrzynni mechanicznej jest zależny od marki.

Test proporcji

Test proporcji pozwala odpowiedzieć na pytanie czy odsetki w jednej, dwóch lub więcej grupach różnią się od siebie istotnie.

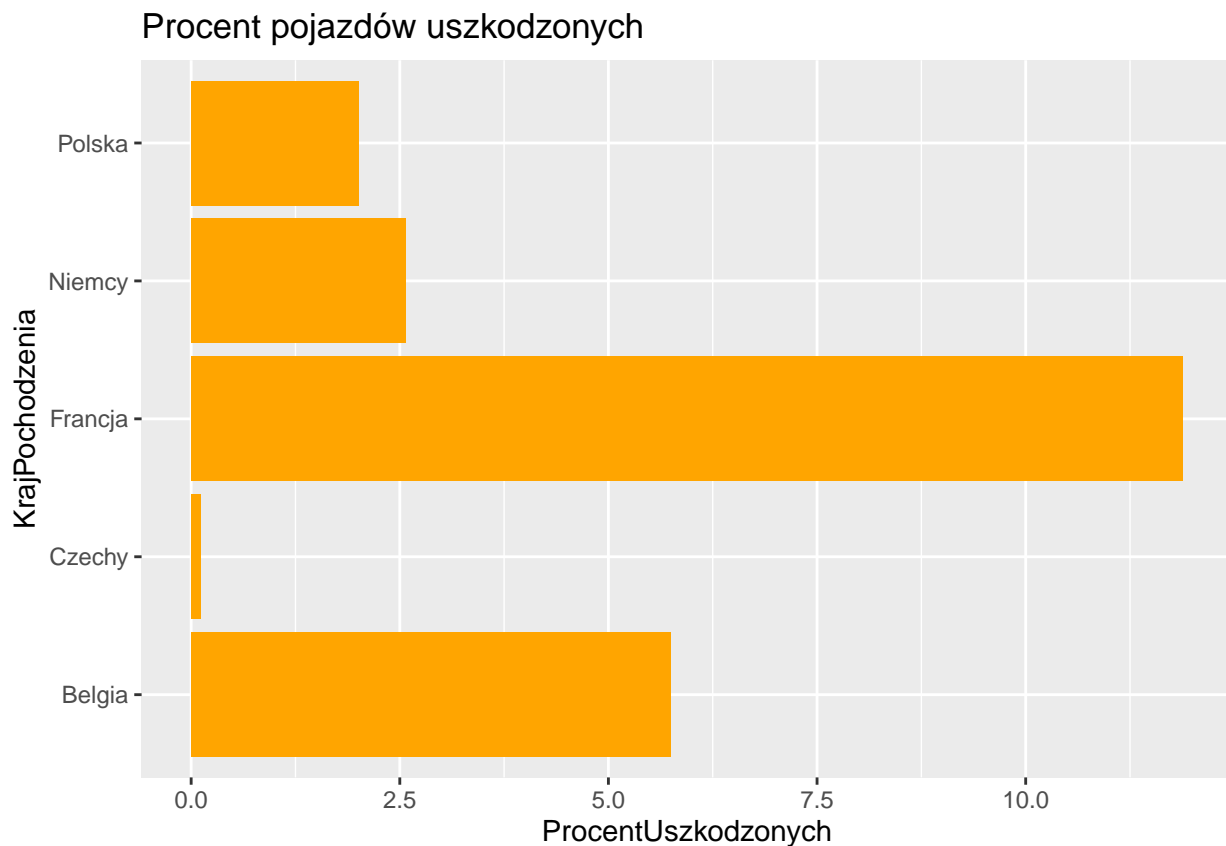
Hipoteza:

H_0 - odsetek pojazdów uszkodzonych jest niezależny od kraju pochodzenia,

H_1 - odsetek pojazdów uszkodzonych jest zależny od kraju pochodzenia,

Poziom istotności $\alpha = 0.05$.

```
vii_proporcja = database %>% group_by(KrajPochodzenia) %>% summarise(Liczebosc = n());  
vii_proporcja$Uszkodzonych = table(database$KrajPochodzenia, database$PojazdUszkodzony)[,2];  
vii_proporcja$ProcentUszkodzonych = (vii_proporcja$Uszkodzonych / vii_proporcja$Liczebosc) * 100;  
vii_proporcja = vii_proporcja %>% filter(Liczebosc > 1000);  
  
ggplot(vii_proporcja, aes(y=KrajPochodzenia, x=ProcentUszkodzonych)) +  
  geom_bar(position="dodge", stat="identity", fill="orange") +  
  ggtitle("Procent pojazdów uszkodzonych")
```



```
vii_test = prop.test(x = vii_proporcja$Uszkodzonych, n = vii_proporcja$Liczebosc);  
print(vii_test);
```

```
##
## 5-sample test for equality of proportions without continuity
## correction
##
## data: vii_proporcja$Uszkodzonych out of vii_proporcja$Liczebnosc
## X-squared = 1074.4, df = 4, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
##      prop 1      prop 2      prop 3      prop 4      prop 5
## 0.057422969 0.001156961 0.118827568 0.025647319 0.020011482
```

Zgodnie z wynikiem testu mamy podstawy do odrzucenia hipotezy zerowej na korzyść hipotezy alternatywnej - odsetek pojazdów uszkodzonych jest zależny od kraju pochodzenia.

Zmienne ilościowe

Test normalności

Test umożliwia weryfikację czy cecha ma rozkład normalny.

Hipoteza:

H_0 - cena pojazdów ma rozkład normalny,

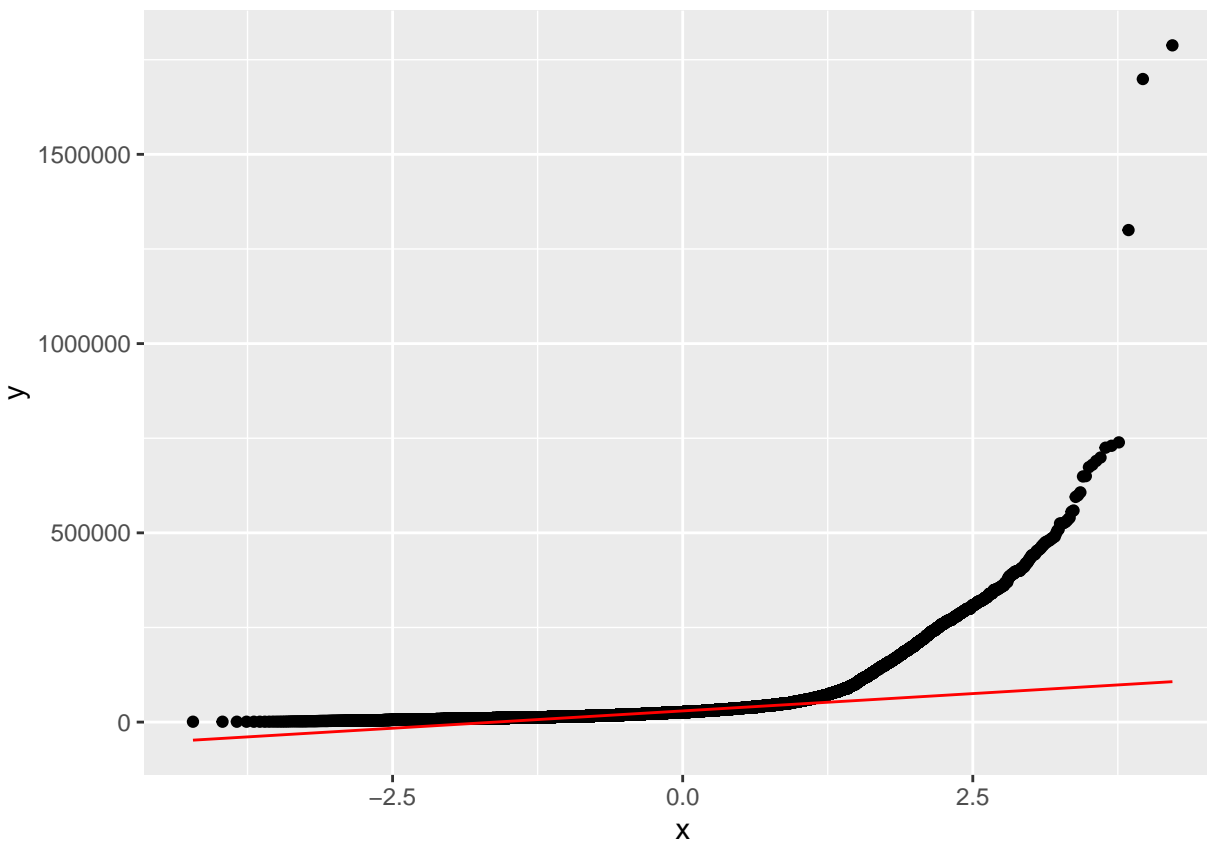
H_1 - cena pojazdów nie ma rozkładu normalnego,

Poziom istotności $\alpha = 0.05$.

```
vii_test = ad.test(database$CenaPLN);
print(vii_test);
```

```
##
## Anderson-Darling normality test
##
## data: database$CenaPLN
## A = 5521.3, p-value < 2.2e-16
```

```
ggplot(database, aes(sample = CenaPLN)) +
  stat_qq() +
  stat_qq_line(color="red");
```



Zgodnie z wynikiem testu mamy podstawy do odrzucenia hipotezy zerowej na korzyść hipotezy alternatywnej
- cena pojazdów nie ma rozkładu normalnego.

Test średnich

Ze względu na to, iż cena nie jest zmienną o rozkładzie normalnym. Wykorzystujemy nieparametryczny test Wilcoxona.

Hipoteza:

H_0 - średnia jest równa 41000,

H_1 - średnia nie jest równa 41000,

Poziom istotności $\alpha = 0.05$.

```
wilcox.test(x = database$CenaPLN, mu = 41000)
```

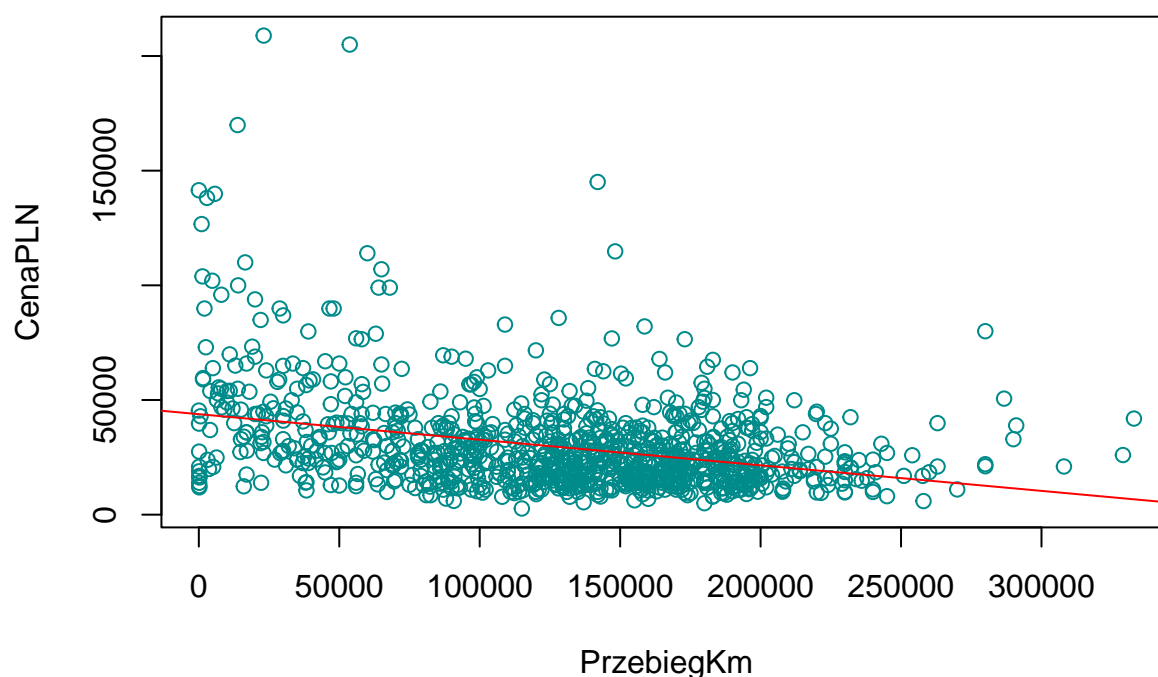
```
##
## Wilcoxon signed rank test with continuity correction
##
## data: database$CenaPLN
## V = 238698925, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 41000
```

Zgodnie z wynikiem testu mamy podstawy do odrzucenia hipotezy zerowej na korzyść hipotezy alternatywnej
- średnia nie jest równa 41000.

IX - Regresja liniowa i inne

Modele dla jednej zmiennej

```
ix_database = select(filter(database, PojazdUszkodzony==FALSE & SkrzyniaBiegowManualna == TRUE),
                      c("CenaPLN", "PrzebiegKm"))
model = lm(CenaPLN ~ ., data = ix_database)
ix_sample = ix_database[sample(nrow(ix_database), size=1000),]
plot(CenaPLN ~ PrzebiegKm, data=ix_sample, col = 'darkcyan')
abline(model, col="red")
```



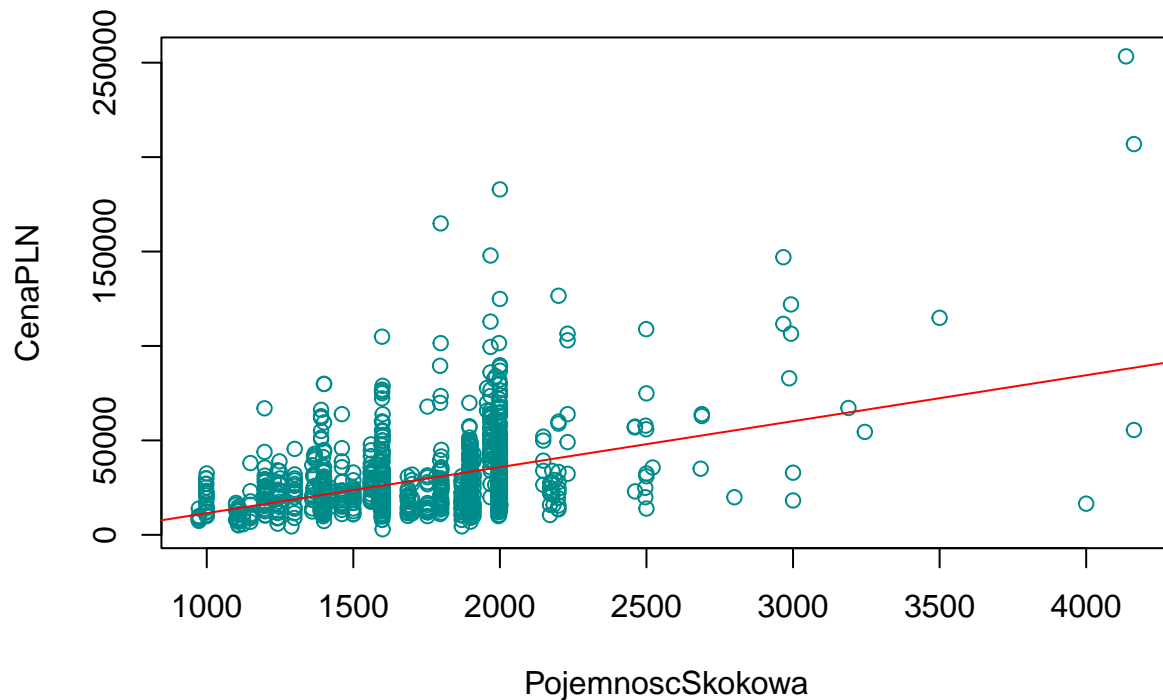
```
summary(model);
```

```
##
## Call:
## lm(formula = CenaPLN ~ ., data = ix_database)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -39901 -11905  -4136   6516  637321
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.382e+04  2.507e+02  174.76  <2e-16 ***
## PrzebiegKm   -1.115e-01  1.746e-03  -63.88  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 20740 on 32774 degrees of freedom
## Multiple R-squared:  0.1107, Adjusted R-squared:  0.1107
## F-statistic: 4081 on 1 and 32774 DF,  p-value: < 2.2e-16
print(model$coefficients);

## (Intercept)  PrzebiegKm
## 43817.457268    -0.111523

ix_database = select(filter(database, PojazdUszkodzony==FALSE & SkrzyniaBiegowManualna == TRUE),
                      c("CenaPLN", "PojemnoscSkokowa"))
model = lm(CenaPLN ~ ., data = ix_database)
ix_sample = ix_database[sample(nrow(ix_database), size=1000),]
plot(CenaPLN ~ PojemnoscSkokowa, data=ix_sample, col = 'darkcyan')
abline(model, col="red")
```



```
summary(model);

##
## Call:
## lm(formula = CenaPLN ~ ., data = ix_database)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -124863  -11495   -3285    6505   566066
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.286e+04  5.297e+02  -24.28  <2e-16 ***
## PojemnoscSkokowa  2.435e+01  2.972e-01   81.91  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20030 on 32774 degrees of freedom
## Multiple R-squared:  0.1699, Adjusted R-squared:  0.1699
## F-statistic:  6710 on 1 and 32774 DF,  p-value: < 2.2e-16
print(model$coefficients);
```

```
##      (Intercept) PojemnoscSkokowa
##      -12862.74176          24.34602
```

Modele dla 2 zmiennych

```
ix_database = select(filter(database, PojazdUszkodzony==FALSE & SkrzyniaBiegowManualna == TRUE),
                      c("CenaPLN", "PrzebiegKm", "PojemnoscSkokowa"))
model = lm(CenaPLN ~ . , data = ix_database)

plot3d(model, col='darkcyan', plane.col = 'red')
rglwidget()

## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead

## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```
summary(model);
```

```
##
## Call:
## lm(formula = CenaPLN ~ ., data = ix_database)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -167185   -9096   -1707    6209   521446
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.538e+03  4.627e+02  -11.97  <2e-16 ***
## PrzebiegKm    -1.597e-01  1.511e-03 -105.69  <2e-16 ***
## PojemnoscSkokowa 3.185e+01  2.663e-01  119.61  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17300 on 32773 degrees of freedom
## Multiple R-squared:  0.3809, Adjusted R-squared:  0.3809
## F-statistic: 1.008e+04 on 2 and 32773 DF,  p-value: < 2.2e-16
```

```
print(model$coefficients);
```

```
##      (Intercept)      PrzebiegKm PojemnoscSkokowa
##      -5537.7325250      -0.1597319       31.8533893
```

Drzewa decyzyjne

```
ix_database = select(filter(database, PojazdUszkodzony==FALSE & SkrzyniaBiegowManualna == TRUE),
                      c("CenaPLN", "RokProdukcji", "PojemnoscSkokowa"))
ix_data_split = initial_split(ix_database, prop = 0.75);
ix_train_data <- training(ix_data_split);
ix_test_data <- testing(ix_data_split);

tree_spec <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("regression")

tree_fit <- tree_spec %>%
  fit(CenaPLN ~ ., data = ix_train_data)

predictions <- tree_fit %>%
  predict(ix_test_data) %>%
  pull(.pred)

metrics <- metric_set(rmse, rsq)
model_performance <- ix_test_data %>%
  mutate(predictions = predictions) %>%
  metrics(truth = CenaPLN, estimate = predictions)

print(model_performance)

## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>      <dbl>
## 1 rmse    standard    12300.
## 2 rsq     standard      0.675
rpart.plot(tree_fit$fit, type = 5, extra = 101, under = TRUE, cex = 0.8, box.palette = "auto")

## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary)
## To silence this warning:
##   Call rpart.plot with roundint=FALSE,
##   or rebuild the rpart model with model=TRUE.
```

