

California Housing: Statystyczna analiza danych

404811, Mykola Haltiuk, czwartek 17⁵⁰
AGH, Wydział Informatyki Elektroniki i Telekomunikacji
Rachunek prawdopodobieństwa i statystyka 2020/2021

Kraków, 23 stycznia 2021

Ja, niżej podpisany(na) własnoręcznym podpisem deklaruję, że przygotowanym(lam) przedstawiony do oceny projekt samodzielnie i żadna jego część nie jest kopią pracy innej osoby.

.....

1 Streszczenie raportu

Raport powstał w oparciu o analizę danych dotyczących cen na nieruchomości w Kalifornii stanem na 1990 rok.

2 Opis danych

Dane do projektu pochodzą ze strony Kaggle. Są one przedstawione w pliku *.csv*. Ze strony można spokojnie ich pobrać.

Otoż, dane składają się z 20640 rekordów o 10 cechach. Każdy rekord odpowiada za pewny blok domów. Opis poszczególnych cech:

- **longitude** - długość geograficzna bloku (*numeryczna*)
- **latitude** - szerokość geograficzna bloku (*numeryczna*)
- **housing_median_age** - mediana wieków domów w bloku (*numeryczna*)
- **total_rooms** - ilość pokoi w bloku (*numeryczna*)
- **total_bedrooms** - ilość sypaleń w bloku (*numeryczna*)
- **population** - populacja w bloku (ilość osób) (*numeryczna*)
- **households** - ilość gospodarstw w bloku (*numeryczna*)
- **median_income** - mediana zarobków w tym bloku (*numeryczna*)
- **median_house_value** - mediana cen na dom (*numeryczna*)

- **ocean_proximity** - bliskość do oceanu (*kategoryczna*)

Dane nie są gotowe do analizy, potrzebują dużo modyfikowania i czyszczenia. Bardzo dużo jest wartości odstających, praktycznie wszystkie są z góry, czyli większe od mediany, co psuje rozkłady, które już nie da się nazwać normalnymi (będzie omówiono później i bardziej szczegółowo).

Istnieje kilka problemów z cechami:

- **housing_median_age** jak i **median_house_value** są ograniczone od góry (*clipped*). Czyli wszystkie wartości powyżej pewnej wartości x są zmniejszone do tej wartości. To powoduje duże skoncentrowanie w pewnych punktach.
- **total_bedrooms** ma niektóre wartości NA, czyli są nieznane (*missing values*).
- **ocean_proximity** jest zmienną kategoryczną o 5 różnych wartości, które nie są uporządkowane, co może sprawić problem, bo trudno będzie przekształcić tą zmienną kategoryczną na numeryczną.

Samo przygotowanie danych zostanie opisane troszeczkę niżej w sekcji **Data cleaning**.

Także niektóre cechy nie pokazują ciekawej informacji, więc zostaną reorganizowane i usunięte, a na ich miejscu zostaną stworzone nowe. Czyli zrobimy tak zwany *feature engineering*. Sam proces zostanie też opisany niżej.

Przed przystąpieniem do przygotowywania danych spróbujmy ich załadować i zobaczyć jak wyglądają.

```
> fpath <- "housing.csv"
> all_housing <- read.csv(fpath, header=TRUE, stringsAsFactors = FALSE)
```

Otoż, plik przez nas załadowany ma nazwę *housing.csv*, wczytujemy go jako **data.frame** zachowując nazwy kolumn.

Zobaczmy czy zgadzają się wymiary z przedstawionymi liczbami wyżej.

```
> dim(all_housing)
```

```
[1] 20640      10
```

Jak widzimy, to rzeczywiście: mamy 20640 rekordów o 10 cechach, czyli 20640 wierszy i 10 kolumn. Zobaczmy także czy nazwy odpowiadają przedstawionym wyżej:

```
> colnames(all_housing)
```

```
[1] "longitude"           "latitude"            "housing_median_age" "total_rooms"
[5] "total_bedrooms"       "population"          "households"         "median_income"
[9] "median_house_value"   "ocean_proximity"
```

Wszystko zgadza się. W końcu zobaczymy mały opis tych danych i 4 pierwsze rekordy, żeby dobrze rozumieć czym są i jak wyglądają.

```

> str(all_housing)
'data.frame': 20640 obs. of 10 variables:
$ longitude      : num -122 -122 -122 -122 -122 ...
$ latitude       : num 37.9 37.9 37.9 37.9 37.9 ...
$ housing_median_age: num 41 21 52 52 52 52 52 52 42 52 ...
$ total_rooms     : num 880 7099 1467 1274 1627 ...
$ total_bedrooms  : num 129 1106 190 235 280 ...
$ population      : num 322 2401 496 558 565 ...
$ households      : num 126 1138 177 219 259 ...
$ median_income    : num 8.33 8.3 7.26 5.64 3.85 ...
$ median_house_value: num 452600 358500 352100 341300 342200 ...
$ ocean_proximity : chr "NEAR BAY" "NEAR BAY" "NEAR BAY" "NEAR BAY" ...
> head(all_housing, 4)
  longitude latitude housing_median_age total_rooms total_bedrooms population households
1   -122.23    37.88             41        880         129       322       126
2   -122.22    37.86             21       7099        1106      2401      1138
3   -122.24    37.85             52       1467         190       496       177
4   -122.25    37.85             52       1274         235       558       219
  median_income median_house_value ocean_proximity
1          8.3252        452600      NEAR BAY
2          8.3014        358500      NEAR BAY
3          7.2574        352100      NEAR BAY
4          5.6431        341300      NEAR BAY

```

Wszystkie kolumny, oprócz **ocean_proximity** są numeryczne, ostatnia jest cechą kategoryczną, więc musimy to później naprawić. W tej chwili możemy sobie zobaczyć wszystkie możliwe wartości tej zmiennej:

```

> levels(as.factor(all_housing$ocean_proximity))
[1] "<1H OCEAN"  "INLAND"     "ISLAND"     "NEAR BAY"    "NEAR OCEAN"

```

Otoż, mamy 5 możliwych wartości:

- **INLAND** - w środku Kalifornii, daleko od oceanu
- **<1H OCEAN** - w jednej godzinie jazdy od oceanu
- **NEAR OCEAN** - blisko oceanu
- **NEAR BAY** - blisko zatoki (San-Francisco)
- **ISLAND** - wyspa

Dokładniej i na karcie to zostanie przedstawione później, gdy będziemy zajmować się tą zmienną kategoryczną.

Otoż, otrzymaliśmy jakąś pierwotną informację na temat tego zbioru. Niżej spróbujemy go przeanalizować jeszcze głębiej, żeby przygotować do prawdziwej statystycznej analizy.

3 Data cleaning

3.1 NA wartości

Sprawdźmy sobie ile jest nieznanych wartości w każdej kolumnie.

```
> colSums(is.na(all_housing))
```

	longitude	latitude	housing_median_age	total_rooms
	0	0	0	0
	total_bedrooms	population	households	median_income
	207	0	0	0
	median_house_value	ocean_proximity		
	0	0		

Jak i było napisane w liście problemów tego zbioru, w kolumnie **total_bedrooms** pojawia się 207 nieznanych wartości, z którymi musimy coś zrobić. Istnieją różne sposoby na rozwiązanie tego problemu:

- Usunięcie wszystkich rekordów posiadających nieznane wartości. W naszym przypadku to jest całkiem sensowne podejście, bo takich rekordów jest dużo mniej w stosunku do ilości wszystkich.
- Usunięcie całej kolumny, ale tak tracimy dużo ciekawej informacji, więc to odrzucamy od razu.
- Przypisanie jakiejś wartości w miejscu nieznanych wartości, np. 0. To nie jest dobrym podejściem i bardzo zależy od samych danych, bo jeżeli dane są z przedziału [1200, 1250], to otrzymujemy bardzo dużo wartości odstającecych, które psują nasz rozkład. Tą możliwość od razu odrzucamy.
- Przypisanie jakiejś statystyki rozkładu w miejscu nieznanych wartości, czyli mediany, wartości średniej i tp. To jest bardzo sensowne podejście i stosuje się najczęściej.
- Analiza zależności z innymi cechami i sprawdzenie możliwości przewidywania nieznanej wartości na podstawie innych cech. To podejście jest rzadko stosowane, ale jest ciekawe i pozwala nam dość logicznie podesbrać te wartości, a nie przypisać po prostu coś.

Otoż, o ile chciałoby się zachować wszystkie rekordy, to odrzucamy pierwszą możliwość. O ile chcemy przeprowadzać testy, to 200 jednakowych wartości może trochę wpływać na wynik, i o ile dodatkowo ostatnie podejście jest najciekawsze, to spróbujmy stworzyć minimalny model regresyjny, żeby podesbrać nieznane wartości, ale na początku musimy sprawdzić czy zdążyliśmy to zrobić bazując się tylko na jednej zmiennej, czyli chcemy sprawdzić jak silnie wpływają inne cechy na **total_bedrooms** i czy są silnie skorelowane.

Żeby policzyć macierz współczynników korelacji, musimy na początku złożyć tylko numeryczne cechy, a przed tym jeszcze usunąć rekordy z nieznanymi wartościami:

```

> cleaned <- all_housing[rowSums(is.na(all_housing)) == 0,]
> colSums(is.na(cleaned)) # rechecking

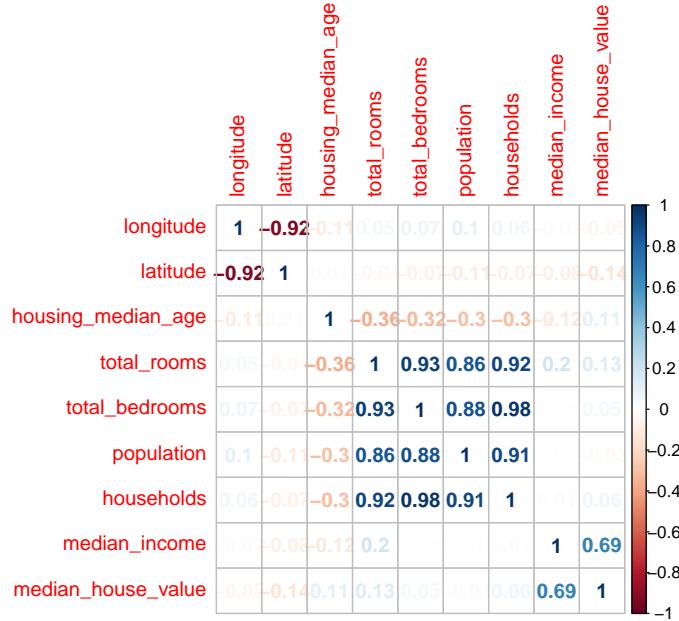
longitude          latitude housing_median_age      total_rooms
0                  0                 0                  0
total_bedrooms     population        households median_income
0                  0                 0                  0
median_house_value ocean_proximity
0                  0

```

```

> cleanedNumeric <- cleaned[, purrr::map_lgl(cleaned, is.numeric)]
> # calculating the correlation matrix
> hcor <- cor(cleanedNumeric)
> # plotting
> corrplot(hcor, method='number')

```



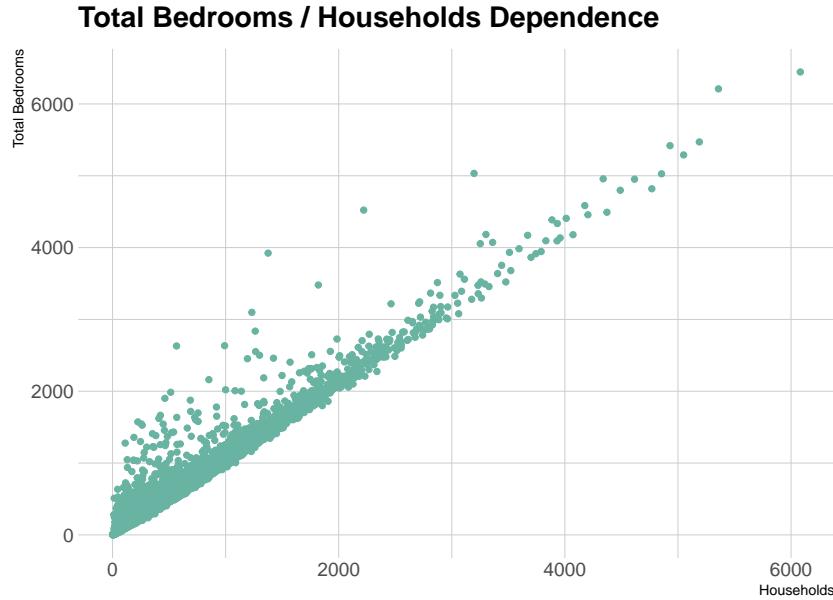
Najbardziej nas interesuje kolumna/wiersz **total_bedrooms**, dlatego, że musimy znaleźć inną cechę, która jest bardzo skorelowana z **total_bedrooms**. Jak widzimy, cecha **households** ma współczynnik korelacji z interesującą nas kolumną równy 0.98, co jest bardzo dużą wartością i możemy z pewnością powiedzieć, że z dość silnym prawdopodobieństwem zgadniemy wartość bliską rzeczywistej **total_bedrooms** mając wartość **households**. Współczynnik korelacji bliski 1 wskazuje, że im więcej gospodarstw, tym więcej sypaleń, co wydaje się być dość logiczne.

Warto także zauważyć, że mamy dużo cech silnie skorelowanych, co nie wpływa dobrze na tworzenie różnych modeli regresyjnych i klasyfikacyjnych, bo zwiększa wymiarowość rekordów nie zwiększając efektywnej informacji. Ten problem rozwiążemy w **Feature engineering**.

Wróćmy do naszych nieznanych wartości. Spróbujmy narysować sobie wykres zależności tych dwóch cech i dodatkowo sprawdźmy dokładniejszą wartość współczynnika korelacji.

```
> brcor <- cor(cleaned$total_bedrooms, cleaned$households)
> brcor
[1] 0.9797283

> ggplot(cleaned, aes(x=households, y=total_bedrooms)) +
+   labs(title="Total Bedrooms / Households Dependence",
+        x="Households", y="Total Bedrooms") +
+   theme(plot.title=element_text(size=30, face="bold", family='sans'),
+         axis.text.x=element_text(size=15, family='sans'),
+         axis.text.y=element_text(size=15, family='sans'),
+         axis.title.x=element_text(size=25, family='sans'),
+         axis.title.y=element_text(size=25, family='sans')) +
+   geom_point( color="#69b3a2" ) +
+   theme_ipsum(base_family = 'sans')
```



Otocz, widzimy, że zależność jest naprawdę bliska liniowej. *Mały komentarz: taki dług kod został wykorzystany do rysowania większości wykresów, więc dalej nie będzie przedstawiony, a będą pokazane same wykresy, żeby nie zaśmiecać powtarzającym się kodem ten raport.*

Zgodnie z zapowiedzią, tworzymy prosty model regresyjny:

```
> bhmodel <- lm(total_bedrooms ~ households, cleaned)
> bhmodel
```

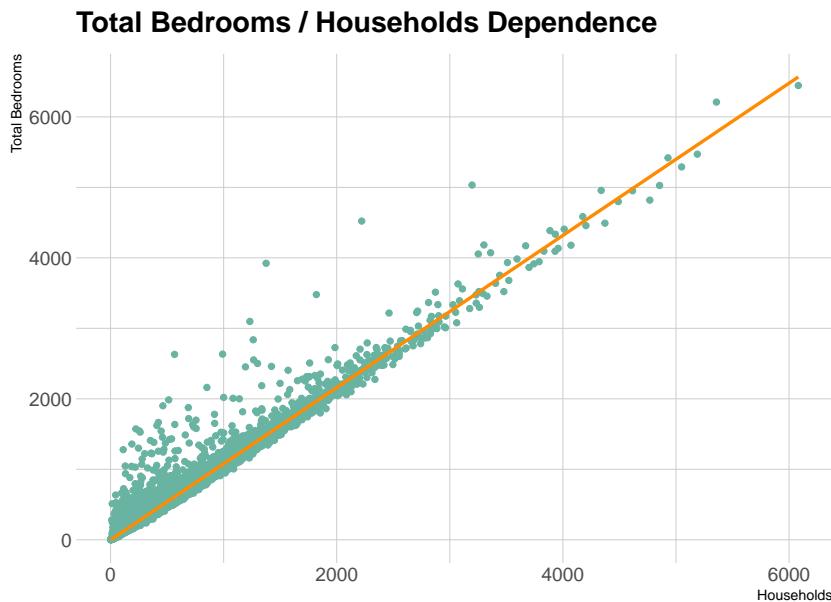
Call:

```
lm(formula = total_bedrooms ~ households, data = cleaned)
```

Coefficients:

(Intercept)	households
-1.465	1.080

Narysujmy sobie to od razu z prostą stworzoną podczas regresji:



Model został stworzony, więc zostało się tylko zamienić wszystkie *NA* na wartości otrzymane z modelu.

```
> # replacing the missing values with predictions of our model
> all_housing$total_bedrooms[is.na(all_housing$total_bedrooms)] =
+   predict(bhmodel, all_housing[is.na(all_housing$total_bedrooms), ])
```

```

> # checking if the null values has left
> colSums(is.na(all_housing))

  longitude      latitude housing_median_age total_rooms
    0             0                 0            0
total_bedrooms population households median_income
    0             0                 0            0
median_house_value ocean_proximity
    0             0

```

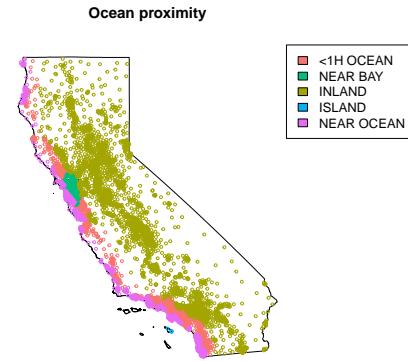
3.2 Cecha kategoryczna *ocean_proximity*

Pora zająć się ***ocean_proximity***, zmienną kategoryczną, którą musimy przekształcić na numeryczną. Istnieją na to różne sposoby:

- **Integer encoding** - zamiana wartości kategorycznych na numeryczne wprost stosując pewną funkcję mapującą. To podejście jest stosowane, gdy jest oczywisty pewny porządek wartości kategorycznych (np. *low* - *medium* - *high*).
- **One-hot encoding** - dekompozycja jednej cechy na kilka tak, żeby każda możliwa wartość kategoryczna przedstawiała nową kolumnę. I wtedy dla każdej kolumny przypisujemy *bool*-owe wartości, czyli pewien rekord miał tą wartość kategoryczną lub nie miał.

Chociaż nasz przypadek bardziej odpowiada podejściu one-hot, nie będziemy komplikować naszych danych, a po prostu spróbujmy stworzyć taki porządek możliwych wartości ***ocean_proximity***, żeby można było go uzasadnić, żeby nie był przypadkowym.

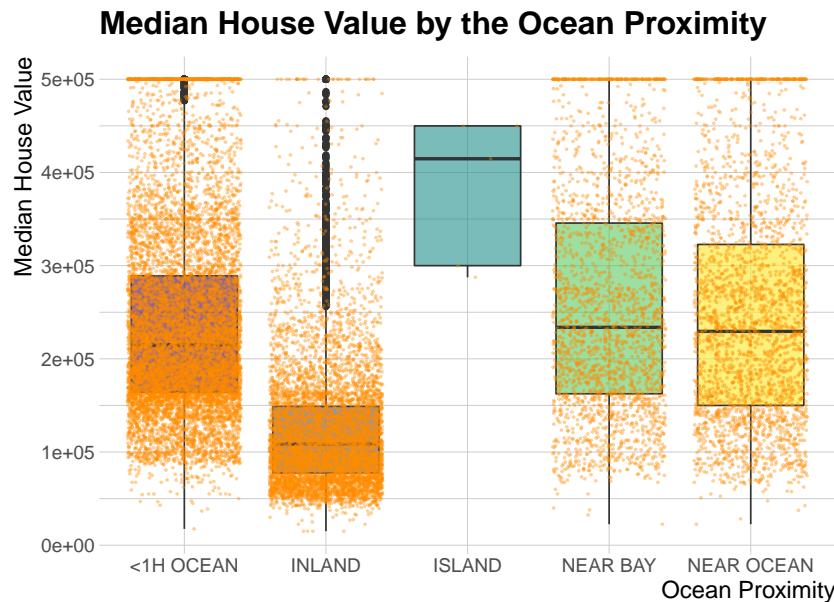
Na początku mając długość i szerokość geograficzną narysujmy sobie jak są rozrzucone te rekordy z różnymi wartościami ***ocean_proximity***.



Tworzymy funkcję mapującą w oparciu o **median_house_value**, bo właśnie cel tego zbioru danych - stworzenie modelu do przewidywania mediany cen domów, żeby firma-klient mogła zdecydować się lub nie na inwestycje.

Otóz, początkowo bazując się na własnym odczuciu stworzymy taki porządek: *INLAND* → *<1H OCEAN* → *NEAR BAY* → *NEAR OCEAN* → *ISLAND*. Uzasadnieniem temu porządkowi jest to, że im bliżej do oceanu, tym droższe stają się mieszkania.

Ale nie możemy budować analizy na własnych odczuciach, więc musimy sprawdzić jakoś naszą propozycję uporządkowania. Porównajmy mediany cen dla poszczególnych wartości kategorycznych i narysujmy od razu całe *box-plot'y* dla porównania:



Byliśmy dość blisko. Okazało się, że *NEAR BAY* ma medianę większą od mediany *NEAR OCEAN*. Pierwsze i trzecie kwantyle też odpowiadają naszemu porządkowi. Przewaga *NEAR BAY* może być objaśniona urbanizacją brzegu zatoki. Takie miasta jak San Francisco, Berkley, Oakland, Richmond, Fremont, San Jose, Silicon Valley są rozrzucone po brzegu zatoki. Sacramento też jest dość blisko, a to powoduje wzrost cen na mieszkanie.

Otóz, nasza funkcja mapująca wygląda następująco: *INLAND* → 1, *<1H OCEAN* → 2, *NEAR OCEAN* → 3, *NEAR BAY* → 4, *ISLAND* → 5.

Zamieniamy naszą cechę kategoryczną na numeryczną.

```
> fac <- factor(all_housing$ocean_proximity,
+                 levels=c('INLAND', '<1H OCEAN', 'NEAR OCEAN', 'NEAR BAY', 'ISLAND'))
```

```

> # let's check the mapping
> data.frame(levels = unique(fac), value = as.numeric(unique(fac)))

  levels value
1  NEAR BAY     4
2 <1H OCEAN     2
3   INLAND     1
4 NEAR OCEAN     3
5   ISLAND     5

> all_housing$ocean_proximity <- as.numeric(fac)

```

3.3 Wartości odstające (*outliers*)

Ze wszystkich wymienionych na początku problemów został się tylko jeden - ograniczone wartości (*capped*). Z box-plotu w poprzedniej sekcji możemy zobaczyć jak dużo wartości cen są na poziomie 500 tysięcy dolarów. Jest to dlatego, jak było wcześniej już powiedziane, że wszystkie ceny powyżej tego progu zostały zapisane równymi temu progowi. To tworzy pewne problemy z rozkładem i może wpływać na regresję dość znacząco.

Znajdźmy ten próg i sprawdźmy ile wartości mu odpowiadają:

```

> lim <- max(all_housing$median_house_value)
> lim

[1] 500001

> sum(all_housing$median_house_value == lim)

[1] 965

```

Czyli tak naprawdę, jeżeli rzeczywisty rozkład wartości jest X , to nasz $Y = \min(X, 500k)$.

Co możemy z tym zrobić?

- Możemy ich usunąć tracąc 5% danych i zamiast dużej ściany rekordów w tym miejscu powstanie przepaść, co też nie jest najlepszym rozwiązaniem.
- W przypadku gdy nie potrzebujemy, żeby model przewidywał wartości powyżej 500k, to najlepszym rozwiązaniem będzie zostawienie tego jak jest.
- W innym przypadku, musielibyśmy znaleźć prawdziwe wartości dla tych rekordów co dla nas nie jest ani istotne, ani możliwe, więc tą opcję od razu odrzucamy.

Usuwamy czy zostawiamy? Dla potrzeb statystycznych zostawimy sobie te dane, bo mogą być ciekawe później, zawsze będziemy mieli możliwość odfiltrowania tych outlier'ów.

Także zauważamy, że pomysł porównywać wartości kategoryczne przy pomocy mediany był dobrym, bo wartość średnia psuje się przy takiej zamianie rozkładów i ograniczaniu wartości, a mediana zostaje się taką samą dopóki ta zmiana rozkładu nie zmienia połowę wartości.

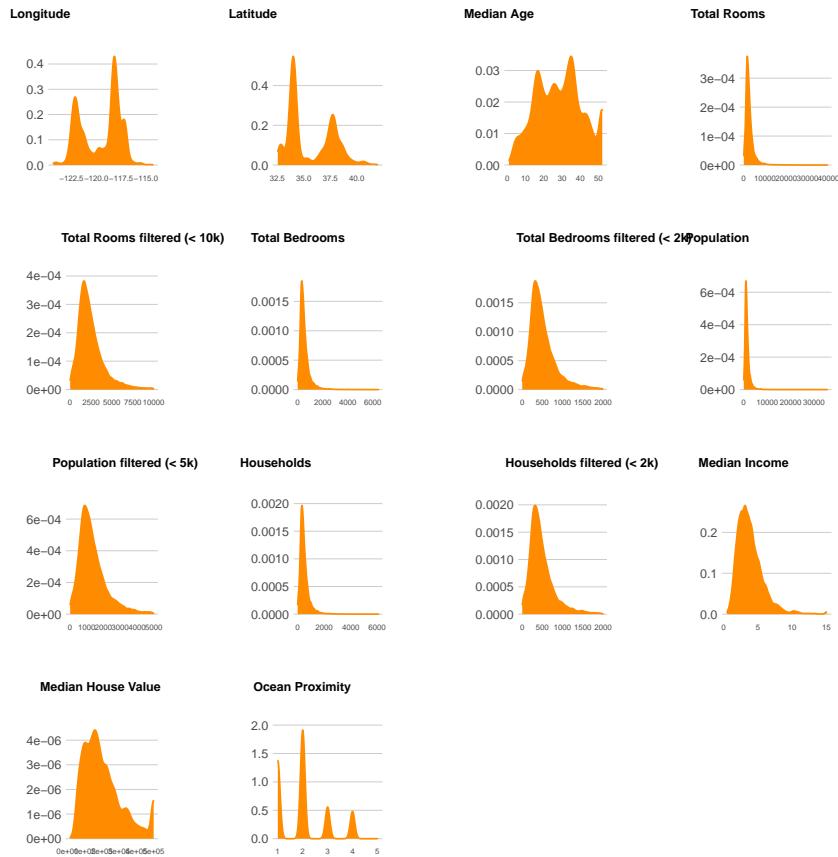
Otoż, wszystkie problemy zostały rozwiązane lub przynajmniej rozpatrzone, nasz zbiór danych już jest zmodyfikowany. Możemy go sobie zapisać jako odrębny plik, żeby potem nie powtarzać tego samego.

```
> write.csv(all_housing, file="cleaned_housing.csv", sep=",",
+           col.names=TRUE, row.names=FALSE)
```

4 Feature engineering

4.1 Tworzenie nowych cech

Otoż, spróbujmy sobie narysować rozkłady wszystkich cech:



Jak widzimy, dużo rozkładów mają ciężki prawy ogon, czyli są *right-tailed*, *heavy-tailed*. Musimy coś z tym zrobić jeżeli chcemy efektywnie testować i stosować dane z tych kolumn/cech. Także wcześniej zauważaliśmy, że **median_house_value** jest bardzo słabo powiązana z większością cech, co nie ułatwia nam życia. Więc spróbujmy rozwiązać od razu kilka problemów przy pomocy *feature engineering*, czyli tworzenie/modyfikacja/usuwanie kolumn tak, żeby dojść do stanu z lepszą wygodą dla nas.

Patrząc na podane nam kolumny, możemy śmiało stwierdzić, że **total_rooms**, **total_bedrooms** absolutnie nic nie wnoszą z praktycznego punktu widzenia, ponieważ bardzo zależą od **households**. Te duże wartości współczynników korelacji też naprawimy w ten sposób. Otóż, przetestujmy sobie takie nowe kolumny/cechy:

- **bedrooms_per_rooms** - ilość sypalni na pokój
- **rooms_per_household** - ilość pokojów na gospodarstwo
- **bedrooms_per_person** - ilość sypalni na osobę
- **rooms_per_person** - ilość pokojów na osobę
- **bedrooms_per_household** - ilość sypalni na gospodarstwo
- **people_per_household** - ilość osób na gospodarstwo

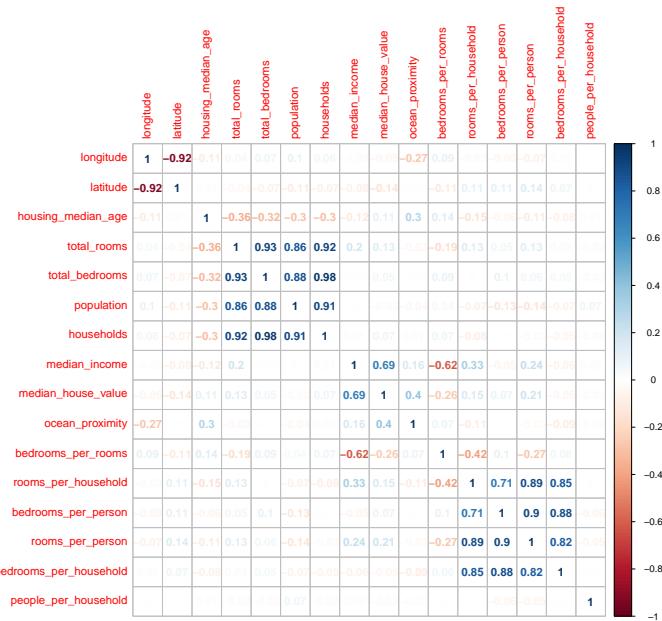
Oczywiście, że nie możemy zostawić wszystkie cechy, musimy usunąć ich tyle, żeby nie stracić informacji i jednocześnie jakąś cecha nie była zdefiniowana jako liniowa kombinacja innych cech.

Moglibyśmy stworzyć jeszcze i inne cechy. Dobrym pomysłem byłoby stworzenie cechy wyznaczającej odległość do najbliższego dużego miasta i podobne geograficzne cechy, bo możemy to zrobić stosując **longitude** i **latitude**. Ale to jest już nadmiar dla projektu naszej skali, więc kontynuujmy z danymi, które mamy.

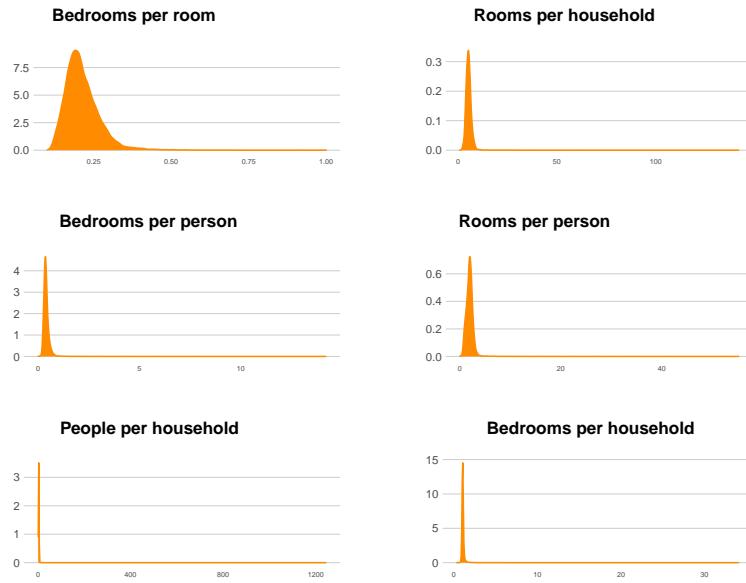
Otóż, stwórzmy takie nowe dane:

```
> housing$bedrooms_per_rooms <- housing$total_bedrooms / housing$total_rooms  
> housing$rooms_per_household <- housing$total_rooms / housing$households  
> housing$bedrooms_per_person <- housing$total_bedrooms / housing$population  
> housing$rooms_per_person <- housing$total_rooms / housing$population  
> housing$bedrooms_per_household <- housing$total_bedrooms / housing$households  
> housing$people_per_household <- housing$population / housing$households
```

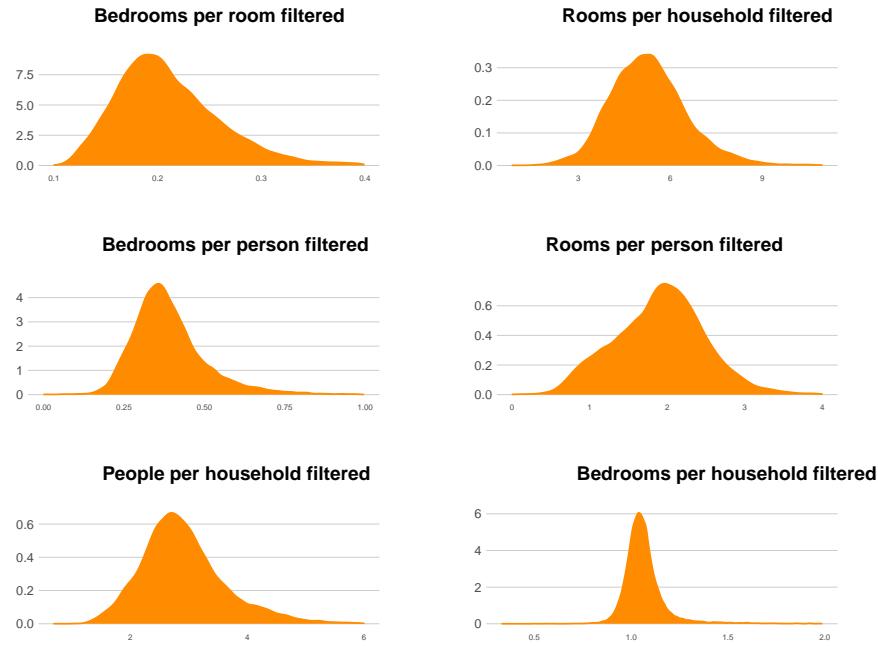
Sprawdźmy teraz macierz współczynników korelacji pomiędzy nowymi i stałymi cechami:



Także sprawdźmy jak wyglądają rozkłady nowych cech:



Widzimy, że jest dużo outlier'ów, więc na początku odfiltrować tak, żeby zobaczyć bliżej największą koncentrację punktów.



Otoż, musimy sobie wybrać cechy, które zostawiamy, a które zostaną wyrzucone.

Jak widzimy, **bedrooms_per_person** nic nie wnosią, współczynnik korelacji z **median_house_value** jest bardzo niski, więc wyrzucamy tą cechę. Dużo ciekawszym jest dla nas **bedrooms_per_rooms**, bo ma dość znaczący wsp. korelacji z interesującą nas cechą, a także nie ma bliskich do liniowych zależności z innymi cechami. Więc usuwamy wszystkie cechy powiązane z sypialniami, oprócz tej.

Także ciekawą dla nas cechą jest **rooms_per_person**, bo ma wsp. korelacji równy 0.21, i jest bardziej wygodną, niż **total_rooms**. Dodatkowo pokazuje ciekawsze dane. Zmieniamy **total_rooms** na nią.

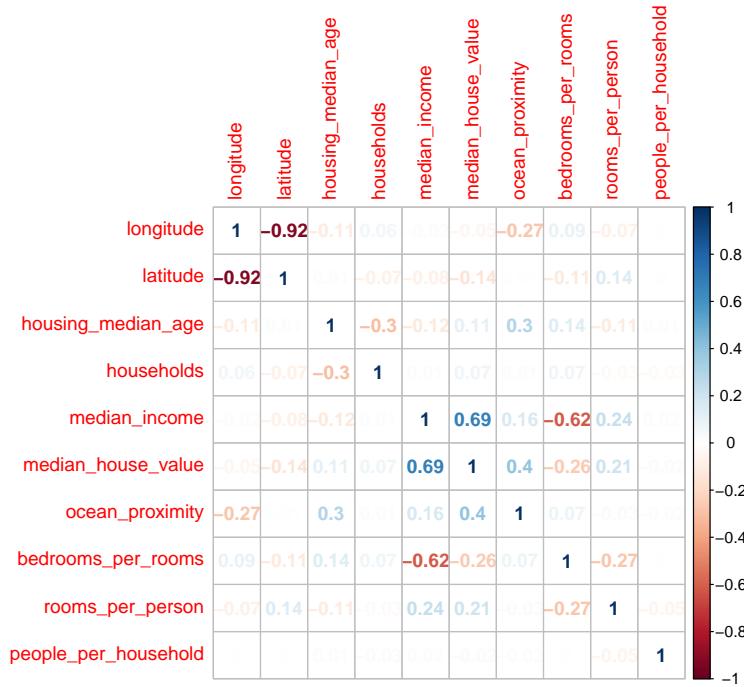
Inną ciekawą cechą nie w jakości znaczającej dla regresji, tylko tym, że ma niskie wsp. korelacji z cechami, bazując się na których, przewidujemy wartość **median_house_value**, także tym, że ma dość bliski do normalnego rozkład, przy najmniej tak to wygląda na wykresie.

Resztę niepotrzebnych cech usuwamy. Popatrzmy jak wygląda w tej chwili nasz zbiór danych.

```
> housingf <- dplyr::select(housing, -c('total_rooms', 'population', 'total_bedrooms',
+                                         'bedrooms_per_household', 'rooms_per_household',
+                                         'bedrooms_per_person'))
```

```
'data.frame': 20640 obs. of 10 variables:
$ longitude      : num -122 -122 -122 -122 -122 ...
$ latitude       : num 37.9 37.9 37.9 37.9 37.9 ...
$ housing_median_age : int 41 21 52 52 52 52 52 42 52 ...
$ households     : int 126 1138 177 219 259 193 514 647 595 714 ...
$ median_income   : num 8.33 8.3 7.26 5.64 3.85 ...
$ median_house_value: num 452600 358500 352100 341300 342200 ...
$ ocean_proximity: int 4 4 4 4 4 4 4 4 4 4 ...
$ bedrooms_per_rooms: num 0.147 0.156 0.13 0.184 0.172 ...
$ rooms_per_person : num 2.73 2.96 2.96 2.28 2.88 ...
$ people_per_household: num 2.56 2.11 2.8 2.55 2.18 ...
```

Także zobaczymy macierz wsp. korelacji po *feature engineering*.



Podsumowując, możemy stwierdzić, że pokonaliśmy dobrą robotę: nie mamy cech, które są silnie powiązane, oprócz **longitude** i **latitude**, ale to wynika z formy Kalifornii, i **bedrooms_per_rooms** i **median_income**, ale to nie wpłynie dość znacząco na nasz model, więc możemy ich zostawić. W końcu, mamy dużo nowych cech o dużo lepszej korelacji z **median_house_value**.

Zapisujemy zmodyfikowane dane do kolejnego pliku.

```
> write.csv(housingf, file="fe_housing.csv", sep=",",
+           col.names=TRUE, row.names=FALSE)
```

4.2 Dodatkowe możliwe zmiany

Pozostają nam kilka problemów, które zauważyliszy:

- *outliers* - wartości odstające dalej bardzo psują nam rozkłady, ale zostały omówione wcześniej
- *scaling* - nasze wartości są z różnych zakresów, z różną skalą, jest to niewygodne dla modelu i lepiej by to naprawić. Istnieją różne sposoby, niektóre z nich są opisane na [Wikipedii](#). Są to standaryzacja, skalowanie w zależności od średniej lub min-max i podobne. Nie będziemy tego robić tylko dlatego, żeby nie stracić same dane, a raczej żeby trzymać ich w początkowym stanie, bo trzymając parametry skalowania zawsze moglibyśmy ich przywrócić.
- *tail-heavy distributions* - jest to nasz największy problem, nawet gorszy od outlier'ów, bo ich zawsze da się usunąć, a tutaj nic nie możemy zrobić, bo to jest wpływ na dane wprost, co powoduje zmiany ich właściwości. Jednym ciekawym rozwiązaniem jest transformacja rozkładu. Jest to też trochę opisane na [Wikipedii](#).

5 Analiza danych

Otóz, doszliśmy do najważniejszej i najtrudniejszej części - analiza samych danych. Jest ona trudna ze względu na bardzo dużą ilość reguł różnorodnych testów, które są potrzebne do przeprowadzenia porządnej analizy. Tym my się właśnie i zajmiemy.

5.1 Wstęp

Musimy sobie dobrze określić reguły, które będziemy stosować dalej, żeby zawsze mieć możliwość odniesienia do nich.

Źródłem reguł został ten artykuł naukowy: *J. Uttley - Power Analysis, Sample Size, and Assessment of Statistical Assumptions—Improving the Evidential Value of Lighting Research* (2019). Można znaleźć całość [tutaj](#). Najbardziej interesują nas punkty 3.1 - 3.4, które są na stronach 5 - 11 w dokumencie, który znajduje się pod przedstawionym linkiem. Opisują one jakie założenia stosujemy do parametrycznych testów i nieparametrycznych, i jak je sprawdzić wraz z możliwymi problemami przy ich stosowaniu. Więc, ja spróbuję tu krótko określić jak będziemy postępowali dalej, bazując się na tym artykule naukowym.

Odnosząc się do Tabeli nr 3 w podanym artykule, możemy sobie rozdzielić najczęściej stosowane testy na parametryczne i nieparametryczne.

Assessment being made	Parametric test	Nonparametric test
Compare one group to a hypothetical value	One-sample <i>t</i> -test	Wilcoxon signed rank test
Compare two independent groups	Independent <i>t</i> -test	Mann-Whitney test
Compare three or more independent groups	One-way ANOVA	Kruskal-Wallis test
Compare two dependent groups (within subjects)	Dependent <i>t</i> -test	Wilcoxon signed rank test
Compare three or more dependent groups	Repeated measures ANOVA	Friedman test
Association between two variables	Pearson correlation	Spearman correlation
Predict value based on another value	Linear/nonlinear regression	Nonparametric regression/logistic regression

Do tej tabeli później będziemy się odwoływać, więc jest bardzo ważna dla nas.

Nie określiliśmy różnicę pomiędzy testami parametrycznymi i nieparametrycznymi. Otóż, parametryczne testy są bardziej precyzyjne, ale wymagają spełnienia pewnych założeń stosujących się samych danych. Bez spełnienia ich, nie możemy stwierdzić, że wynik otrzymany z tego testu jest odpowiadający rzeczywistości (nigdy tego nie możemy stwierdzić, bo statystyka nigdy nie określa czegoś na 100%, ale wtedy niepewność jest bardzo duża). Właśnie możliwe problemy związane z niedotrzymaniem tych założeń są przedstawione w punkcie 3.4 w podanym artykule. Także w punkcie 3.1 jest podana informacja, że z 50 sprawdzonych artykułów naukowych tylko 22% mieli sprawdzone te założenia, co sprawia, że wyniki w nich podane mogą silnie różnić się od rzeczywistości. To potwierdza ważność testowania założeń i jednocześnie pokazuje jak rzadko to jest robione w akademice, i to jest duży problem. W punkcie 3.5 są przedstawione przykłady takiego niedotrzymania reguł. Ale nie o tym tutaj mówimy, więc będziemy starać się w miarę możliwości przeprowadzać wszystkie możliwe badanie przed stwierdzeniem czegokolwiek.

Z tabeli nr 2 podanego artykułu określamy najczęstsze założenia testu parametrycznego.

Assumption	Description
Data are measured at least at interval level	The response or property being measured should be recorded using a dependent variable on an interval scale, minimum, or on a continuous scale. The intervals on the scale should represent differences of equal magnitude. For example, if a 1–5 rating scale is used to measure a participant's perceived brightness of a space, the difference in perceived brightness between ratings of 1 and 2 should be the same as it is between ratings of 4 and 5
Data are independent	Data from one participant should not influence data from another participant, which can be addressed through randomization in experimental design. In within-subjects designs, we do not expect the responses from the same participant to be independent, but responses between different participants in within-subjects designs should be independent. In regression analysis, the errors in the regression model should also be uncorrelated
Data are normally distributed	The raw data within each condition approximate a normal distribution or the residuals (individual minus the mean value) approximate a normal distribution, depending on the type of test being carried out
Variance is the same throughout the data	When comparing more than one group of participants, each of these groups should have approximately equal variance. If carrying out a correlation, the variance of one of your variables should be stable at all levels of the other variable. This is known as homogeneity of variance, or homoscedasticity, particularly in relation to regression analysis. In within-subjects designs with three or more conditions, an assumption of sphericity is also made. Sphericity refers to the variances of the differences between pairs of conditions being equal across all combinations of conditions

Z podanej tablicy najbardziej nas ciekawią ostatni dwa punkty, ponieważ pierwsze dwa muszą być uwzględnione jeszcze przy tworzeniu samych danych, co do ostatnich dwóch, to właśnie ich możemy sprawdzić przed stosowaniem testów i artykuł nam podaje całą listę sposobów na to, które będziemy stosować.

Otóż, przejdźmy już do analizy podstawowej naszego zbioru danych.

5.2 Analiza podstawowa

Trochę już wcześniej zostały przedstawione te dane, więc bez zbędnego opisu obliczmy sobie wszystkie momenty i inne elementy statystyki opisowej.

```
> head(housing, 4)

  longitude latitude housing_median_age households median_income median_house_value
1 -122.23    37.88             41        126      8.3252        452600
2 -122.22    37.86             21       1138      8.3014        358500
3 -122.24    37.85             52        177      7.2574        352100
4 -122.25    37.85             52        219      5.6431        341300

ocean_proximity bedrooms_per_rooms rooms_per_person people_per_household
1                  4          0.1465909      2.732919      2.555556
2                  4          0.1557966      2.956685      2.109842
3                  4          0.1295160      2.957661      2.802260
4                  4          0.1844584      2.283154      2.547945

> summary(housing)

  longitude      latitude      housing_median_age     households      median_income
Min.   : -124.3  Min.   :32.54  Min.   : 1.00  Min.   : 1.0  Min.   : 0.4999
1st Qu.: -121.8  1st Qu.:33.93  1st Qu.:18.00  1st Qu.: 280.0  1st Qu.: 2.5634
Median : -118.5  Median :34.26  Median :29.00  Median : 409.0  Median : 3.5348
Mean   : -119.6  Mean   :35.63  Mean   :28.64  Mean   : 499.5  Mean   : 3.8707
3rd Qu.: -118.0  3rd Qu.:37.71  3rd Qu.:37.00  3rd Qu.: 605.0  3rd Qu.: 4.7432
Max.   : -114.3  Max.   :41.95  Max.   :52.00  Max.   :6082.0  Max.   :15.0001

  median_house_value ocean_proximity bedrooms_per_rooms rooms_per_person
Min.   : 14999  Min.   :1.000  Min.   :0.1000  Min.   : 0.00255
1st Qu.:119600  1st Qu.:1.000  1st Qu.:0.1754  1st Qu.: 1.52238
Median :179700  Median :2.000  Median :0.2033  Median : 1.93794
Mean   :206856  Mean   :2.034  Mean   :0.2131  Mean   : 1.97697
3rd Qu.:264725  3rd Qu.:2.000  3rd Qu.:0.2399  3rd Qu.: 2.29609
Max.   :500001  Max.   :5.000  Max.   :1.0000  Max.   :55.22222

  people_per_household
Min.   : 0.6923
1st Qu.: 2.4297
Median : 2.8181
Mean   : 3.0707
3rd Qu.: 3.2823
Max.   :1243.3333

> # all the descriptive values (look at the skew and kurt, should see tail-heavy)
> describeBy(housing)

      vars      n      mean       sd      median      trimmed      mad      min
longitude      1 20640 -119.57    2.00     -118.49     -119.52    1.90 -124.35
latitude       2 20640   35.63    2.14      34.26      35.51    1.82   32.54
```

housing_median_age	3	20640	28.64	12.59	29.00	28.49	14.83	1.00
households	4	20640	499.54	382.33	409.00	441.20	223.87	1.00
median_income	5	20640	3.87	1.90	3.53	3.65	1.58	0.50
median_house_value	6	20640	206855.82	115395.62	179700.00	192773.00	101409.84	14999.00
ocean_proximity	7	20640	2.03	0.94	2.00	1.92	1.48	1.00
bedrooms_per_rooms	8	20640	0.21	0.06	0.20	0.21	0.05	0.10
rooms_per_person	9	20640	1.98	1.15	1.94	1.92	0.57	0.00
people_per_household	10	20640	3.07	10.39	2.82	2.86	0.62	0.69
			max	range	skew	kurtosis	se	
longitude		-114.31	10.04	-0.30	-1.33	0.01		
latitude		41.95	9.41	0.47	-1.12	0.01		
housing_median_age		52.00	51.00	0.06	-0.80	0.09		
households		6082.00	6081.00	3.41	22.05	2.66		
median_income		15.00	14.50	1.65	4.95	0.01		
median_house_value		500001.00	485002.00	0.98	0.33	803.22		
ocean_proximity		5.00	4.00	0.73	-0.29	0.01		
bedrooms_per_rooms		1.00	0.90	2.24	14.23	0.00		
rooms_per_person		55.22	55.22	17.77	600.46	0.01		
people_per_household		1243.33	1242.64	97.63	10647.40	0.07		

```

> # calculating all the moments
> all.moments(housing)

[,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
[1,] 1.0000  1.00000  1.00000  1.0000  1.0000000 1.000000e+00 1.0000000 1.0000000
[2,] -119.5697 35.63186 28.63949 499.5397 3.870671 2.068558e+05 2.034012 0.2131251
[3,] 14300.9282 1274.19162 978.60877 395708.8499 18.591242 5.610483e+10 5.028198 0.0487909
[,9]      [,10]
[1,] 1.000000  1.000000
[2,] 1.976970  3.070655
[3,] 5.221706 117.293722

> # calculating all the central moments
> all.moments(housing, central=TRUE)

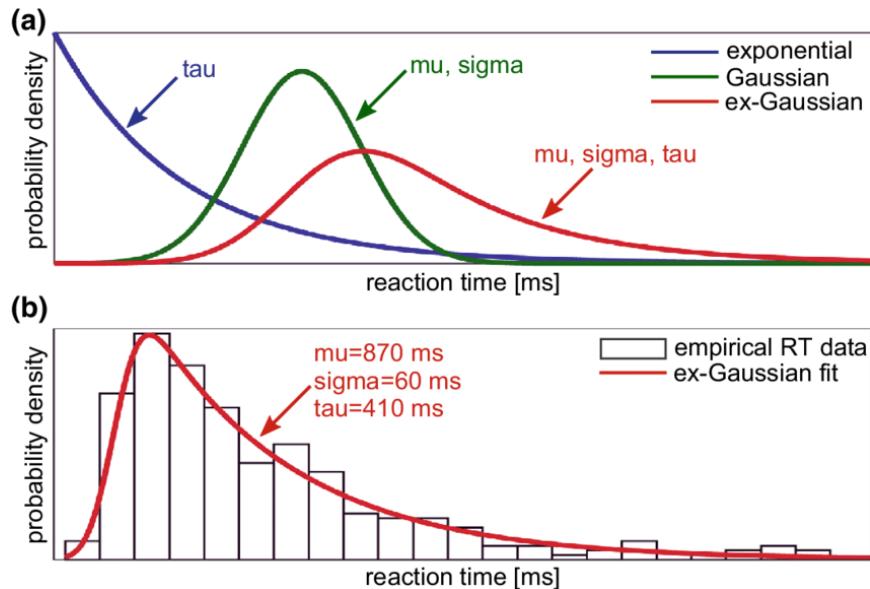
[,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
[2,] -2.928235e-15 -2.239724e-15 4.516628e-16 2.767261e-14 1.072893e-16 -1.107186e-11
[3,] 4.013945e+00 4.562072e+00 1.583886e+02 1.461690e+05 3.609148e+00 1.331550e+10
[,7]      [,8]      [,9]      [,10]
[1,] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
[2,] 9.501444e-17 -4.412787e-18 -9.759097e-17 -1.772269e-16
[3,] 8.909944e-01 3.368585e-03 1.313298e+00 1.078648e+02

```

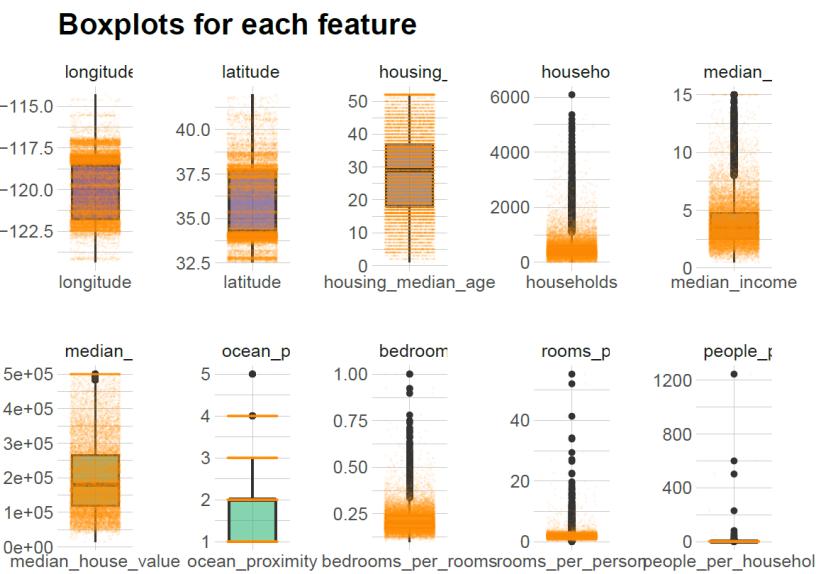
Mały komentarz: [1,] odpowiada za zerowy moment, [2,] za pierwszy i td...

Jak i zostało zaznaczono w komentarzu, jeżeli popatrzyć na współczynnik skośności i kurtozę, to można zobaczyć, że są one w większości swojej dodatnie,

co powoduje, że mają duży ogon, czyli są rozciagnięte bardziej w prawo, bardzo podobnie do rozkładu ex-Gaussian, czyli eksponencjalno-normalnego. Niżej jest on przedstawiony.



Narysujmy sobie jeszcze boxploty do każdej cechy.



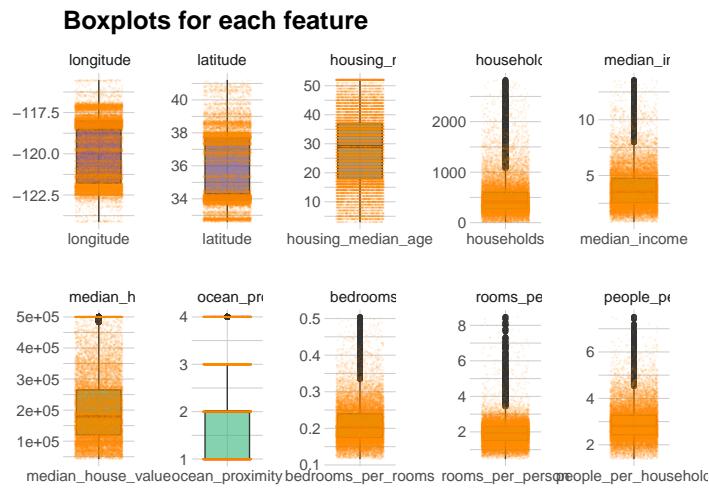
Widzimy dużą ilość outlier'ów, które psują nam skalę, a także wartości różnych statystyk, więc stwórzmy sobie zbiór bez nich.

```
> # replacing outliers with NAs to remove them while melting
> outlier.replace <- function(x){
+   quantiles <- quantile( x, c(.003, .997) )
+   x[ x < quantiles[1] ] <- NA
+   x[ x > quantiles[2] ] <- NA
+   return(x)
+ }
> no_outs <- housing
> for(i in names(no_outs)){
+   no_outs[[i]] <- outlier.replace(no_outs[[i]])
+ }
> # outliers per column
> colSums(is.na(no_outs))
```

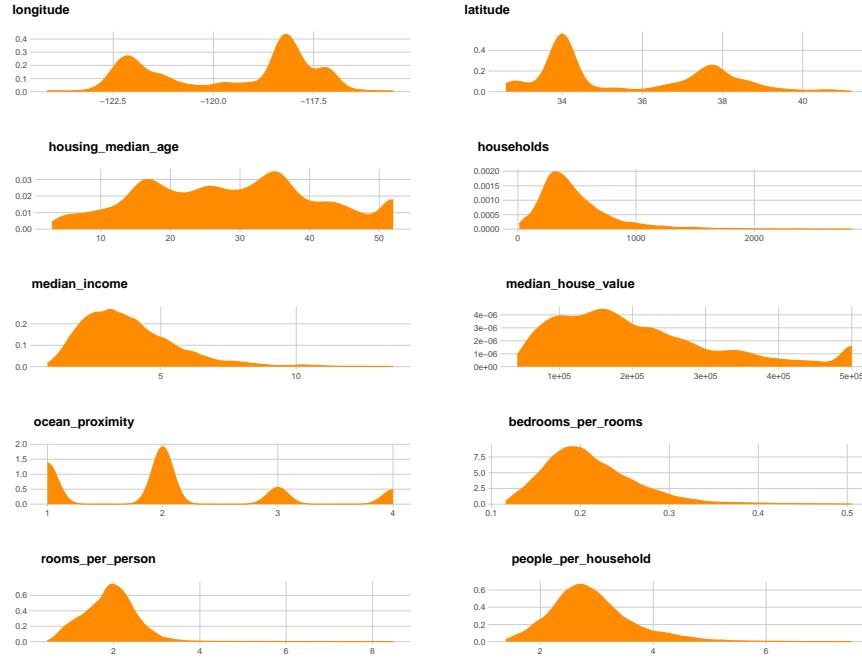
	longitude	latitude	housing_median_age	households
median_income	117	120	62	124
median_house_value	124	54	ocean_proximity	bedrooms_per_rooms
rooms_per_person	people_per_household	124	5	124
people_per_household	124	124		

Także warto spojrzeć na *trimmed*, ponieważ to pokazuje jak zmieniają się wartości po usunięciu outlier'ów, i widzimy, że w przypadku niektórych cech to jest dość silna zmiana.

Narysujmy teraz sobie nowe boxploty.



Narysujmy sobie jeszcze gęstości rozkładów *density plots*, czyli tak naprawdę rozmażane histogramy.



5.3 Analiza rodzaju rozkładu badanych cech

Wydzielimy sobie 3 najciekawsze dla nas cechy: **bedrooms_per_rooms**, **median_income** i **median_house_value**. Oceńmy na ile one odpowiadają rozkładu normalnemu, żeby potem móc przeprowadzać różnorodne testy na nich i mieć możliwość stosowania estymatorów.

Zróbnmy sobie plan testowania normalności:

- Wizualne testowanie
 - Histogram
 - QQ-plot
 - Boxplot
- Oszacowanie statystyk opisowych
 - Współczynnik skośności
 - Kurtoza
- Statystyczne testy odchylenia od rozkładu normalnego (niżej przedstawione możliwe testy)

- Shapiro-Wilk test
- Kolmogorov-Smirnov test
- Anderson-Darling test
- D'Agostino-Pearson omnibus test
- Jarque-Bera test
- i inne...

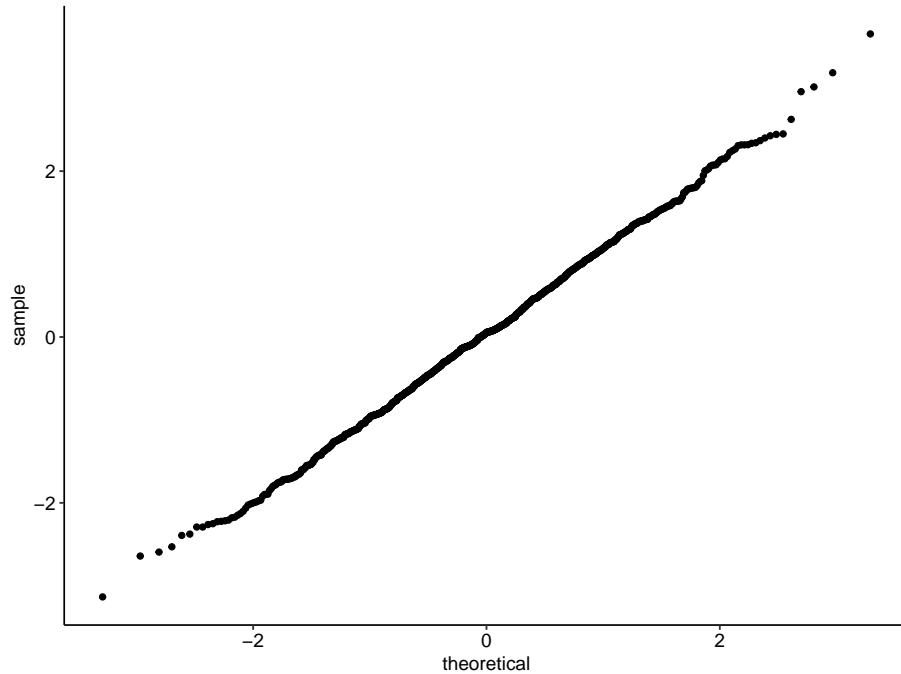
Także warto zauważyc, że w ogólnosci stosujemy *density plots*, poniewaz są ciekawsze i przyjemniejsze wizualnie. Ale w tym rozdziale będziemy stosować histogram.

Parametr histogramu, tak zwany *binwidth* może być wyznaczony z reguły Freedman-Diaconis, która określa go jako:

$$\text{binwidth} = \frac{2IQR}{\sqrt[3]{n}}$$

Także będziemy stosować QQ-plot. Niżej jest wygląd tego wykresu dla danych o rozkładzie normalnym (czyli to jest prosta).

```
> ggplot(data.frame(d = rnorm(1000)), aes(sample=d)) + stat_qq()
```



Podczas stosowania statystyki opisowej do sprawdzenia normalności będziemy posługiwać się *z*-statystyką bazującą się na współczynniku skośności i kurtozie. Jest ona wyznaczona jako:

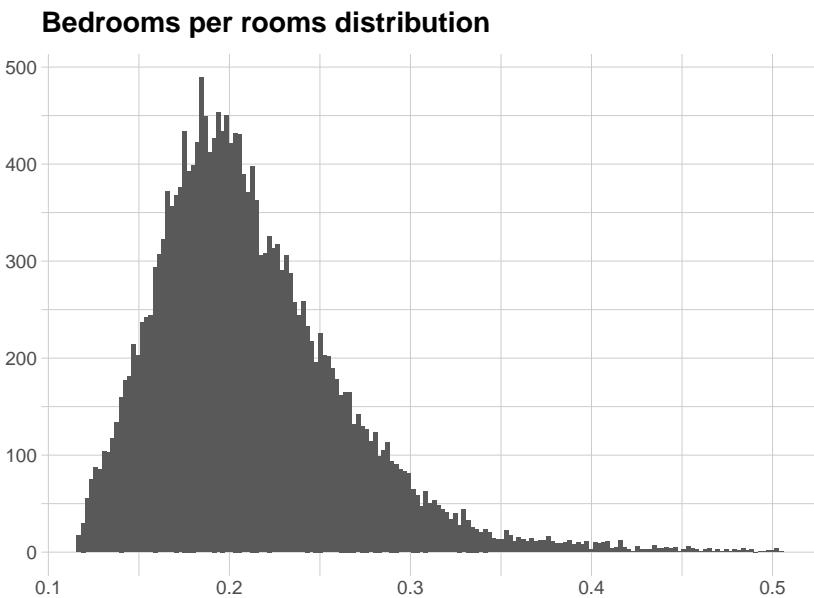
$$z = \frac{skew}{2SE} \quad z = \frac{kurtosis}{2SE}$$

I w końcu, ze wszystkich pokazanych wyżej możliwych testów na normalność będziemy stosowali *Shapiro-Wilk test*, ponieważ jest on bardziej czutliwy i mocny, zgodnie z pracami cytowanymi w artykule. A także warto zauważyc, że powinien być przeprowadzany na próbkach o rozmiarze nie większym od 50. Jest to związane, że w opisie tego testu od 1965 roku przez Shapiro i Wilk'a byli stosowane takie ilości, bo nie mieli dostępu do większych. Jest to też opisane w artykule. O ile mamy dużą ilość rekordów, to nie możemy wprost polegać na przypadkowości wybierania próbki. Dlatego będziemy robić po 3 testy dla różnych próbek 50-elementowych.

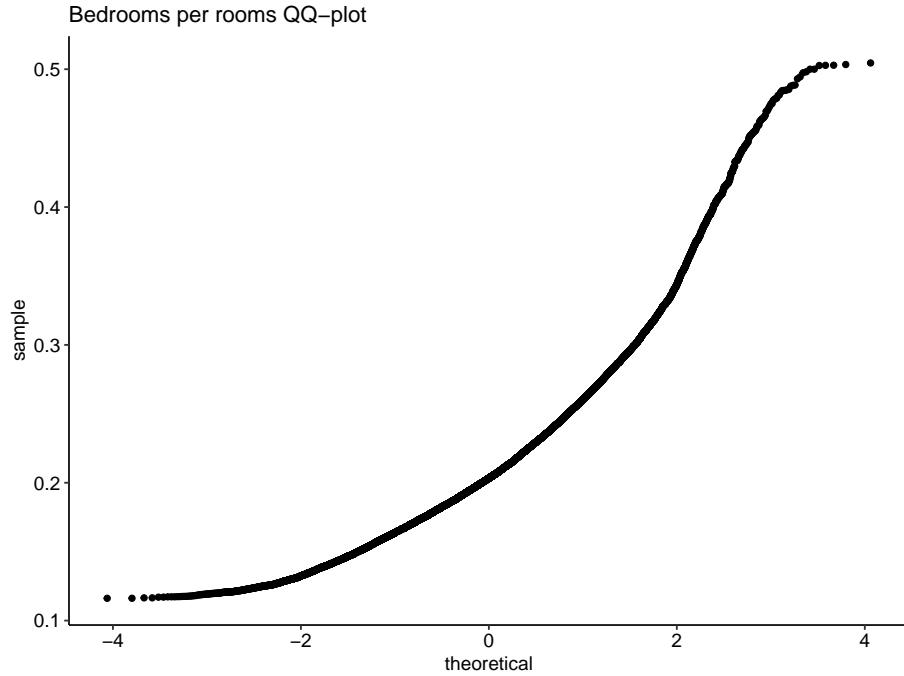
Podczas przeprowadzenia testu zakładamy, że H_0 - hipoteza zerowa - stwierdza, że rozkład jest normalnym, a H_1 - hipoteza alternatywna - stwierdza, że rozkład takim nie jest.

5.3.1 bedrooms_per_rooms

```
> gap <- IQR(no_outs$bedrooms_per_rooms, na.rm=TRUE) /
+   (length(no_outs$bedrooms_per_rooms)) ^ (1/3))
> no_outs %>% ggplot(aes(x=bedrooms_per_rooms)) + plot.properties +
+   geom_histogram(binwidth = gap) +
+   labs(title='Bedrooms per rooms distribution')
```



Z histogramu widać ten ogon, i możemy już przewidywać, że rozkład nie będzie rozkładem normalnym, ale przeprowadźmy dalszą analizę.



Widzimy bardzo duże odchylenie od prostej, więc możemy być pewnymi, że to nie jest rozkład normalny, ale przeprowadźmy dalszą analizę, żeby sobie dodatkowo zobaczyć jak będzie się zachowywał rozkład nie normalny. Dla innych cech potem, jeżeli zobaczymy, że zbiór nie odpowiada rozkładowi normalnemu, to nie będziemy kontynuowali obliczeń dla niego.

Następnym punktem jest boxplot, ale został już wyżej przedstawiony. Punkty pomiędzy 1 i 3 kwartylem są dość podobnie rozłożone odnośnie mediany, wąsy są dość blisko długości $1.5IQR$, chociaż wyraźnie widać, że górny jest troszeczkę większy. Co pokazuje nienormalność - to ilość outlier'ów i ich tendencja pojawiać się z góry, czyli nie ma dolnych outlier'ów.

Dalej testujemy wartość z -statystyki bazującej się na współczynniku skośności i kurtozie. Powinien być testowany zgodnie z artykułem na próbkach nie większych od 200.

```
> stat.desc(sample(no_outs$bedrooms_per_rooms, 150), norm=TRUE)
```

	nbr.val	nbr.null	nbr.na	min	max	range	sum
	1.490000e+02	0.000000e+00	1.000000e+00	1.189563e-01	4.735140e-01	3.545577e-01	3.188117e+01
	median	mean	SE.mean	CI.mean.0.95	var	std.dev	coef.var
	2.010187e-01	2.139676e-01	5.123579e-03	1.012482e-02	3.911408e-03	6.254125e-02	2.922931e-01

```

skewness      skew.2SE      kurtosis      kurt.2SE      normtest.W      normtest.p
1.523940e+00  3.834990e+00  3.441771e+00  4.358216e+00  8.890042e-01  3.625958e-09

```

Z tych wyników najbardziej ciekawią nas **skew.2SE** i **kurt.2SE**. Dla małych próbek i $\alpha = 0.05$ prógiem są wartości ± 1.96 , ale zwiększeniem próbki zmniejsza się SE , więc można zwiększać próg. Oczywiście ten próg nie jest wartością deterministyczną, i stosowany może zostać wybrany na podstawie naszej opinii. W tym przypadku dostajemy, że obie wartości są > 3.6 , co jest dość znaczącą wartością dla próbki rozmiarem 150. Więc widzimy odchylenie od rozkładu normalnego.

Został nam się test Shapiro-Wilk'a.

```

> shapiro.test(sample(no_outs$bedrooms_per_rooms, 50))

Shapiro-Wilk normality test

data: sample(no_outs$bedrooms_per_rooms, 50)
W = 0.80651, p-value = 1.243e-06

> shapiro.test(sample(no_outs$bedrooms_per_rooms, 50))

Shapiro-Wilk normality test

data: sample(no_outs$bedrooms_per_rooms, 50)
W = 0.95984, p-value = 0.09331

> shapiro.test(sample(no_outs$bedrooms_per_rooms, 50))

Shapiro-Wilk normality test

data: sample(no_outs$bedrooms_per_rooms, 50)
W = 0.94154, p-value = 0.01551

```

Jak widzimy, p wartość jest generalnie mała. (*O ile za każdym razem komplikacji sprawozdania zmienia się próbka to trudno jest komentować otrzymane wyniki*). Jednego razu wyskoczyła p wartość równa 0.89, co właśnie pokazuje, że zrobiliśmy dobrze robiąc kilka testów jednocześnie. W tym wartość statystyki W jest dość duża, ale nie dostatecznie (> 0.98), żeby móc trochę zmieniać próg p wartości.

Otoż, pokazaliśmy, że rozkład badanej cechy **nie jest normalnym**.

Spróbujmy transformować cechę. Może wtedy dostaniemy rozkład normalny.

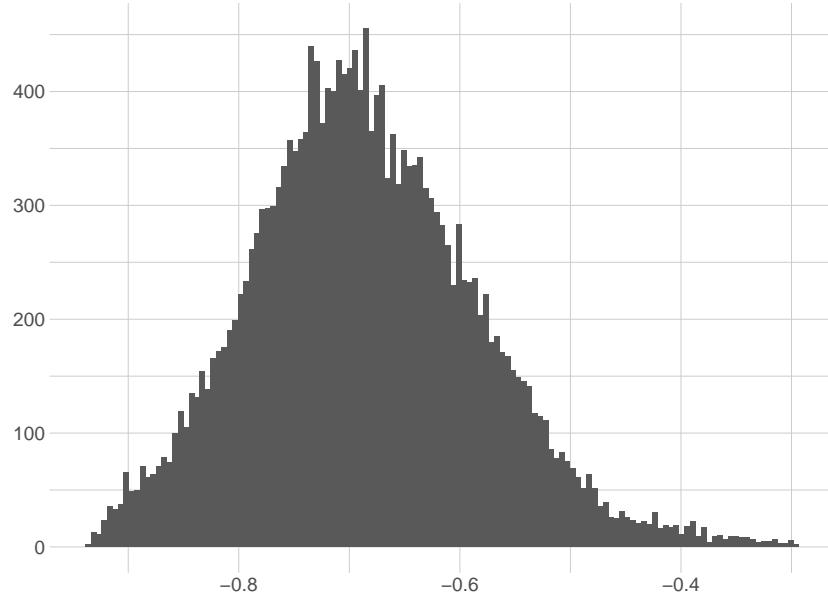
5.3.2 *bedrooms_per_rooms* - log10 transformacja

Dla początku przeprowadźmy samą transformację.

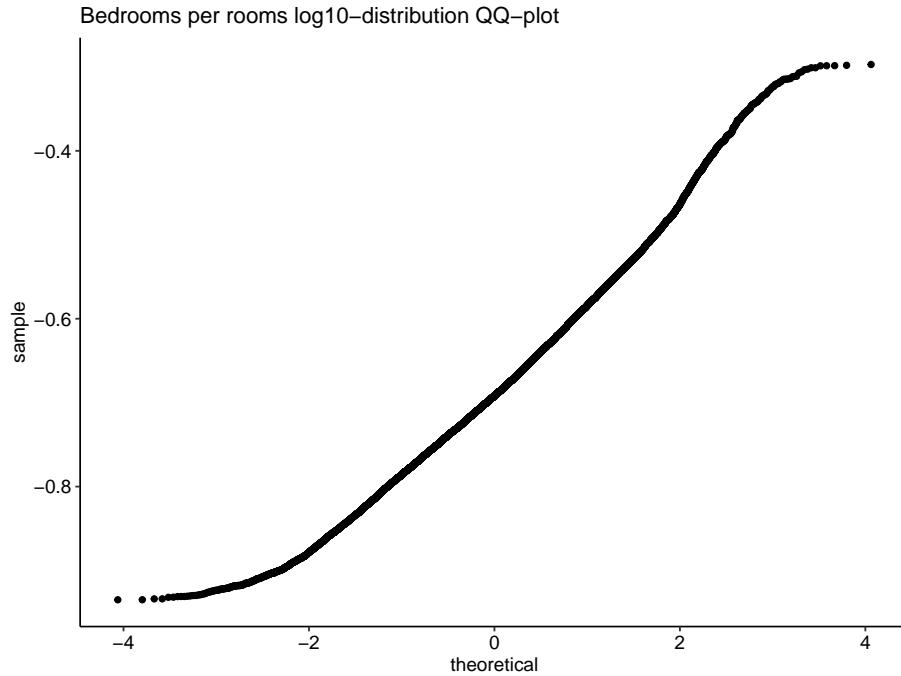
```
> # log10 distr (could be ln, but the difference is only the constant)
> bprlog <- data.frame(d = log10(no_outs$bedrooms_per_rooms))
```

```
$title
[1] "Bedrooms per rooms log10-distribution"

attr(,"class")
[1] "labels"
```



Ooo, dużo lepszy histogram wyszedł nam, nie widać dużego ogonu, wygląda jako rozkład normalny. Zobaczmy co otrzymamy dalej.



Widzimy pewne wykrzywienie w QQ-plot, ale to jeszcze nie jest powód ostatecznie stwierdzić, że to nie jest rozkładem normalnym. Idziemy dalej.

Nie będziemy rysowali boxplot, lepiej przejdźmy od razu do liczb, bo wykresy są dość podobne do rozkładu normalnego, więc nie możemy nic stwierdzić w żadną stronę.

```
> stat.desc(sample(bprlog$d, 150), norm=TRUE)
```

	nbr.val	nbr.null	nbr.na	min	max	range
	1.480000e+02	0.000000e+00	2.000000e+00	-8.951653e-01	-4.479377e-01	4.472276e-01
	sum	median	mean	SE.mean	CI.mean.0.95	var
	-1.029235e+02	-7.011664e-01	-6.954291e-01	8.480609e-03	1.675966e-02	1.064427e-02
	std.dev	coef.var	skewness	skew.2SE	kurtosis	kurt.2SE
	1.031711e-01	-1.483560e-01	2.179415e-01	5.466417e-01	-6.709773e-01	-8.468737e-01
	normtest.W	normtest.p				
	9.831007e-01	6.606263e-02				

Wartości, które nas ciekawią są dość małe, więc póki co ten rozkład odpowiada normalnemu.

```
> shapiro.test(sample(bprlog$d, 50))
```

Shapiro-Wilk normality test

```
data: sample(bprlog$d, 50)
W = 0.96903, p-value = 0.2215
```

```

> shapiro.test(sample(bprlog$d, 50))

Shapiro-Wilk normality test

data: sample(bprlog$d, 50)
W = 0.97103, p-value = 0.2548

> shapiro.test(sample(bprlog$d, 50))

Shapiro-Wilk normality test

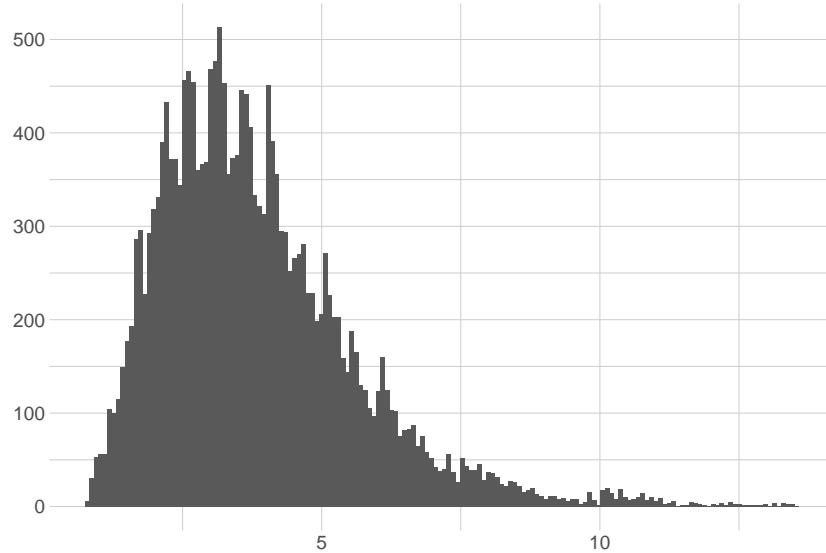
data: sample(bprlog$d, 50)
W = 0.9775, p-value = 0.4519

```

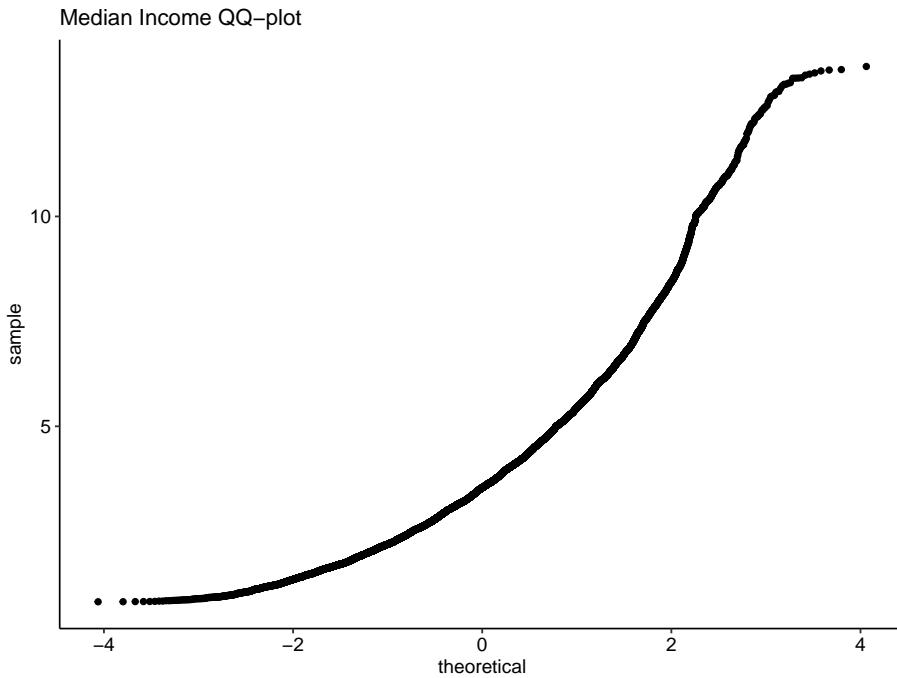
Jest ten sam problem, komplikacja sprawozdania zmienia wartości, tak będzie i dalej przy losowaniu n-elementowej próbki. Widzimy, że p wartość jest dość wysoka, więc nie mamy podstaw do odrzucenia zerowej hipotezy. Więc, możemy stwierdzić, że ten rozkład jest bliskim do normalnego.

5.3.3 median_income

Median Income distribution



Widzimy bardzo duży ogon, sprawdźmy jeszcze QQ-plot, żeby przekonać się, i możemy odrzucić od razu.



Widzimy duże odchylenie od prostej, jeszcze większe, niż dla pierwszej cechy.
Stwierdzamy, że dany rozkład nie jest rozkładem normalnym i odrzucamy go.
Spróbujmy transformować cechę. Może wtedy dostaniemy rozkład normalny.

5.3.4 *median_income* - log10 transformacja

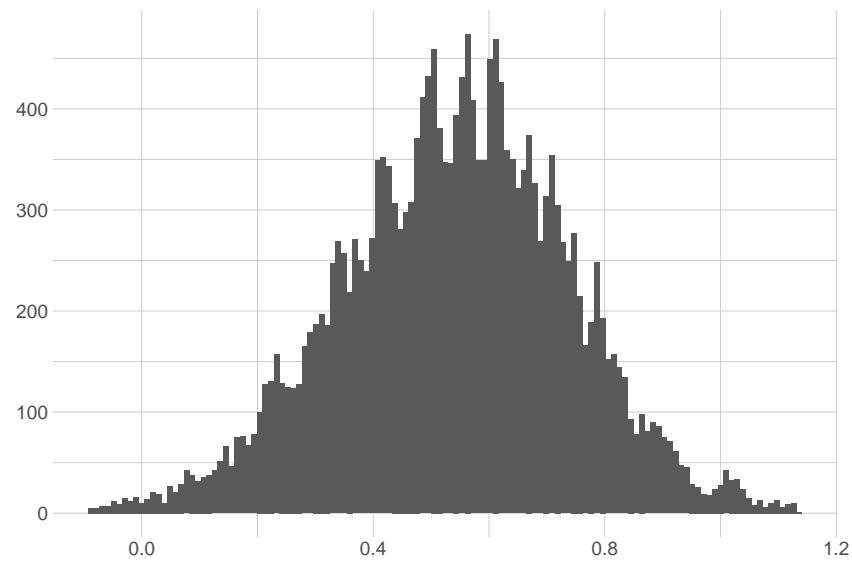
Dla początku przeprowadźmy samą transformację.

```
> # log10 distr (could be ln, but the difference is only the constant)
> milog <- data.frame(d = log10(no_outs$median_income))
```

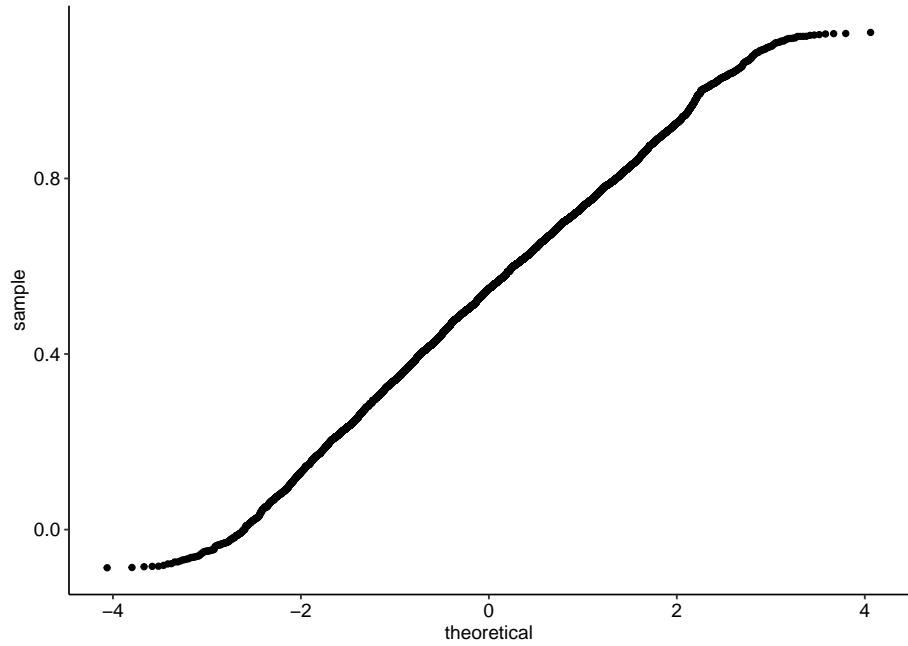
Rozkład z nast. strony jest dość podobny do normalnego, chociaż widać niektóre zaburzenia, ale one są równomiernie rozłożone.

Widać prostą linię na QQ-plot z nast. strony z pewnymi odkształceniami na końcach, ale to nie gra dużej roli.

Median Income log10-distribution



Median Income log10-distribution QQ-plot



```
> stat.desc(sample(milog$d, 150), norm=TRUE)

  nbr.val    nbr.null    nbr.na      min      max    range     sum
150.000000000  0.00000000  0.00000000  0.12525349  1.04497462  0.91972113 83.46741741
      median      mean    SE.mean CI.mean.0.95      var   std.dev  coef.var
  0.56465381  0.55644945  0.01520418  0.03004366  0.03467506  0.18621240  0.33464387
  skewness    skew.2SE   kurtosis  kurt.2SE  normtest.W  normtest.p
 -0.13934383 -0.35181019  0.06036220  0.07668291  0.98639182  0.14873843
```

Wartości, które nas ciekawią są dość niskie.

```
> shapiro.test(sample(milog$d, 50))
```

Shapiro-Wilk normality test

```
data: sample(milog$d, 50)
W = 0.95119, p-value = 0.0382
```

```
> shapiro.test(sample(milog$d, 50))
```

Shapiro-Wilk normality test

```
data: sample(milog$d, 50)
W = 0.98217, p-value = 0.6463
```

```
> shapiro.test(sample(milog$d, 50))
```

Shapiro-Wilk normality test

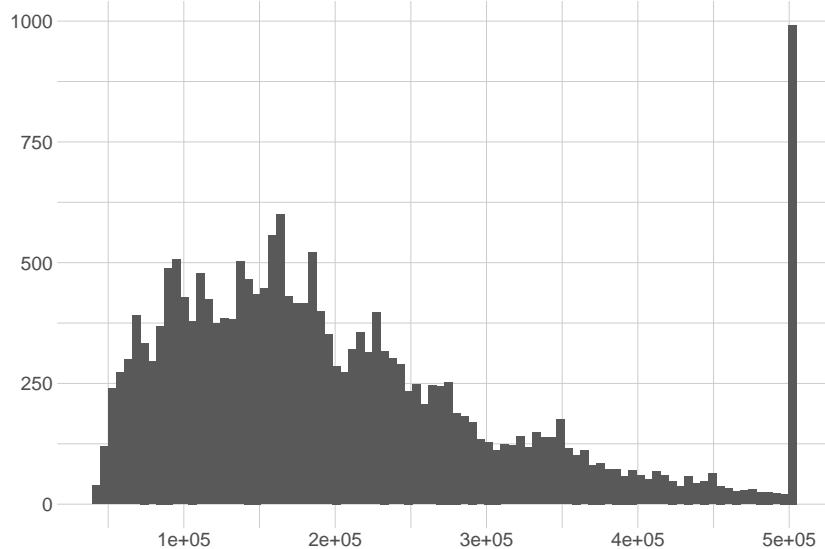
```
data: sample(milog$d, 50)
W = 0.97355, p-value = 0.3331
```

Widzimy duże wartości p , w tym i duże wartości W , więc stwierdzamy, że log10-transformowany rozkład tej cechy, jest rozkładem normalnym.

5.3.5 median_house_value

```
> gap <- IQR(no_outs$median_house_value, na.rm=TRUE) /  
+   (length(na.omit(no_outs$median_house_value)) ^ (1/3))  
> no_outs %>% ggplot(aes(x=median_house_value)) + plot.properties +  
+   geom_histogram(binwidth = gap) +  
+   labs(title='Median House Value distribution')
```

Median House Value distribution



Rozkład nawet blisko nie jest podobny do rozkładu normalnego. Bardzo psuje widok duża ilość wartości równych 500001 (skąd się pojawiły było opisane wcześniej). Ale to nie zmienia faktu, że nawet środkowa część nie jest rozkładem normalnym. Odrzucamy od razu.

5.3.6 median_house_value - log10 transformacja

Dla początku przeprowadźmy samą transformację.

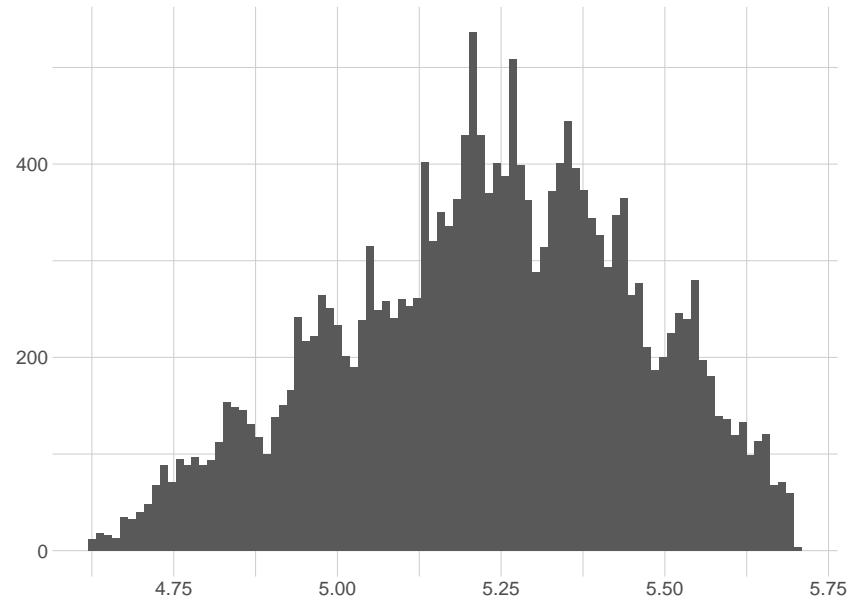
```
> # log10 distr (could be ln, but the difference is only the constant)  
> mhvlog <- data.frame(d = log10(no_outs$median_house_value))
```

Dodatkowo odrzucimy te outlier'y. $\log_{10}(500001) = 5.69897\dots$

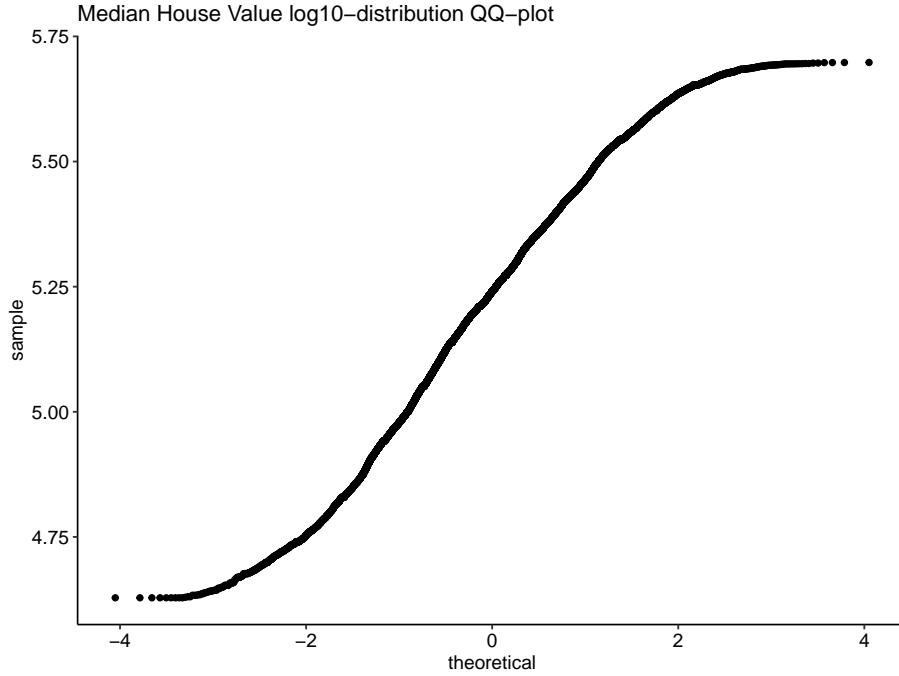
```
> mhvlog <- mhvlog %>% filter(d < 5.698)
```

```
$title  
[1] "Median House Value log10-distribution"
```

```
attr(,"class")  
[1] "labels"
```



To nam bardzo słabo przypomina rozkład normalny. Spróbujmy jedzce na-rysować QQ-plot.



Mamy linię bliską do prostej, ale z dużymi wykrzywieniami na końcach. Możemy uważać ten rozkład za bliski normalnemu, ale nie normalnym. Czyli przy stosowaniu testów, możemy być pewnymi, że możliwy błąd nie będzie drastycznym.

5.4 Estymatory

Stwórzmy sobie wygodną strukturę do przechowywania estymowanych wartości.

```
> estimations <- data.frame(mean=double(),
+                               meanlow=double(),
+                               meanup=double(),
+                               var=double(),
+                               varlow=double(),
+                               varup=double()
+ )
```

Wzór estymatora punktowego nieobciążonego dla wartości średniej:

$$\bar{x} = \frac{1}{n} \sum_{x_i \in X} x_i$$

Wzór estymatora punktowego nieobciążonego dla wariancji:

$$S^2 = \frac{1}{n-1} \sum_{x_i \in X} (x_i - \bar{x})^2$$

Wzór na przedział ufności dla wartości średniej w rozkładzie o nieznanym odchyleniu standardowym:

$$\bar{x} \pm t_{\frac{\alpha}{2}} \frac{s}{\sqrt{n}}$$

Wzór na przedział ufności dla wariancji:

$$\left[S \sqrt{\frac{n-1}{\chi^2_{1-\frac{\alpha}{2}, n-1}}}, S \sqrt{\frac{n-1}{\chi^2_{\frac{\alpha}{2}, n-1}}} \right]$$

5.4.1 median_income

O ile dany rozkład nie jest rozkładem normalnym, to możemy ograniczyć się do estymacji średniej i wariancji, ale już bez przedziałów ufności, ponieważ one zakładają, że rozkład jest normalny.

```
> income = na.omit(no_outs$median_income)
> mu1 = mean(income)
> var1 = var(income)
> newr1 <- data.frame(mu1, NA, NA, var1, NA, NA)
> names(newr1) <- names(estimations)
> estimations <- rbind(estimations, newr1)
```

5.4.2 median_income log10-transformed

Tylko dla pokazania jak wyglądało by liczenie wszystkich estymatorów, przedstawiamy ten proces dla normalnego rozkładu, który uzyskaliśmy transformując jedną z cech.

```
> bprlogn = na.omit(bprlog$d)
> n4 = length(bprlogn)
> s4 = sd(bprlogn)
> SE4 = s4/sqrt(n4)
> E4 = qt(.975, df = n4 - 1) * SE4
> mu4 = mean(bprlogn)
> var4 = var(bprlogn)
> newr4 <- data.frame(mu4, mu4 - E4, mu4 + E4, var4,
+                         var4*(n4-1)/qchisq(.975, n4-1),
+                         var4*(n4-1)/qchisq(.025, n4-1))
> names(newr4) <- names(estimations)
> estimations <- rbind(estimations, newr4)
```

5.4.3 median_house_value

```
> house.value = na.omit(no_outs$median_house_value)
> mu2 = mean(house.value)
> var2 = var(house.value)
> newr2 <- data.frame(mu2, NA, NA, var2, NA, NA)
> names(newr2) <- names( estimations )
> estimations <- rbind( estimations, newr2)
```

5.4.4 bedrooms_per_rooms

```
> bpr = na.omit(no_outs$bedrooms_per_rooms)
> mu3 = mean(bpr)
> var3 = var(bpr)
> newr3 <- data.frame(mu3, NA, NA, var3, NA, NA)
> names(newr3) <- names( estimations )
> estimations <- rbind( estimations, newr3)
```

Zobaczmy teraz jak wygląda nasza tablica.

```
> # 1 - median_income
> # 2 - median_income log-transformed
> # 3 - median_house_value
> # 4 - bedrooms_per_room
> estimations
```

	mean	meanlow	meanup	var	varlow	varup
1	3.847174e+00	NA	NA	3.234598e+00	NA	NA
2	-6.856271e-01	-0.6870271	-0.6842271	1.046696e-02	0.01026732	0.01067251
3	2.073107e+05	NA	NA	1.327183e+10	NA	NA
4	2.122168e-01	NA	NA	2.817922e-03	NA	NA

5.5 Analiza zależności pomiędzy danymi

W tej podsekcji będziemy zadawać pytania, a potem na nich odpowidać stosując odpowiednie testy statystyczne.

5.5.1 Jak bliskość do oceanu wpływa na cenę?

Patrząc do tablicy przedstawionej wyżej i wziętej z artykułu, i biorąc pod uwagę, że **median_house_value** nie jest rozkładem normalnym, to musimy wykorzystać test nieparametryczny, w naszym przypadku - **Kruskal-Wallis test**. On przewiduje, że dane muszą być niezależnymi. Zgodnie z artykułem dane niezależne dla grup oznaczają, że dane rekordów wśród jednej grupy muszą być niezależne. Możemy przyjąć, że one takimi są.

Dodając, nie będziemy sprawdzać dla wszystkich grup z **ocean_proximity**, usuniemy *ISLAND*, bo jest małoliczną grupą, co może nam zaszkodzić w prowadzeniu testu.

Jednym założeniem jest jednakowa forma i skala w różnych grupach. My nie mamy drastycznych różnic w ilości rekordów każdej grupie i taką samą skalę, więc przytępujemy do testu.

H_0 - hipotezą zerową - jest to, że wszystkie mediany są równe, czyli w każdej grupie wartość domów nie różni się. Alternatywną hipotezą jest, że przynajmniej w jednej grupie jest inna tendencja. Rysunki do tego były przedstawione jeszcze na etapie zamiany kategorycznej zmiennej na numeryczną.

```
> krusktmpdata <- no_outs %>% filter(ocean_proximity != 5)
> kruskal.test(median_house_value ~ ocean_proximity, data = krusktmpdata)

Kruskal-Wallis rank sum test

data: median_house_value by ocean_proximity
Kruskal-Wallis chi-squared = 6581.7, df = 3, p-value < 2.2e-16
```

Otrzymaliśmy bardzo małą wartość p , więc odrzucamy hipotezę zerową na korzyść alternatywnej. Czyli bliskość do oceanu wpływa na ceny mieszkań.

5.5.2 Jak różni się cena na mieszkania pomiędzy *NEAR BAY* i *NEAR OCEAN*?

To jest kontynuacja poprzedniego pytania, ponieważ dostaliśmy wynik, że przynajmniej w jednej grupie jest inna tendencja dla cen, i jest to dość logiczne, im bliżej do wody, tym droższe są mieszkania. Ale mieliśmy wcześniej problem z wyznaczaniem różnicy pomiędzy *NEAR BAY* i *NEAR OCEAN*. Ustaliliśmy, że dla *NEAR BAY* ceny są większe, ale czy naprawdę była jakaś różnica w tym? Może ceny nie różnią się pomiędzy tymi dwoma kategoriami? Sprawdźmy to.

Nasze dane w grupach dalej nie mają rozkładu normalnego, więc musimy wykorzystać test nieparametryczny, w naszym przypadku to **Mann-Whitney test**.

```
> whitntmpdata <- no_outs %>% filter(ocean_proximity == 3 || ocean_proximity == 4)
> wilcox.test(whitntmpdata$median_house_value, whitntmpdata$ocean_proximity)

Wilcoxon rank sum test with continuity correction

data: whitntmpdata$median_house_value and whitntmpdata$ocean_proximity
W = 424792110, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Test daje nam wynik dla wartości p bardzo mały, co oznacza, że analogiczna hipoteza zerowa jak i w poprzednim teście zostaje odrzucona. Czyli jest różnica pomiędzy cenami w grupach *NEAR BAY* i *NEAR OCEAN*.

5.5.3 Czy wpływa *median_income* na *median_house_value*?

Wcześniej widzieliśmy bardzo duży współczynnik korelacji między tymi cechami. Sprawdźmy przy pomocy testu **Spearman'a**.

Hipotezą zerową jest H_0 - nie istnieje monotonicznej zależności pomiędzy cechami. Odpowiednio hipoteza alternatywna stwierdza, że istnieje.

```
> cor.test(no_outs$median_house_value, no_outs$median_income, method = "spearman")  
  
Spearman's rank correlation rho  
  
data: no_outs$median_house_value and no_outs$median_income  
S = 4.6573e+11, p-value < 2.2e-16  
alternative hypothesis: true rho is not equal to 0  
sample estimates:  
rho  
0.6739262
```

Wartość p jest znikoma, więc odrzucam hipotezę zerową. Stwierdzamy, że istnieje monotoniczna zależność pomiędzy danymi cechami. Dodatkowo współczynnik $\rho = 0.6739262$ pokazuje jak silnie zależy jedno od drugiego.

Test **Spearman'a** jest nieparametrycznym, musielibyśmy taki wykorzystać ze względu na nienormalny rozkład cech.

5.5.4 Jak *rooms_per_person* wpływa na *median_house_value*?

Korzystamy z tego samego testu **Spearman'a**.

```
> cor.test(no_outs$median_house_value, no_outs$rooms_per_person, method = "spearman")  
  
Spearman's rank correlation rho  
  
data: no_outs$median_house_value and no_outs$rooms_per_person  
S = 8.4815e+11, p-value < 2.2e-16  
alternative hypothesis: true rho is not equal to 0  
sample estimates:  
rho  
0.4060968
```

Wartość p jest bliska 0, więc odrzucamy H_0 . Współczynnik $\rho = 0.4$ wskazuje jak silnie są zależne, bo są one zależne monotonicznie, o tym nam mówi hipoteza alternatywna.

5.6 Regresja

Regresja potrzebuje spełnienia dużej ilości założeń, więc stworzymy je, i potem będziemy testowali czy nasze założenia zachodzą.

Mamy taką listę założeń:

- Wszystkie współczynniki i składniki bez zmiennej są stałymi, czyli inaczej mówiąc, model jest liniowy. To samo stosuje się *error term*, czyli naszych *residuals*. Zakładamy, że są z współczynnikiem równym 1. Powinna zachodzić następująca równość:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \epsilon$$

Wszystkie $\beta_1 \dots \beta_k = const$

- Średnia populacji *residuals* jest równa 0, to oznacza, że nasz model nie robi predykcję powyżej, lub poniżej rzeczywistych wartości.
- Wszystkie zmienne są nieskorelowane z *residuals*, czyli zachowujemy definicję *residuals* jako losowy błąd.
- Same wartości w *residuals* nie są z sobą skorelowane, czyli nie da się przewidzieć następną wartość mając wartości, które pojawiły się przed.
- Wariancja *residuals* jest stałą
- Żadna zmienna nie jest idealną liniową funkcją pozostałych. Także nie powinno się pojawić bardzo silnych korelacji.
- **Opcjonalny.** Rozkład *residuals* jest normalny. Niespełnienie tego warunku nie jest założeniem samego modelu OLS, ale musi zachodzić, jeżeli chcemy badać różne hipotezy i tworzyć przedziały ufności i inne.

Oj, trochę jest do sprawdzenia, może dlatego zgodnie z statystyką podaną wyżej tylko 22% prac naukowych sprawdzili wszystkie założenia w swoich testach?

Na sam początek, tworzymy nasz model i sprawdzamy go:

```
> model <- lm(median_house_value ~ median_income + ocean_proximity +
+               bedrooms_per_rooms + rooms_per_person + latitude +
+               housing_median_age, no_outs, na.action=na.omit)
> summary(model)

Call:
lm(formula = median_house_value ~ median_income + ocean_proximity +
    bedrooms_per_rooms + rooms_per_person + latitude + housing_median_age,
    data = no_outs, na.action = na.omit)

Residuals:
    Min      1Q  Median      3Q     Max 
-503913 -40937 -8003   30481  459952 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -108778.18    10138.94 -10.73  <2e-16 ***
median_income    49370.81     398.80  123.80  <2e-16 ***
```

```

ocean_proximity      24590.99      571.16    43.05 <2e-16 ***
bedrooms_per_rooms  626137.65    13324.83    46.99 <2e-16 ***
rooms_per_person     36326.22     855.67    42.45 <2e-16 ***
latitude            -4612.76    240.42   -19.19 <2e-16 ***
housing_median_age   1277.77     41.48    30.80 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 68350 on 20078 degrees of freedom
(555 observations deleted due to missingness)
Multiple R-squared: 0.6399, Adjusted R-squared: 0.6398
F-statistic: 5946 on 6 and 20078 DF, p-value: < 2.2e-16

Widzimy, że wszystkie nasze zmienne są bardzo znaczące. I chociaż R^2 nie jest aż tak duża, ale wciąż wystarczająca dla nas. Dla każdej zmiennej wartość p jest bliska零, czyli odrzucamy hipotezę zerową, że nie są powiązane i zostawiamy wszystkie zmienne. Zaczniemy sprawdzać założenia.

- Wszystkie współczynniki są wartościami stałymi, więc pierwsze założenie jest spełnione.
- Sprawdźmy średnią *residuals*:

```

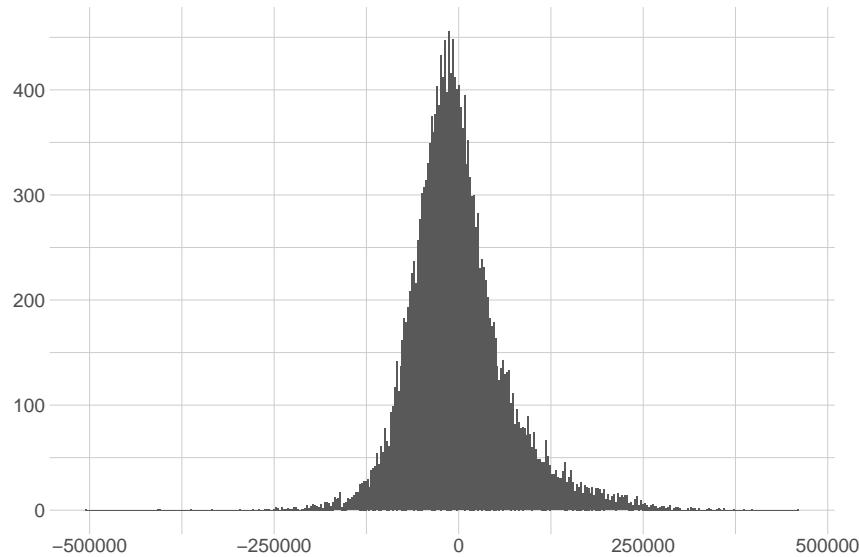
> mean(model$residuals)
[1] -2.321714e-11

```

Wartość jest bardzo bliska 0, więc zakładamy, że założenie jest spełnione, biorąc dodatkowo pod uwagę, że same wartości *residuals* są rzędu dziesiątek tysięcy i więcej.

Narysujmy sobie od razu histogram *residuals*.

Model residuals distribution



Widzimy, że rozkład jest bardzo podobny do normalnego, sprawdzimy to później jeszcze i dokładniej.

- Sprawdzamy wsp. korelacji

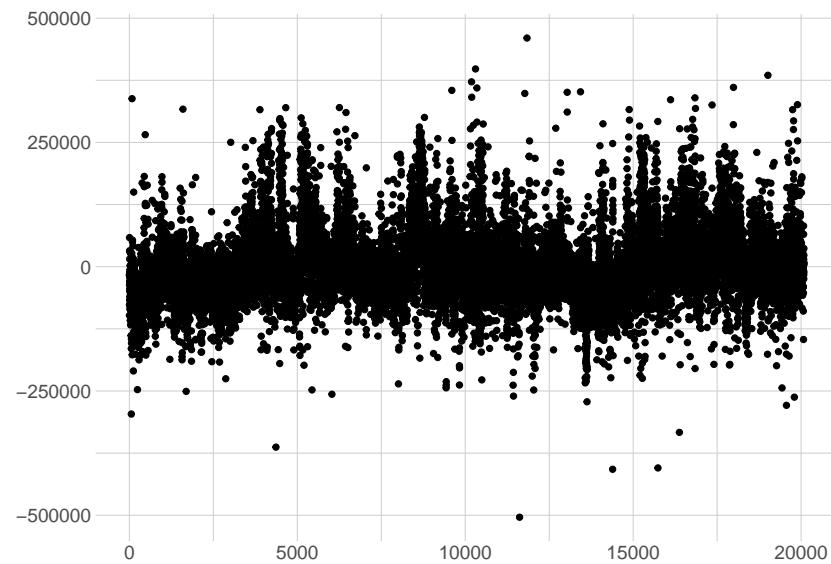
```
> cor(model$model, model$residuals)
```

```
[,1]
median_house_value 6.000820e-01
median_income      9.459248e-17
ocean_proximity    1.346156e-15
bedrooms_per_rooms 5.304133e-17
rooms_per_person   7.446344e-17
latitude          2.135699e-17
housing_median_age 2.104424e-16
```

Korelacja jest praktycznie zero, w szczególności dla zmiennych bazując się na których, robimy prewidowanie Y . Założenie spełnione.

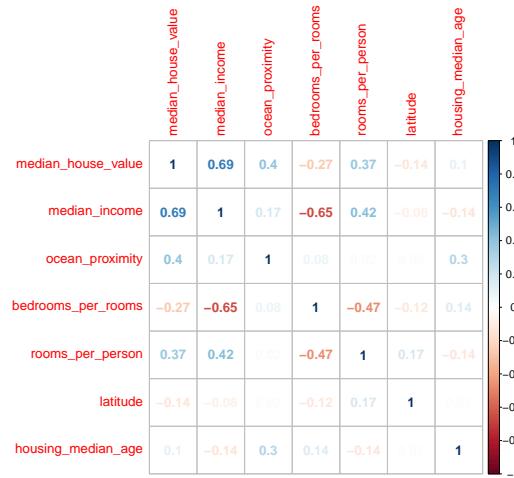
- Możemy sprawdzić niezależność wartości $residuals$ wewnątrz siebie rysując wykres zależny od indeksu.

Model Residuals



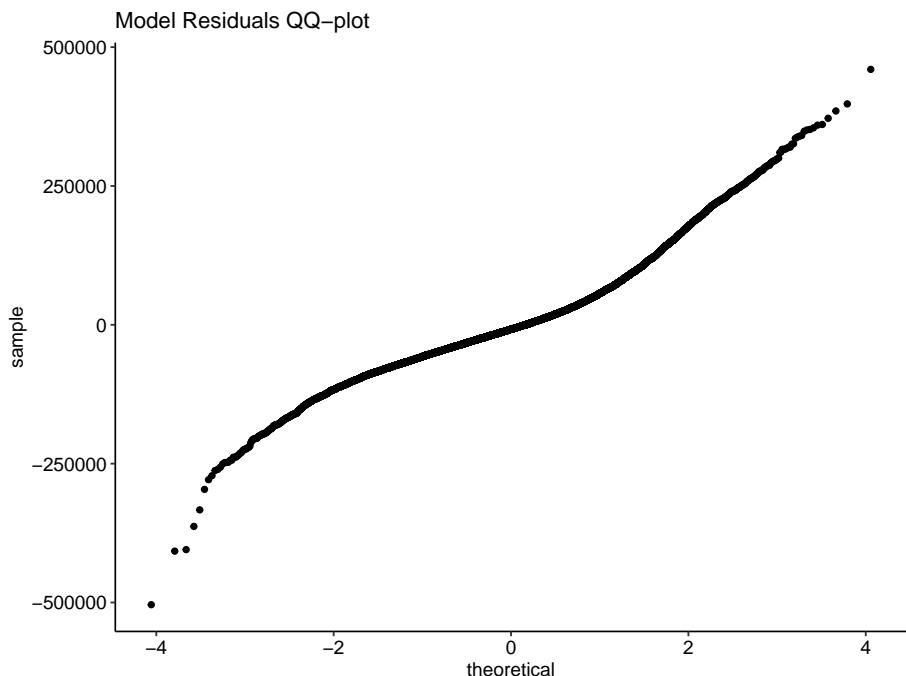
Nie widzimy żadnych zależności. Założenie spełnione.

- Wykorzystując wykres z poprzedniego punktu, możemy stwierdzić, że *residuals* ma stałą wariancję. Nie da się stwierdzić, że gdzieś odchylenie jest większe lub mniejsze. Założenie spełnione.
- Sprawdzmy korelację pomiędzy zmiennymi.



Najbardziej nas interesują zmienne X , a nie Y . Widzimy, że nigdzie nie ma idealnej korelacji lub bardzo silnej. Właśnie przetwarzając cechy w **feature engineering** od razu wspominaliśmy, że musimy spełnić to założenie. I jest ono spełnione.

- Ostatnie opcjonalne założenie potrzebne nam do podalszego przeprowadzenia testów statystycznych. Sprawdźmy po prostu stosując QQ-plot i **Shapiro-Wilk test**. Nie będziemy powtarzać całej procedury jak przy testowaniu cech.



Linia jest dość bliska do prostej, ale wciąż ma pewne odkształcenia. Zobaczmy co nam powie test. Zrobimy ich 5 ze względu na wyżej opisane przyczyny.

```
> shapiro.test(sample(model$residuals, 50))

Shapiro-Wilk normality test

data: sample(model$residuals, 50)
W = 0.93357, p-value = 0.007562

> shapiro.test(sample(model$residuals, 50))

Shapiro-Wilk normality test
```

```

data: sample(model$residuals, 50)
W = 0.93895, p-value = 0.01224

> shapiro.test(sample(model$residuals, 50))

Shapiro-Wilk normality test

data: sample(model$residuals, 50)
W = 0.97377, p-value = 0.3271

> shapiro.test(sample(model$residuals, 50))

Shapiro-Wilk normality test

data: sample(model$residuals, 50)
W = 0.87969, p-value = 0.0001095

> shapiro.test(sample(model$residuals, 50))

Shapiro-Wilk normality test

data: sample(model$residuals, 50)
W = 0.97024, p-value = 0.2367

```

Niektóre próbki tworzą rozkład normalny, niektóre nie, same wartości p nie są drastycznie małe. Wartości W też są dość wysokie. Nie możemy stwierdzić, że rozkład jest normalnym, jest dość bliskim do normalnego, ale nie normalnym. Więc przeprowadzając testy statystyczne musimy być świadomi tego, że nasze wyniki mogą trochę się różnić od rzeczywistych.

Podsumowując, stworzyliśmy model regresyjny, który odpowiada wszystkim must-have założeniom. Jest on przydatny do przewidywania mediany cen na mieszkanie mając pewne dane o miejscu i bloku domów.

6 Wnioski

Podeszliśmy do końca naszej analizy, która została dość szczegółowo przeprowadzona. Nieprzydatne do analizy dane przygotowaliśmy, transformowaliśmy w takie, które odpowiadają potrzebnym nam założeniom. A potem na ich podstawie zbadaliśmy najprostsze statystyki opisujące te dane, zbadaliśmy bardzo dokładnie kilka cech i stworzyliśmy dla nich estymatory punktowe i przedziałowe. Dostaliśmy odpowiedzi na kilka pytań nas interesujących stosując testy statystyczne, a w końcu stworzyliśmy model regresyjny, z pomocą którego możemy przewidywać ceny mieszkań mając pewne początkowe dane.