

# Pricing

2023-01-04

Realizado por : Jose Delgado Serrano (Grupo 1) GitHub: <https://github.com/Josdelser/PreciosMadrid-FID>  
Kaggle: <https://www.kaggle.com/datasets/mapecode/madrid-province-rent-data>

Paquetes y librerías

```
#install.packages("tidyverse")  
#install.packages("dplyr")  
#install.packages("rattle")  
# libraries  
library(rpart)  
library(rpart.plot)  
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.  
## Versión 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Escriba 'rattle()' para agitar, sacudir y rotar sus datos.
```

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
## filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
## intersect, setdiff, setequal, union
```

```
library(ggplot2)  
library(RColorBrewer)  
library(caret)
```

```
## Loading required package: lattice
```

```
library(ggfortify)
library(readr)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
require(corrplot)
```

```
## Loading required package: corrplot
```

```
## corrplot 0.92 loaded
```

```
set.seed(1)
```

Lectura del dataset

```
precios_madrid <- read.csv("PreciosMadrid.csv")
head(precios_madrid)
```

```
##      web_id                                     url
## 1 99439319 https://www.idealista.com/en/inmueble/99439319/
## 2 99439586 https://www.idealista.com/en/inmueble/99439586/
## 3 99439169 https://www.idealista.com/en/inmueble/99439169/
## 4 26925909 https://www.idealista.com/en/inmueble/26925909/
## 5 99440018 https://www.idealista.com/en/inmueble/99440018/
## 6 99440142 https://www.idealista.com/en/inmueble/99440142/
##                                     title      type price deposit
## 1      Flat / apartment for rent in pablo luna, 4      Flat  1400      NA
## 2      Penthouse for rent in calle de Bolivia Penthouse  1300      1
## 3      Duplex for rent in calle de la constancia Duplex   950      1
## 4 Flat / apartment for rent in Urb. el viso, El Viso Flat  2975      1
## 5      Studio flat for rent in luis cabrera Studio   650      1
## 6 Flat / apartment for rent in calle de Nieremberg Flat  1200      NA
## private_owner professional_name floor_built floor_area floor year_built
## 1      False Silcasas Ochocientas      60      NA 3rd      1954
## 2      False      Cruzity      77      NA 6th      1961
## 3      False      Mm Home      72      68 3rd      1999
## 4      False      B&H Partners      160      NA 3rd      NA
## 5      False Madrid en Propiedad      30      NA 4th      NA
## 6      False Extra Inmobiliaria      54      47 4th      2009
## orientation bedrooms bathrooms second_hand lift garage_included furnished
## 1      2      1      True True      False      True
## 2      2      2      True True      False      True
## 3      east      1      1      True True      False      True
## 4      west      4      3      True True      True      True
## 5      0      1      True True      False      True
## 6      west      1      1      True True      False      True
## equipped_kitchen fitted_wardrobes air_conditioning terrace balcony storeroom
## 1      True      True      True      True      False      False
## 2      True      False      False      False      False      False
## 3      True      True      True      True      False      False
```

```
## 4      True      True      True      True      False      False
## 5      True      True      False     False      True      False
## 6      True      True      True      False     False      True
##  swimming_pool garden_area
## 1      False     False
## 2      False     False
## 3      False     False
## 4      True      False
## 5      False     False
## 6      True      False
##
## 1      pablo luna, 4, Subdistrict Castilla, District Chamartín, Madrid, Madrid city,
## 2      Calle de Bolivia, Subdistrict Bernabéu-Hispanoamérica, District Chamartín, Madrid, Madrid city,
## 3      Calle de la constancia, Urb. no, Subdistrict Prosperidad, District Chamartín, Madrid, Madrid city,
## 4      Urb. el viso, Subdistrict El Viso, District Chamartín, Madrid, Madrid city,
## 5      luis cabrera, Subdistrict Prosperidad, District Chamartín, Madrid, Madrid city,
## 6      Calle de Nieremberg, Subdistrict Ciudad Jardín, District Chamartín, Madrid, Madrid city,
##  district      subdistrict postalcode last_update
## 1 Chamartín      Castilla      28046  7 November
## 2 Chamartín Bernabéu-Hispanoamérica 28016  7 November
## 3 Chamartín      Prosperidad  28002  7 November
## 4 Chamartín      El Viso      NA      7 November
## 5 Chamartín      Prosperidad  28002  7 November
## 6 Chamartín      Ciudad Jardín 28002  7 November
```

```
colnames(precios_madrid)
```

```
## [1] "web_id"      "url"         "title"
## [4] "type"        "price"       "deposit"
## [7] "private_owner" "professional_name" "floor_built"
## [10] "floor_area"   "floor"       "year_built"
## [13] "orientation"  "bedrooms"    "bathrooms"
## [16] "second_hand"  "lift"        "garage_included"
## [19] "furnished"    "equipped_kitchen" "fitted_wardrobes"
## [22] "air_conditioning" "terrace"     "balcony"
## [25] "storeroom"    "swimming_pool" "garden_area"
## [28] "location"     "district"    "subdistrict"
## [31] "postalcode"   "last_update"
```

Despues de analizar el dataset elijo las columnas que parecen mas interesantes

```
predata1 <- select(precios_madrid,price,floor_built,bathrooms,terrace,bedrooms,postalcode,garage_included)
head(predata1)
```

```
##  price floor_built bathrooms terrace bedrooms postalcode garage_included
## 1  1400         60         1   True         2     28046      False
## 2  1300         77         2  False         2     28016      False
## 3   950         72         1   True         1     28002      False
## 4 2975        160         3   True         4         NA       True
## 5   650         30         1  False         0     28002      False
## 6 1200         54         1  False         1     28002      False
```

```
colnames(predata1)
```

```
## [1] "price"          "floor_built"    "bathrooms"      "terrace"  
## [5] "bedrooms"       "postalcode"     "garage_included"
```

Elimino pisos que cuesten 0, tengan 0 habitaciones, estén repetidos o sean NA. También para acotar el dataset vamos a coger solo los anuncios de 10 postalcode

```
predata2 = subset(predata1, price>0 & bedrooms>1 & postalcode>=28001 & postalcode<=28011)  
predata2 <- na.omit(predata2)  
predata <- unique(predata2)
```

Convertir las columnas de valores char("True", "False") en num(1,0). También estandarizo todo en numeric

```
predata$terrace <- as.numeric(as.logical(predata$terrace))  
predata$garage_included <- as.numeric(as.logical(predata$garage_included))  
predata$postalcode <- as.numeric(as.integer(predata$postalcode))  
predata$price <- as.numeric(as.integer(predata$price))  
predata$floor_built <- as.numeric(as.integer(predata$floor_built))  
predata$bathrooms <- as.numeric(as.integer(predata$bathrooms))  
predata$bedrooms <- as.numeric(as.integer(predata$bedrooms))
```

Finalmente después del preprocesamiento de datos, obtenemos el dataset final

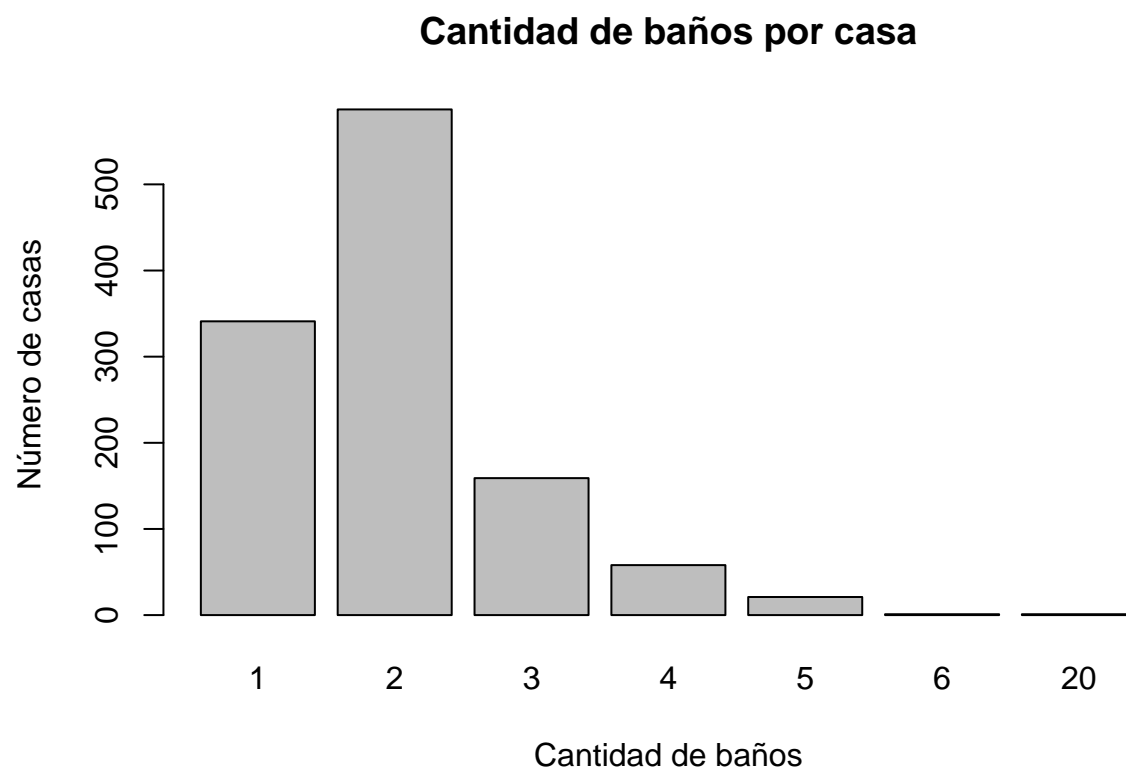
```
data<-predata
```

Una vez con los datos bien definidos, pasamos a la visualización.

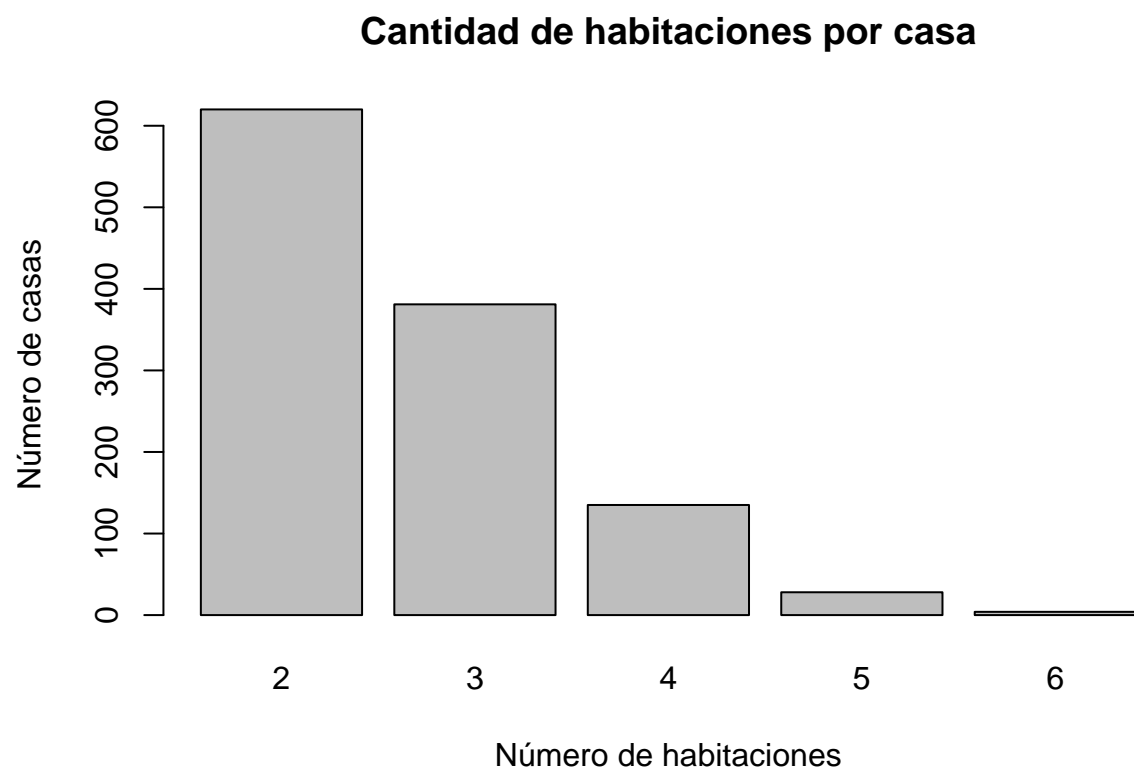
```
barplot(table(data$terrace),  
main="Número de casas con terraza",  
xlab="Tiene o no terraza",  
ylab="Número de casas",)
```



```
barplot(table(data$bathrooms),  
main="Cantidad de baños por casa",  
xlab="Cantidad de baños",  
ylab="Número de casas",)
```

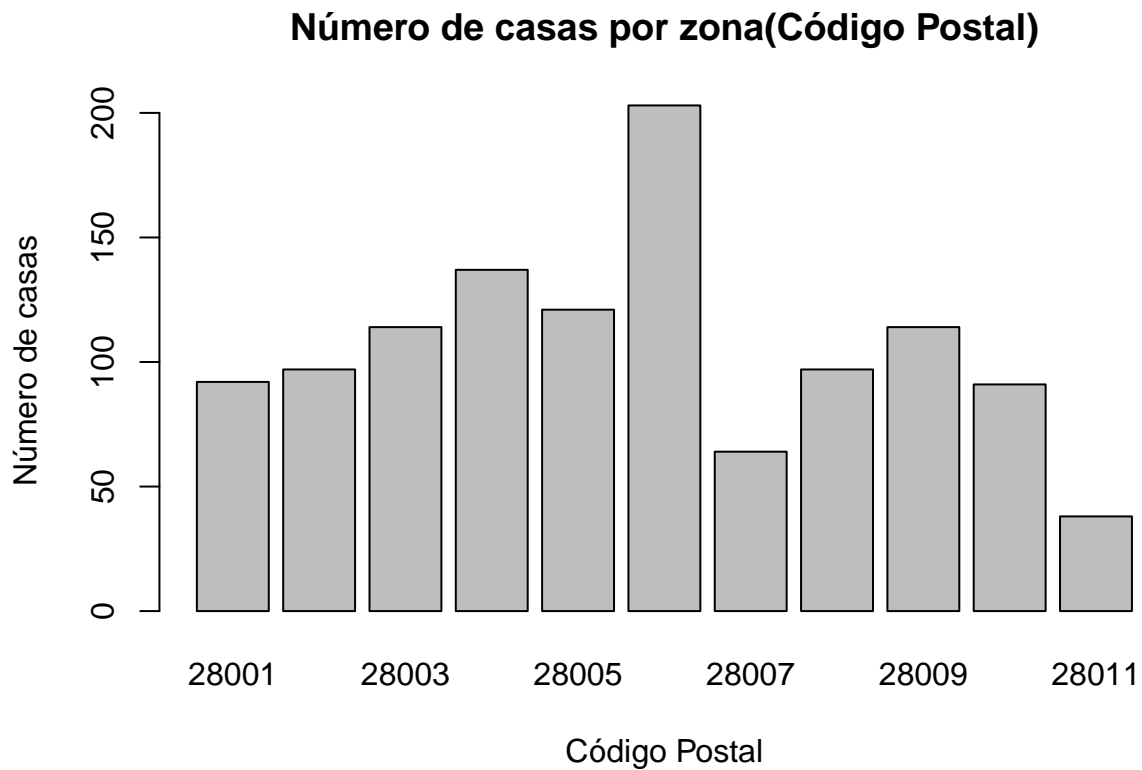


```
barplot(table(data$bedrooms),  
main="Cantidad de habitaciones por casa",  
xlab="Número de habitaciones",  
ylab="Número de casas",)
```



Primero vamos a ver el número de casas por código postal.

```
barplot(table(data$postalcode),  
main="Número de casas por zona(Código Postal)",  
xlab="Código Postal",  
ylab="Número de casas",)
```



Vamos a etiquetar y categorizar según los metros construidos, con el objetivo de ver los tipos de viviendas

```
data_metros <- data
```

```
head(data_metros[order(data_metros$floor_built),])
```

```
##      price floor_built bathrooms terrace bedrooms postcode garage_included
## 7018  1750         30         1      0         2     28009           0
## 2497   800         35         1      0         2     28005           0
## 1376  1990         40         1      0         2     28009           0
## 1537   850         40         1      0         2     28009           0
## 1550   750         40         1      0         2     28009           0
## 1627   950         40         1      0         2     28010           0
```

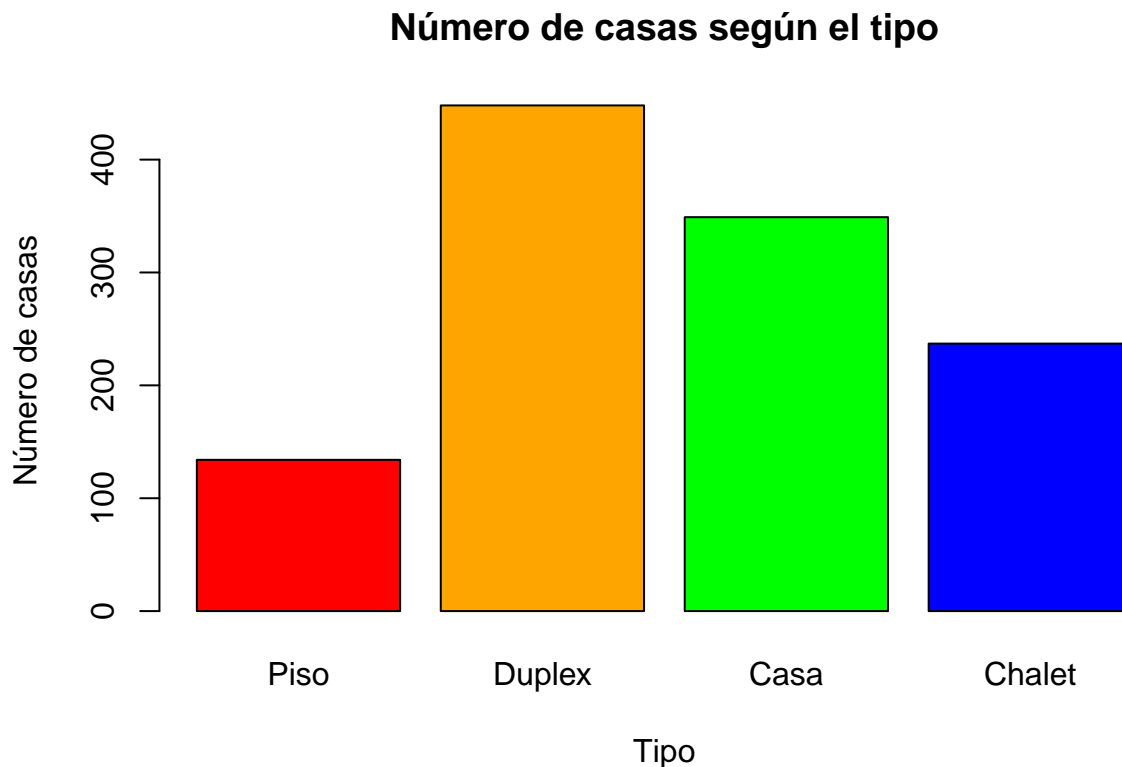
```
rangos <- c(0,65,100,150,Inf)
```

```
values <- c ('Piso','Duplex','Casa','Chalet')
```

```
data_metros$tipo <- cut(data_metros$floor_built, breaks = rangos, labels = values)
```

```
barplot(table(data_metros$tipo),
main="Número de casas según el tipo",
xlab="Tipo",
ylab="Número de casas",
col=c("red","orange","green","blue"),)
```





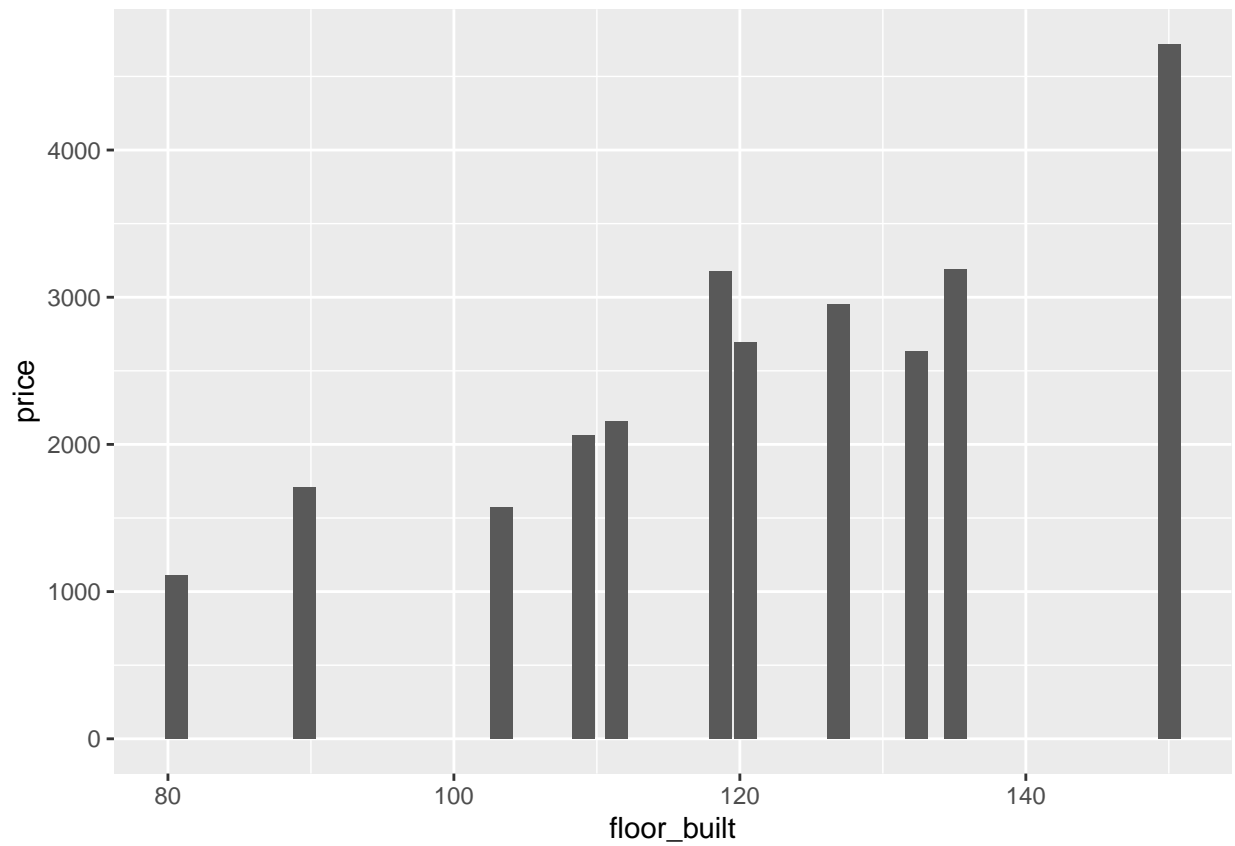
Vamos a calcular el precio medio, metros construidos y los tipos de casas dependiendo de la zona (Código postal). Para el tipo de casa pasamos a numeric con la asignación por defecto el 1 a pisos, 2 a duplex... De esta manera obtendremos una media del tipo de casa por zona

```
data_metros$tipo <- as.numeric(as.factor(data_metros$tipo))
media1 <- aggregate(data_metros[, c(1,2,8)], list(data_metros$postalcode), mean)
media1
```

```
##      Group.1    price floor_built    tipo
## 1    28001 4720.315  150.02174 3.000000
## 2    28002 2155.979  111.34021 2.402062
## 3    28003 2632.711  132.33333 2.859649
## 4    28004 3177.555  118.61314 2.671533
## 5    28005 1710.264   89.57025 2.066116
## 6    28006 3186.567  135.08867 2.807882
## 7    28007 1570.141  103.35938 2.468750
## 8    28008 2063.856  109.04124 2.402062
## 9    28009 2694.211  120.39474 2.657895
## 10   28010 2953.846  126.87912 2.637363
## 11   28011 1108.816   80.57895 1.842105
```

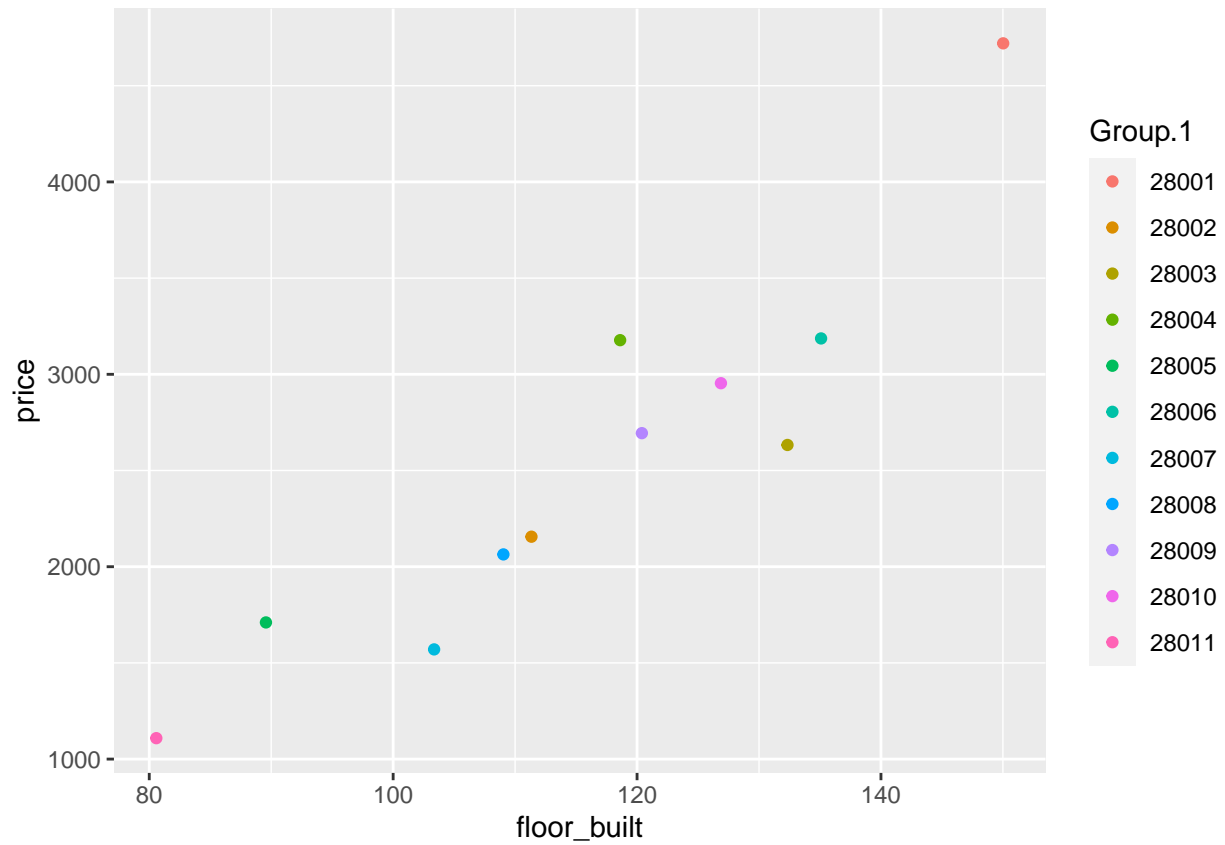
Vamos a graficar la media de precio según los metros construidos de media en barras, para ver si vemos algo interesante

```
ggplot(media1, aes(x = floor_built, y = price)) +  
  geom_col()
```



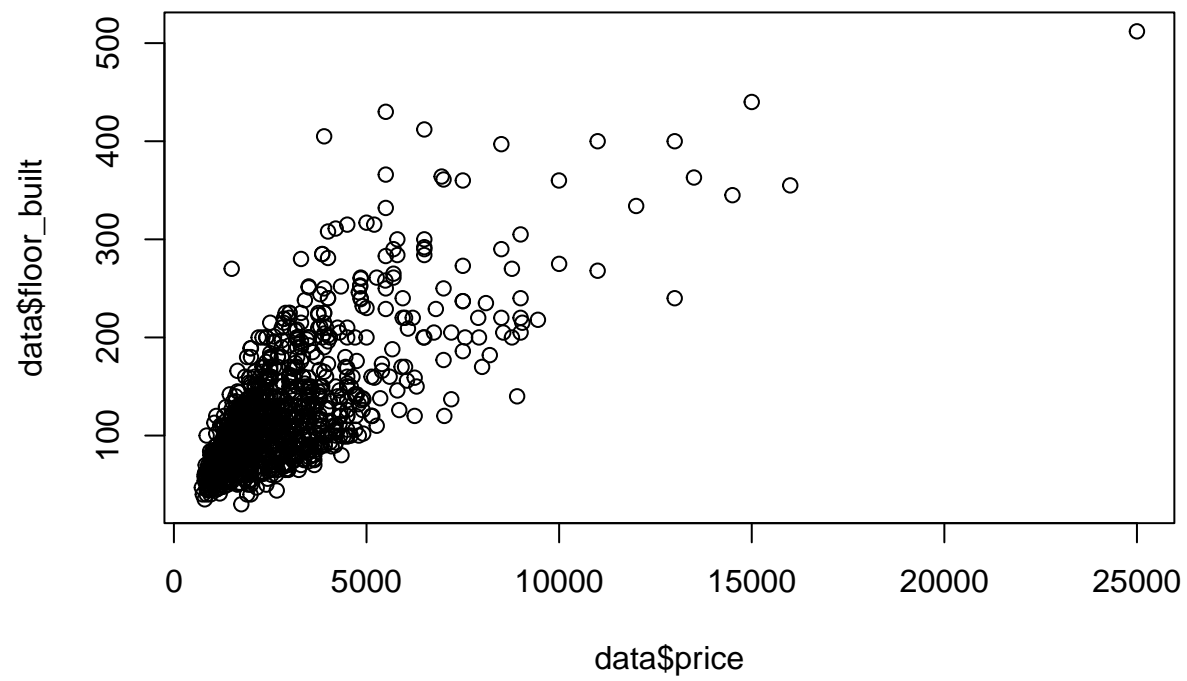
Nos percatamos de que en algunos casos no se cumple que a mayor metros mayor precio. Vamos a graficar la media de precio según los metros construidos de media por zonas en puntos

```
#Pasamos a factor el codigo postal para ver mejor el codigo de colores  
media1$Group.1 <- as.factor(as.numeric(media1$Group.1))  
ggplot(media1, aes(x= floor_built, y=price, colour=Group.1)) + geom_point()
```

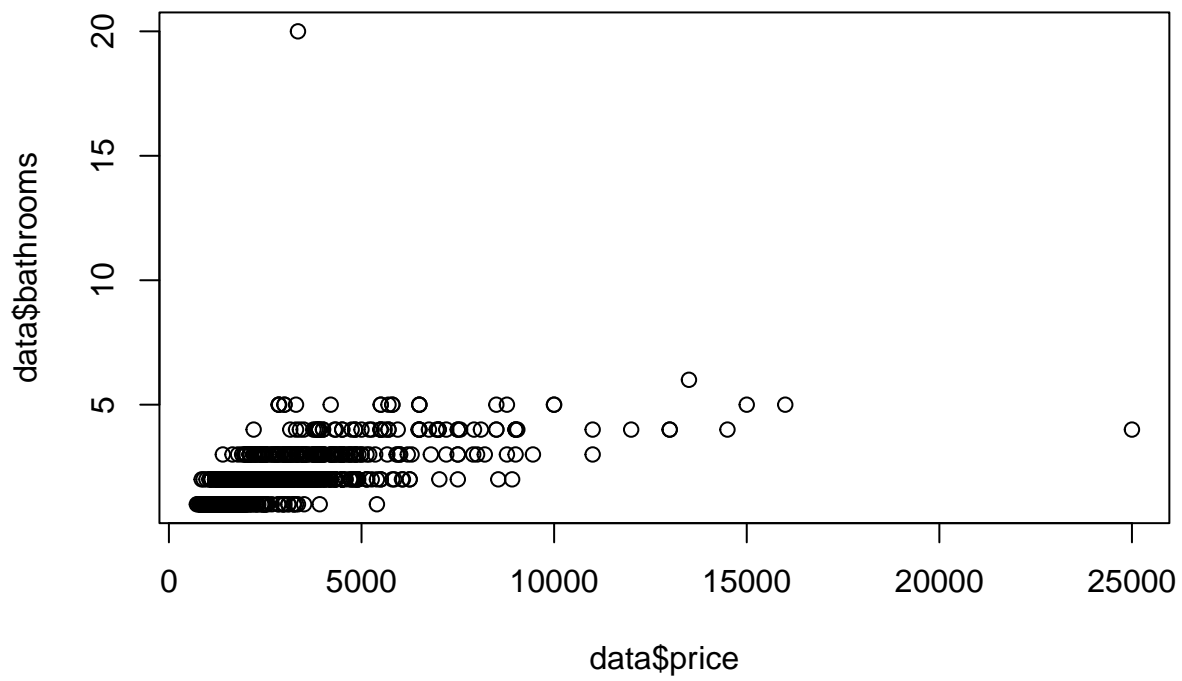


También vamos a hacer algunos analisis para ver como influyen otras variables, como el garaje, terraza...  
Gráficas de dispersión, para observar la relación entre las variables.

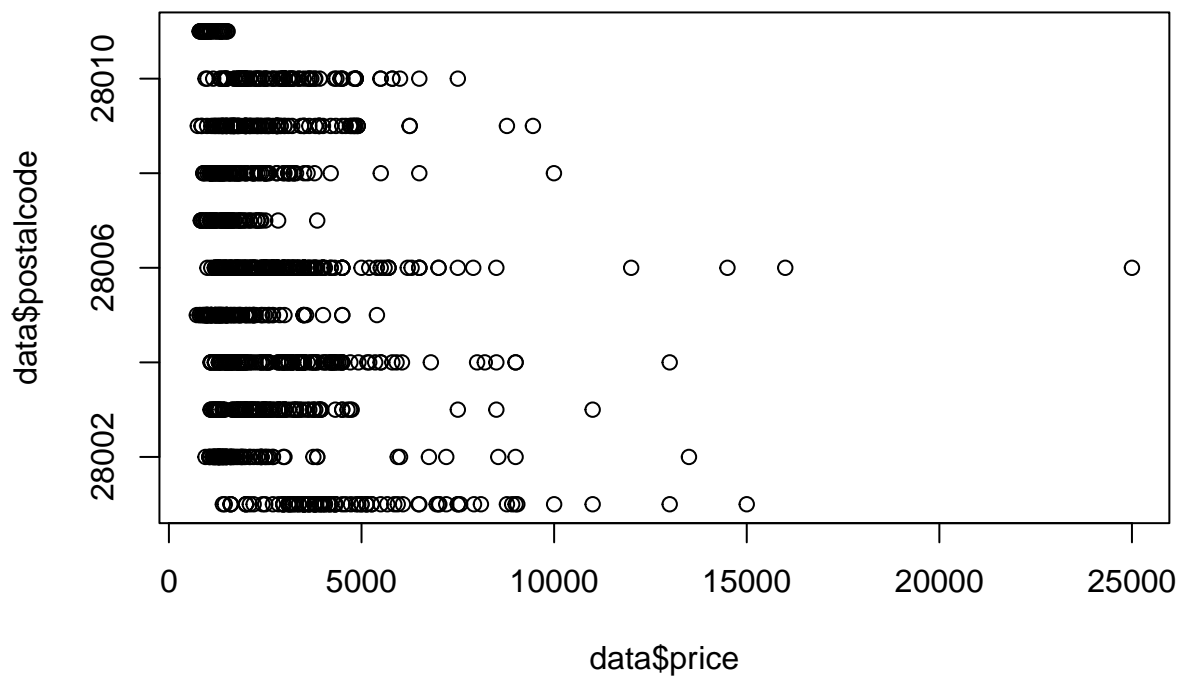
```
plot(x = data$price, y = data$floor_built)
```



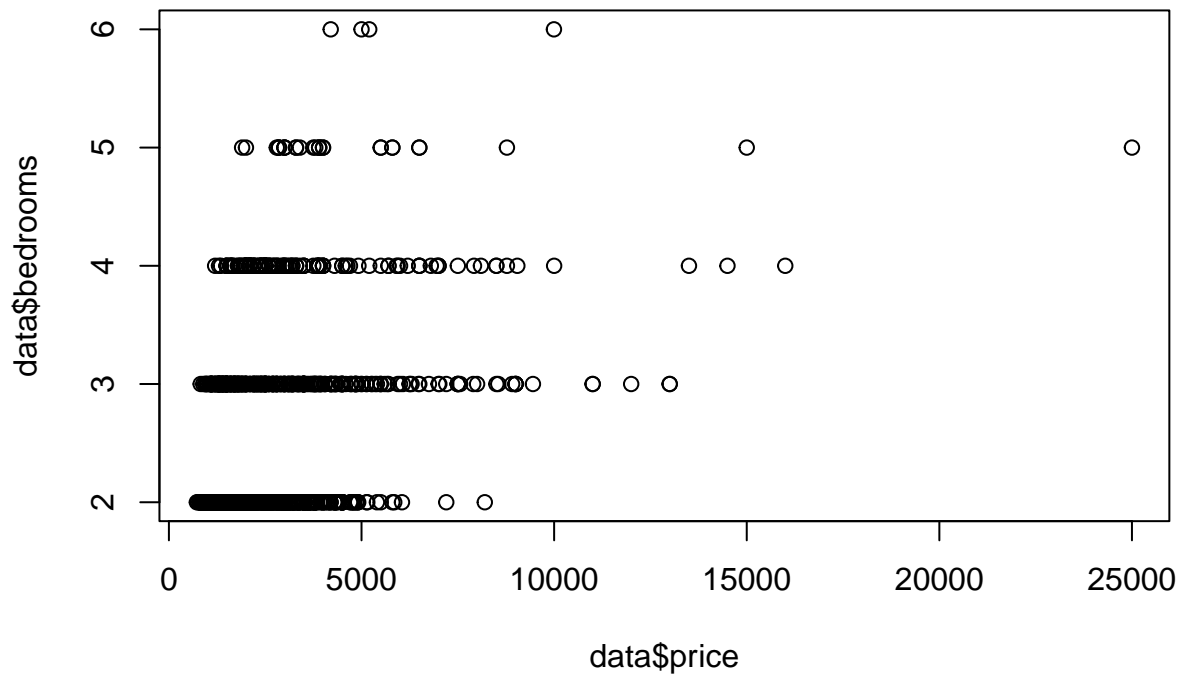
```
plot(x = data$price, y = data$floor_built)
```



```
plot(x = data$price, y = data$postalcode)
```



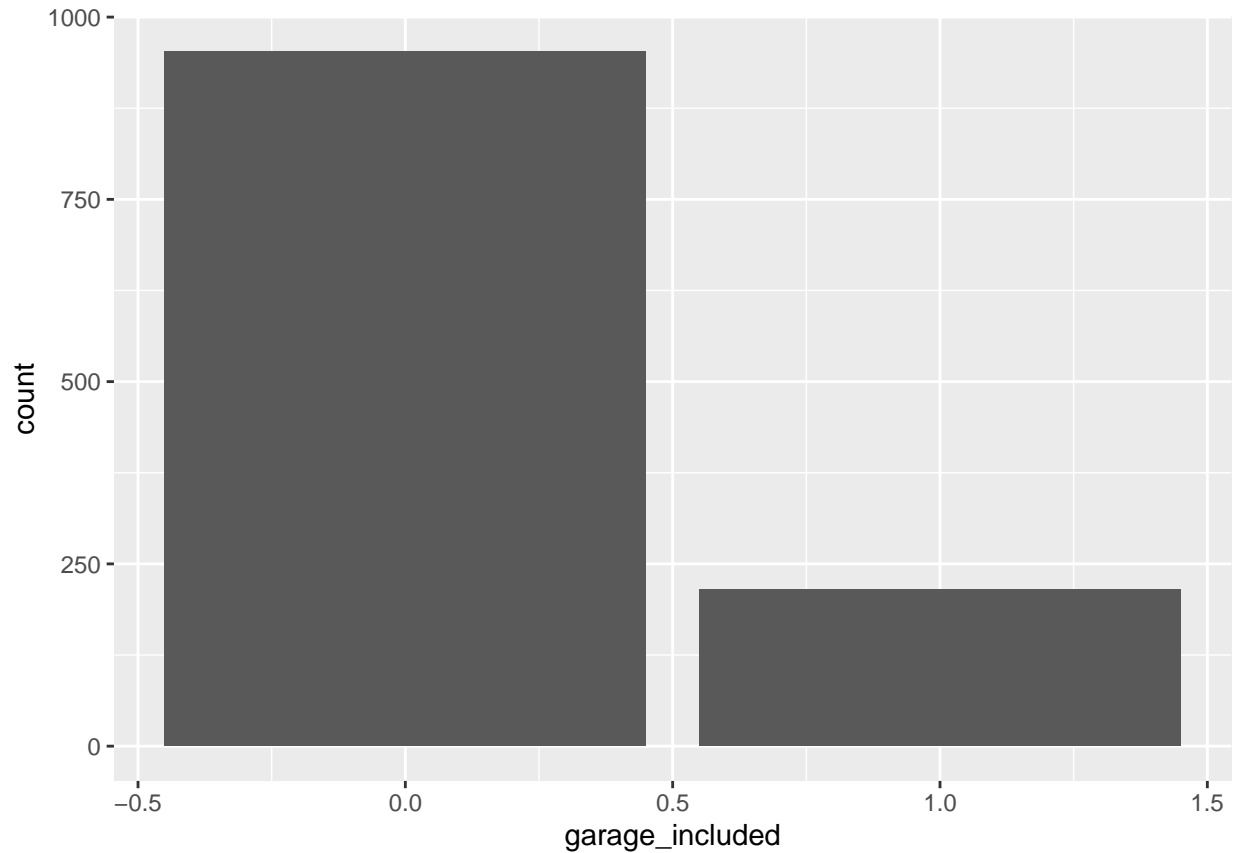
```
plot(x = data$price, y = data$bedrooms)
```



Gráfica para saber dependiendo del tipo de casa si lleva garage o no. Podríamos hacer esto con las diferentes variables, solo habría que cambiar el valor del aes.

```
a<- ggplot(data_metros, aes(garage_included))
a + geom_bar(aes(fill = tipo))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a 'group' aesthetic or to convert a numerical
##   variable into a factor?
```



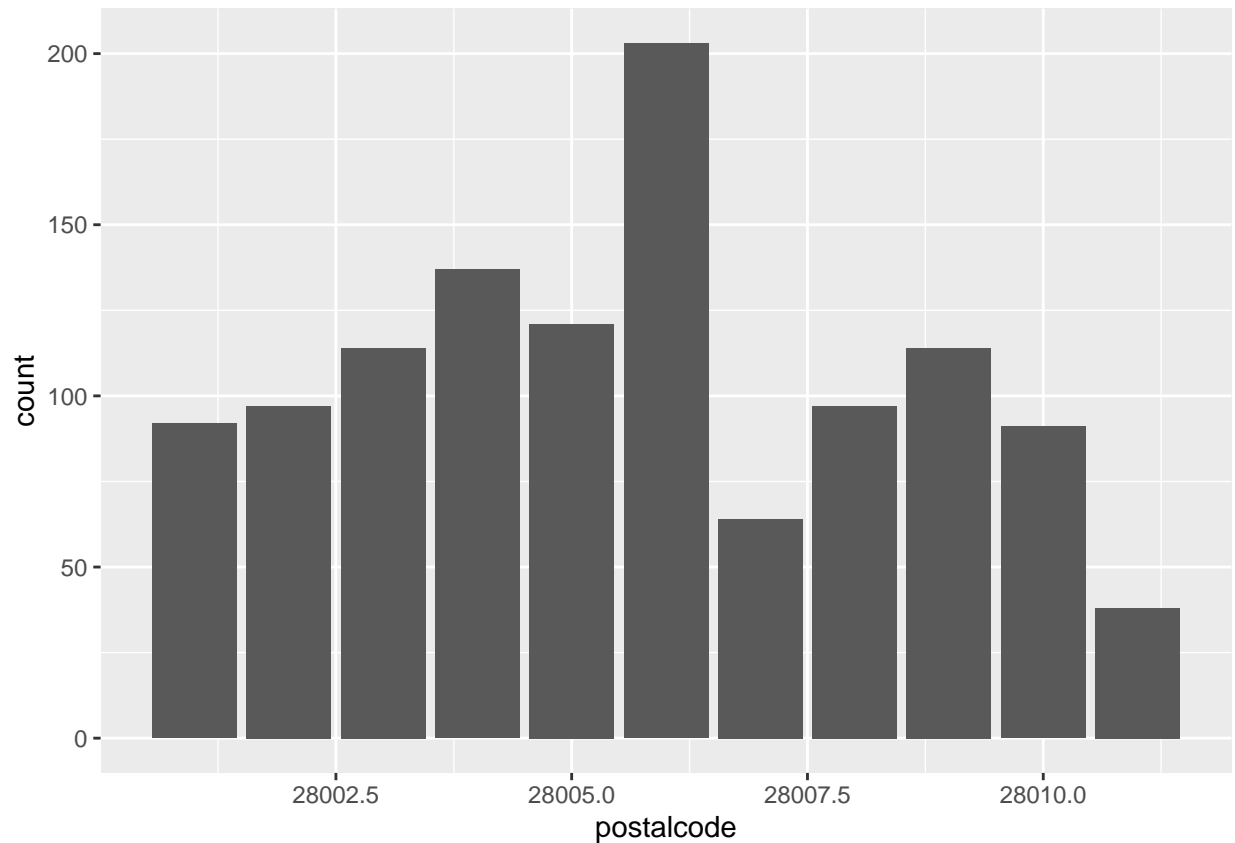
Gráfica según el número de tipos de casas por zona

```
a<- ggplot(data_metros, aes(postalcode))
```

```
a + geom_bar(aes(fill = tipo))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a 'group' aesthetic or to convert a numerical
##   variable into a factor?
```





Análisis supervisado:

Después de haber estado jugando con las variables, vamos a aplicar diferentes técnicas. Primero categorizamos los precios del dataset

```
data_analisis<-data
nrow(data_analisis)
```

```
## [1] 1168
```

```
rangos <- c(0,1300,2500,Inf)
values <- c ('barata','normal','cara')
```

```
data_analisis$categoria <- cut(data_analisis$price, breaks = rangos, labels = values)
```

```
data_analisis <- data_analisis %>%
select(-c(price))
```

```
data_agrupada <- data_analisis %>%
group_by( floor_built, bathrooms, bedrooms, garage_included, terrace,postalcode,categoria) %>%
count() %>%
arrange(desc(n()))
```

```
data_agrupada <- data_agrupada %>%
  rename(TotalCasas = n)
head(data_agrupada)
```

```
## # A tibble: 6 x 8
## # Groups:   floor_built, bathrooms, bedrooms, garage_included, terrace,
## #   postalcode, categoria [6]
##   floor_built bathrooms bedrooms garage_included terrace posta~1 categ~2 Total~3
##         <dbl>      <dbl>    <dbl>         <dbl>    <dbl>    <dbl> <fct>      <int>
## 1         30         1        2             0        0    28009 normal        1
## 2         35         1        2             0        0    28005 barata        1
## 3         40         1        2             0        0    28005 normal        1
## 4         40         1        2             0        0    28009 barata        2
## 5         40         1        2             0        0    28009 normal        1
## 6         40         1        2             0        0    28010 barata        1
## # ... with abbreviated variable names 1: postalcode, 2: categoria,
## #   3: TotalCasas
```

Primero partimos el dataset para tener datos de entrenamiento y datos de validacion.

```
train <-createDataPartition(data_agrupada$categoria, p = 0.7, list=FALSE)
validos <-createDataPartition(data_agrupada$categoria, p = 0.7, list=FALSE)
data_train <- data_agrupada[train,]
data_val <- data_agrupada[validos,]
nrow(data_train)
```

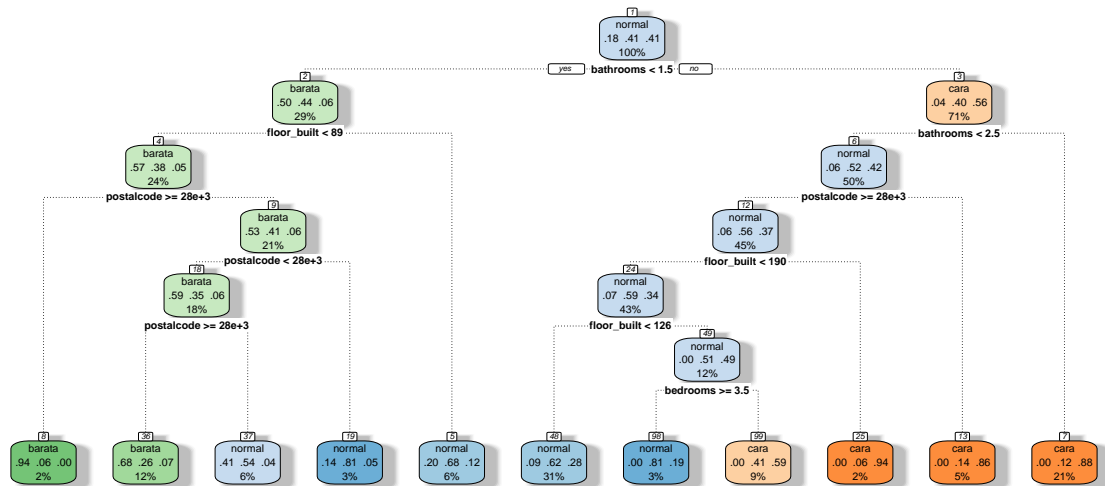
```
## [1] 737
```

```
nrow(data_val)
```

```
## [1] 737
```

Realizando el arbol podemos ver las diferentes variables que afectan a su precio y en que nos podemos basar para aproximar el precio medio de una vivienda.

```
arbol <- rpart(formula = categoria ~ ., data = data_train, method = 'class')
fancyRpartPlot(arbol)
```

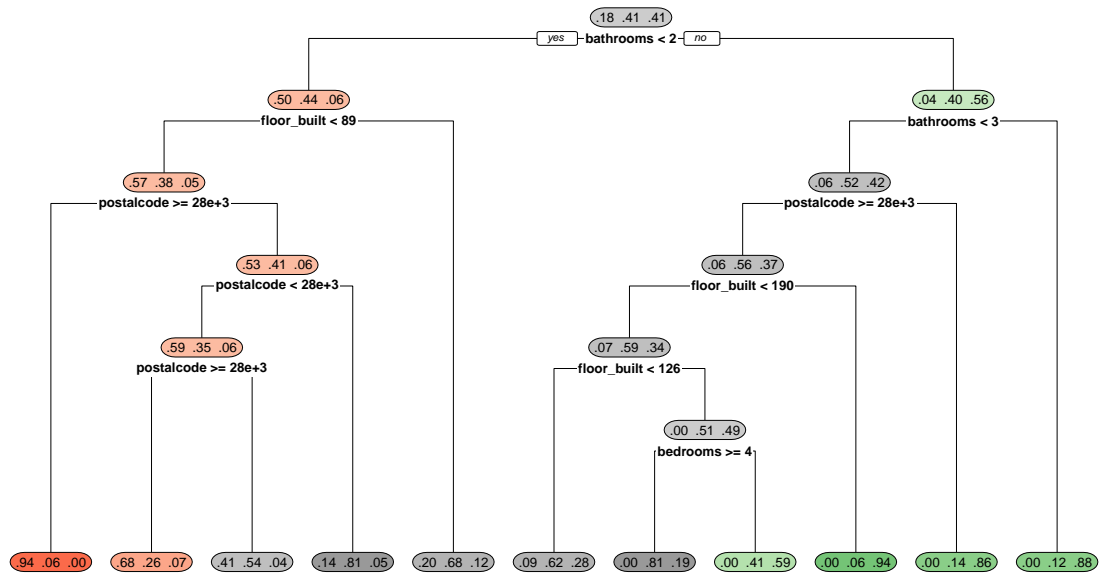


Rattle 2023-ene.-09 10:56:59 josed

Hacemos un par de cambios en la visualizacion para verlo mejor.

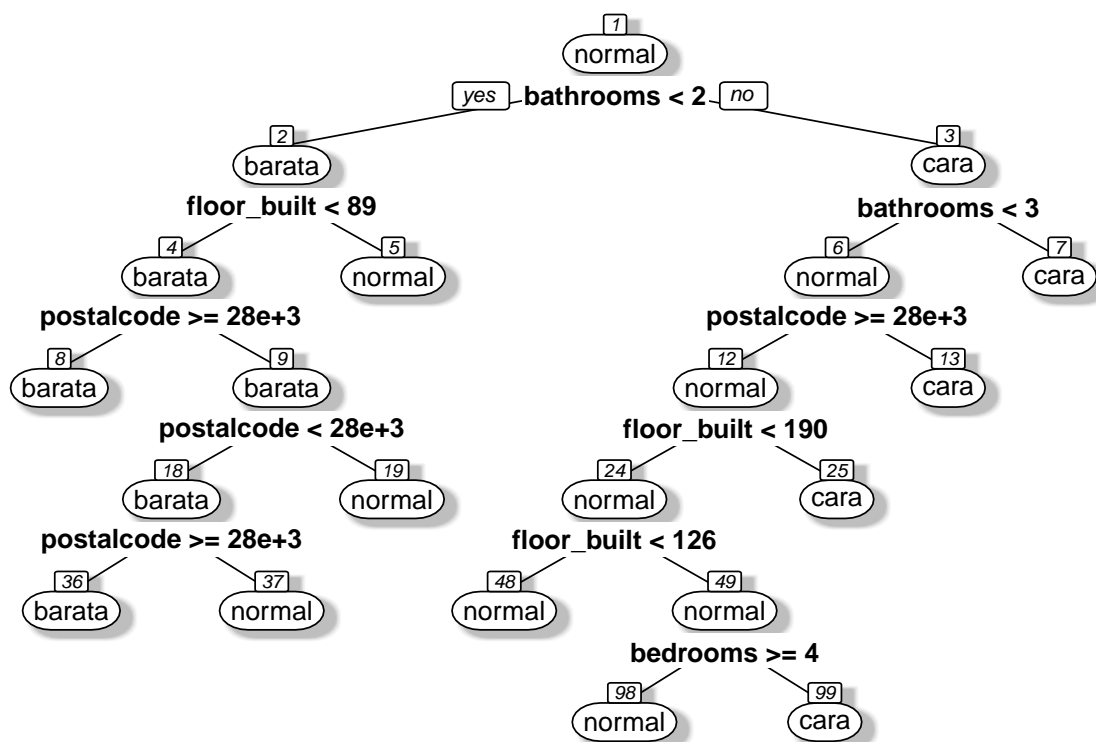
```
rpart.plot(arbol, extra = 5)
```

■ barata  
■ normal  
■ cara



```

prp(arbol, type = 2, nn = TRUE,
    fallen.leaves = FALSE,
    varlen = 0, shadow.col = "gray")
  
```



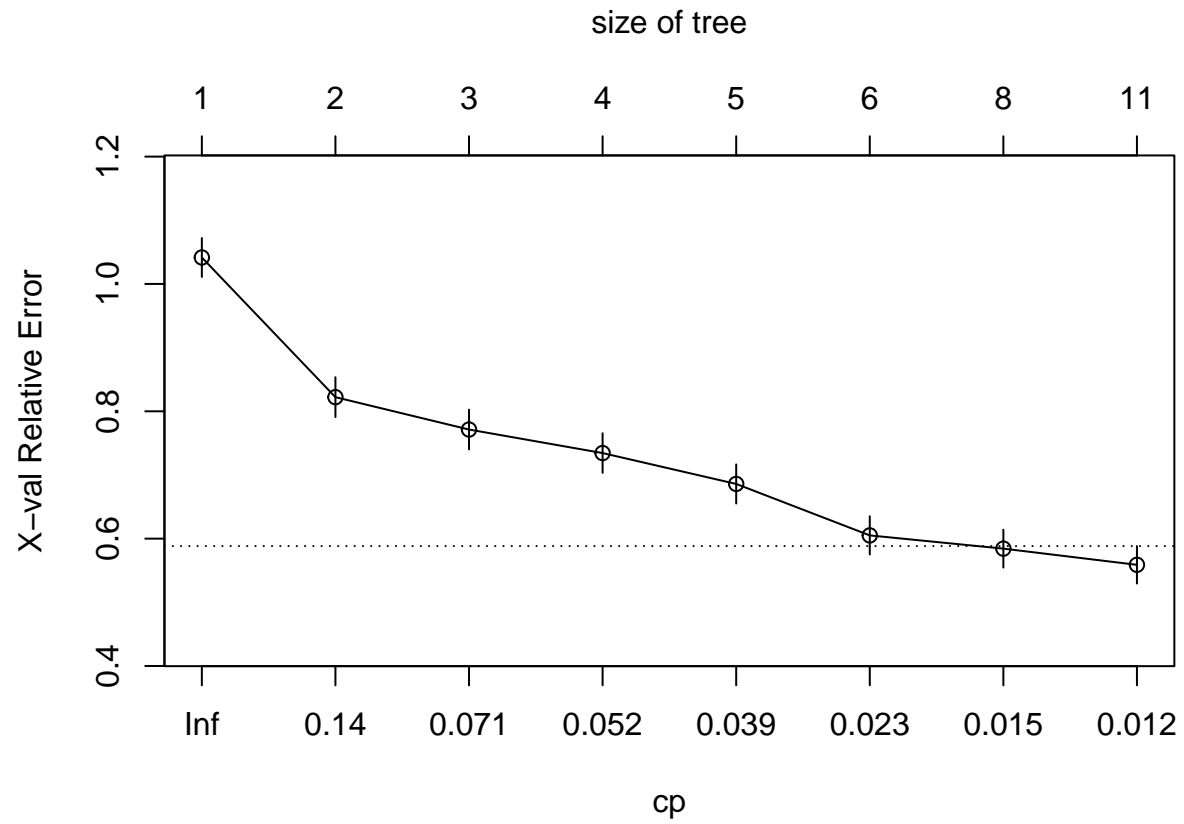
Observamos que los metros cuadrados y los cuartos de baños son las variables que mas afectan.

Vamos a ver el cp

```
arbol$cptable
```

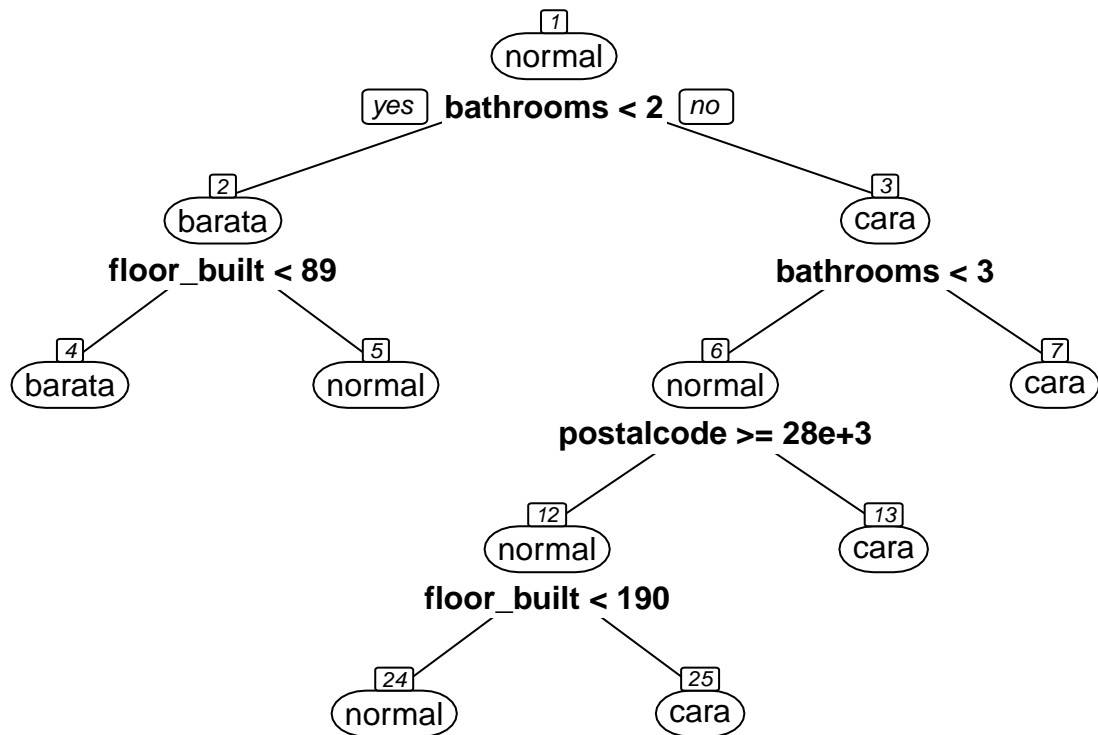
```
##          CP nsplit rel error   xerror   xstd
## 1 0.21709007    0 1.0000000 1.0415704 0.03055270
## 2 0.08775982    1 0.7829099 0.8221709 0.03133039
## 3 0.05773672    2 0.6951501 0.7713626 0.03121074
## 4 0.04618938    3 0.6374134 0.7344111 0.03105267
## 5 0.03233256    4 0.5912240 0.6859122 0.03075268
## 6 0.01616628    5 0.5588915 0.6050808 0.03001068
## 7 0.01385681    7 0.5265589 0.5842956 0.02976880
## 8 0.01000000   10 0.4849885 0.5588915 0.02944345
```

```
plotcp(arbol)
```



Podamos el árbol para reducirlo, usando el cp obtenido anteriormente

```
arbol_podado <- prune(arbol, cp = 0.017)
prp(arbol_podado, type = 2, nn = TRUE,
    fallen.leaves = FALSE,
    varlen = 0)
```



Predecimos en el data de validacion y sacamos la matrix de confusion

```

categoria_pred <- predict(arbol, newdata = data_val, type="class")

confusion_m <- table(data_val$categoria, categoria_pred)
confusion_m

```

```

##      categoria_pred
##      barata normal cara
## barata      73    56   0
## normal      21   231  52
## cara         4    93 207

```

Calculamos el acierto

```

acierto <- sum(diag(confusion_m)) / sum(confusion_m)
acierto

```

```
## [1] 0.6933514
```

Vemos que solo tenemos un 67% de acierto, por lo que vamos a controlar el algoritmo de particion para ver si mejoramos

```
controlador <- rpart.control(minsplit = 10, minbucket = round(10 / 3), maxdepth = 4, cp = 0,015)
```

Luego volvemos a repetir todo el proceso

```
arbol2 <- rpart(categoria ~., data = data_train, method = 'class', control = controlador)

categoria_pred1 <- predict(arbol2, newdata = data_val, type="class")
confusion_m1 <- table(data_val$categoria, categoria_pred1)
acierto2 <- sum(diag(confusion_m1)) / sum(confusion_m1)
acierto2
```

```
## [1] 0.6621438
```

Vemos que no mejoramos, por lo cual con la primera iteracion hemos sacado un % optimo.

Ahora pasamos a aplicar un metodo de regresion multiple para calcular el precio en funcion de las demas variables. Para esto si vamos a necesitar el precio y no la categoria

```
train_rg <- createDataPartition(data$price, p = 0.7, list=FALSE)
data_train_rg <- data[train_rg,]
data_val_rg <- data[-train_rg,]
nrow(data_train_rg)
```

```
## [1] 820
```

```
nrow(data_val_rg)
```

```
## [1] 348
```

```
regresion_mul <- lm(formula = price ~ ., data = data_train_rg)

summary(regresion_mul)
```

```
##
## Call:
## lm(formula = price ~ ., data = data_train_rg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4839.6  -644.1  -176.8   485.5 12975.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.298e+06  4.231e+05   3.067  0.00223 **
## floor_built   2.647e+01  1.074e+00  24.646 < 2e-16 ***
## bathrooms    1.659e+02  5.135e+01   3.230  0.00129 **
## terrace     -2.560e+02  9.684e+01  -2.643  0.00837 **
## bedrooms    -5.159e+02  7.081e+01  -7.286 7.54e-13 ***
## postalcode   -4.632e+01  1.511e+01  -3.066  0.00224 **
## garage_included -2.742e+02  1.169e+02  -2.346  0.01921 *
## ---
```



```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1204 on 813 degrees of freedom
## Multiple R-squared:  0.6096, Adjusted R-squared:  0.6067
## F-statistic: 211.6 on 6 and 813 DF,  p-value: < 2.2e-16
```

Segun el  $R^2$  el modelo puede explicar en un 61% la variabilidad del precio, ya que  $R^2=0,6132$  Tambien vemos que alguna variable predictor está relacionada con el precio ya que el p-value es bastante infimo.. No encontramos ninguna variable con un p-value alto por lo que nos indican que todas contribuyen en parte al modelo.

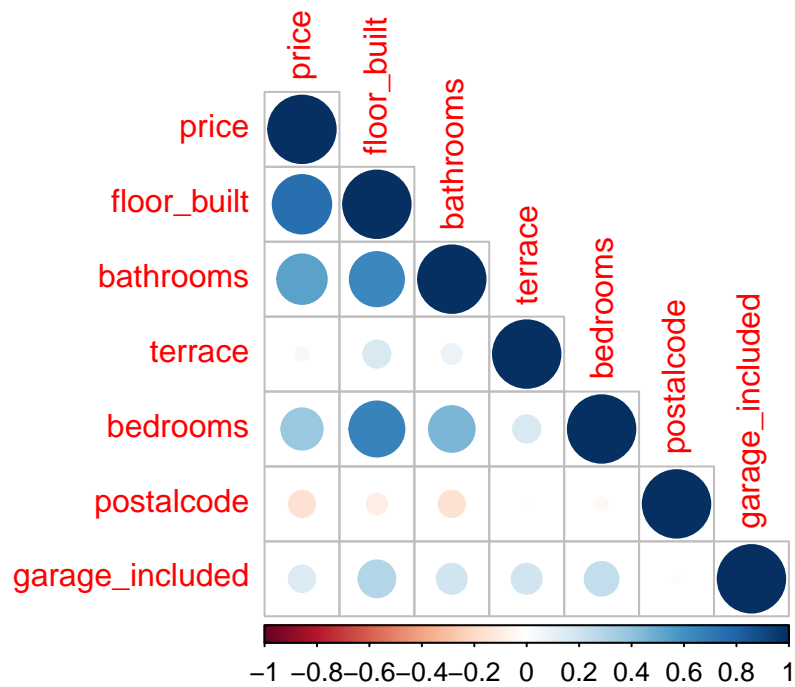
El metodo step, nos arroja que los metros construidos y los cuarto de baños son las variables que mas correlacion tienen. Los

```
step(regresion_mul, direction = "both", trace = 0)
```

```
##
## Call:
## lm(formula = price ~ floor_built + bathrooms + terrace + bedrooms +
##     postcode + garage_included, data = data_train_rg)
##
## Coefficients:
##      (Intercept)      floor_built      bathrooms      terrace
##      1297834.55           26.47          165.88        -255.98
##      bedrooms      postcode  garage_included
##      -515.94         -46.32         -274.16
```

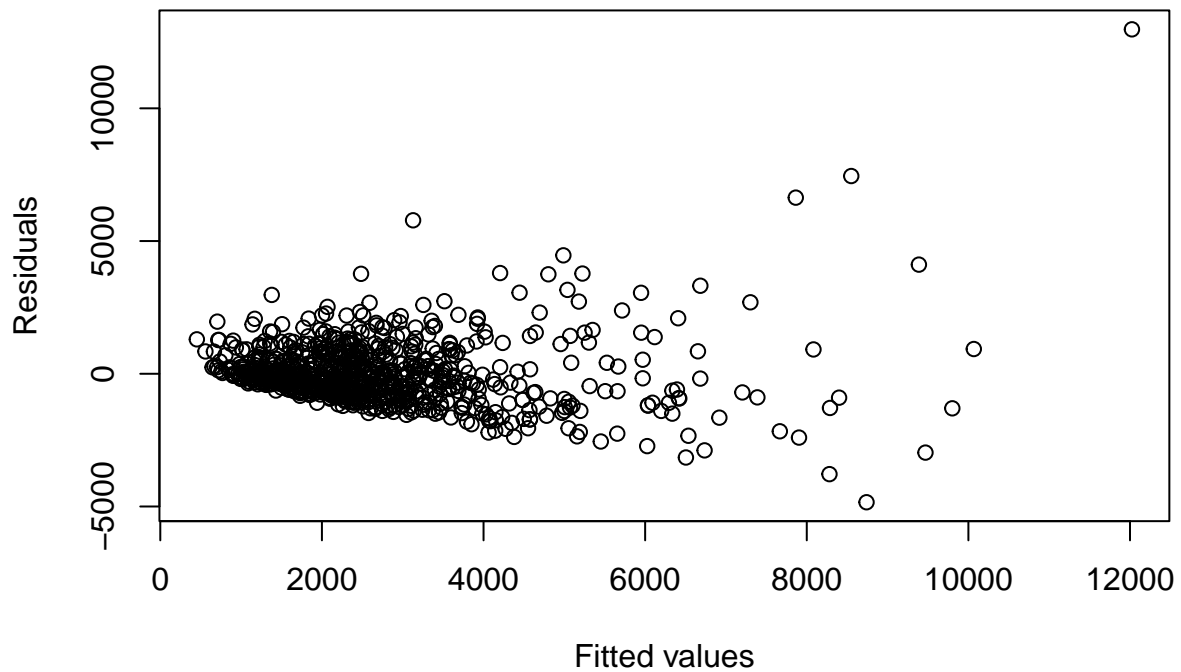
Hacemos un grafico corrplot para ver la correlacion con las variables, donde sacamos que las variables que mas influyen son los metros construidos, los cuarto de baños y las camas. Vemos según diferentes métodos que esas son las variables con mayor correlacion.

```
corrplot(cor(data_train_rg) ,type = "lower",)
```



Ahora pasamos a graficar los residuos en funcion de los valores ajustados, es decir distancias entre los estimados y reales.

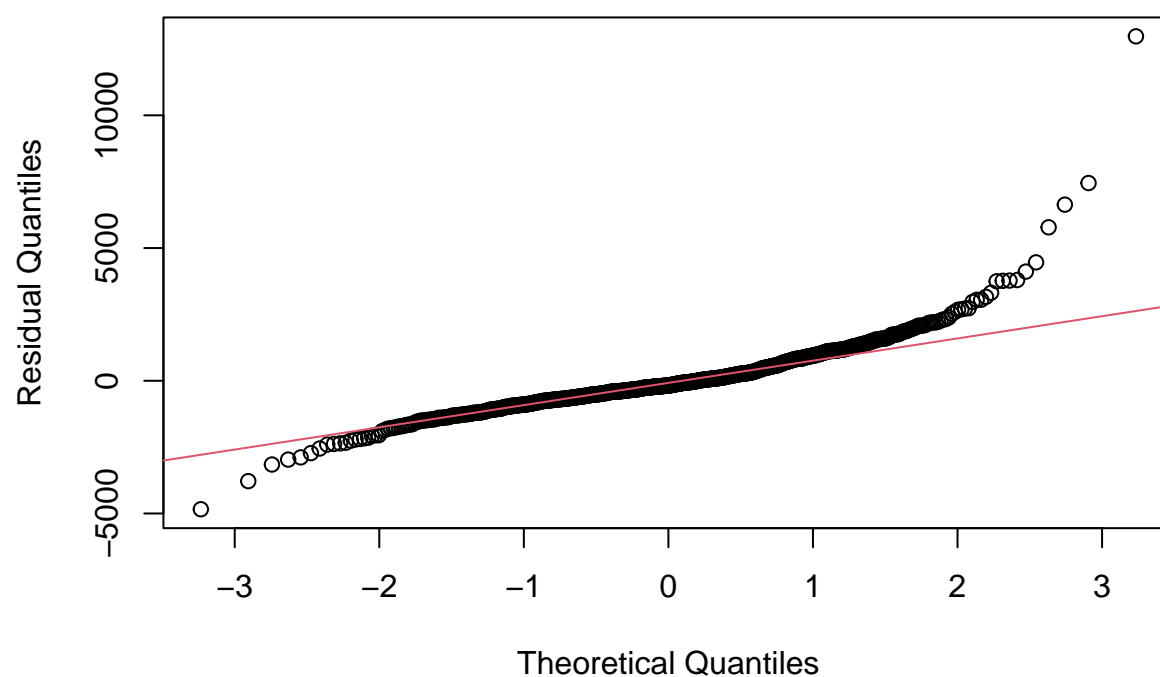
```
plot(regresion_mul$fitted.values, regresion_mul$residuals,
     xlab = "Fitted values", ylab = "Residuals")
```



Sacamos el grafico Q-Q para comprar los residuos de dos distribuciones de probabilidad cuando trazamos los cuantiles entre ellos. No apreciamos ningun patron y podemos intuir que esta formanod una linea por lo que concluimos con que el modelo es bueno.

```
qqnorm(regresion_mul$residuals, ylab = "Residual Quantiles")  
qqline(regresion_mul$residuals, col = 2)
```

## Normal Q-Q Plot



Y vamos a predecir según la regresion multiple, luego compararemos con el arbol de decision.

```
data_val_regression <- predict(regresion_mul, data_val_rg)
data_val_regression
```

##	7	15	16	17	24	28	50
##	1182.1640	1186.8785	1186.8785	1570.4776	1332.6273	2176.8177	2001.0302
##	60	66	73	95	98	129	189
##	2318.6390	1478.0198	2364.4993	2160.1908	3129.6451	2637.8706	1914.5549
##	222	224	246	288	330	373	377
##	1723.3020	5389.9496	1894.3737	5389.9496	2040.9094	1354.3814	1235.1000
##	403	412	429	517	568	569	575
##	2345.1076	4328.1084	1319.2155	2547.7174	1438.1406	1120.5332	2018.0146
##	589	599	603	606	613	636	642
##	1446.8379	2062.6623	2205.6950	1398.6189	2367.2154	2371.5737	1583.8894
##	643	664	682	702	704	718	721
##	2023.1393	2539.0754	6282.7678	2023.1393	1746.2651	2450.9759	6461.2702
##	725	742	1098	1175	1188	1193	1200
##	3095.2468	2099.8254	1851.2819	2143.9132	1392.0030	1799.6596	3170.1603
##	1205	1218	1220	1228	1273	1328	1333
##	1627.6209	2435.0545	1400.5215	1360.9985	2658.8165	2007.6487	1941.4808
##	1356	1378	1384	1392	1414	1427	1429
##	2484.2404	2598.6286	3119.4579	1220.1430	3325.2158	1839.5388	1537.3955
##	1438	1463	1474	1476	1481	1495	1515
##	1082.9128	2540.9241	1008.0477	1795.9095	1775.6021	2774.9510	1612.2607
##	1529	1541	1545	1547	1550	1564	1571

##	3977.4164	1034.5151	1588.0979	1993.6286	716.9064	1976.6455	3143.6929
##	1586	1602	1610	1621	1625	1656	1665
##	1087.4499	2246.2101	2411.3019	2421.6414	2874.6329	3397.3636	1473.3078
##	1674	1692	1713	1737	1777	1787	1806
##	1259.4860	1180.0839	2212.7719	1946.0443	1471.2252	4877.5874	2667.2786
##	1807	1808	1814	1816	1823	1835	1841
##	2159.8371	6061.5457	2962.5561	8124.7908	3470.7846	3457.3728	1698.7360
##	1867	1874	1882	1891	1892	1913	1920
##	1511.5641	3999.3455	3661.7103	948.6745	2616.3974	4212.1778	4670.5145
##	1923	1942	1956	1960	1968	1990	2003
##	1464.6105	3667.6915	2000.5730	2219.3865	4527.0704	1096.3234	1895.1632
##	2013	2016	2063	2069	2079	2086	2094
##	2128.5517	3231.1576	5125.9117	2720.2134	2667.2786	1136.6623	4826.9090
##	2147	2151	2178	2187	2272	2273	2287
##	4483.6199	1822.3757	2186.6607	1981.1800	1149.2582	2245.8539	2570.2598
##	2289	2301	2307	2337	2346	2354	2473
##	2501.0436	1709.4317	1888.0888	6315.0642	1546.2690	1147.0018	1286.1321
##	2482	2496	2499	2503	2507	2511	2558
##	2536.3870	1954.8913	1751.8495	1133.6152	1861.2904	1960.8725	2603.1644
##	2570	2572	2589	2599	2612	2623	2710
##	2126.7513	1510.9269	1554.3774	2691.6200	1514.8544	1117.8436	2179.6861
##	2712	2727	2766	3595	4239	4646	4659
##	2365.3140	2271.7131	3591.1552	5593.5489	3014.0708	4118.6016	1702.9945
##	4672	4680	4699	4704	4706	4724	4763
##	3333.2771	1027.8979	1477.8436	2708.7554	2691.4413	2624.6387	2173.0702
##	4783	4784	4796	4805	4809	4810	4831
##	790.0476	1821.9197	1160.2349	2049.4305	1723.1245	1710.0689	1590.7875
##	4843	4850	4872	4875	4878	4930	4931
##	9945.2372	4374.8845	1301.2691	1292.5718	1695.8702	2186.4820	1776.0593
##	4935	4937	4945	4952	4977	5026	5031
##	1663.1153	1570.6576	1557.2458	2318.1817	2146.6028	1133.7675	1239.6370
##	5042	5059	5074	5609	5613	5619	5622
##	1411.8532	2007.1914	2148.2257	1526.8785	2487.9905	1082.9141	1233.0211
##	5680	5694	5695	5717	5722	6289	6291
##	950.5771	2391.6052	977.0445	1175.7280	1709.4342	1632.5694	4158.6083
##	6299	6305	6316	6317	6332	6349	6354
##	2676.1281	2724.3484	2847.9880	3331.0447	6184.3289	2618.4788	1519.6254
##	6367	6376	6378	6385	6397	6408	6413
##	7180.2493	3344.8873	1941.4808	3179.4670	2748.9119	2084.3612	1921.6293
##	6419	6434	6442	6449	6454	6461	6463
##	4270.4591	2385.4453	1828.5343	4377.1156	2748.9144	5138.0516	3636.0286
##	6486	6488	6500	6505	6506	6511	6512
##	2380.2723	2080.4337	2179.6848	3583.0938	2179.6873	5303.9303	4437.9370
##	6513	6514	6517	6524	6532	6533	6534
##	3158.9783	5848.3153	2523.7609	3386.0332	3000.1740	10259.9763	2576.6957
##	6537	6552	6557	6570	6573	6582	6590
##	6964.6578	3106.0436	4820.4429	2410.8170	2854.7814	2391.4240	3020.6602
##	6595	6600	6603	6612	6624	6630	6631
##	3397.1849	5708.5478	2093.8455	1917.2710	1795.6298	2841.3696	1796.0857
##	6633	6648	6663	6668	6672	6678	6685
##	2345.1076	3370.7175	4787.9943	3247.0778	5185.0051	2371.5750	2133.3685
##	6686	6700	6701	6704	6748	6775	6779
##	1884.1852	1385.2071	2123.8842	1762.5440	3966.6929	1544.0114	3324.4011
##	6780	6786	6794	6817	6826	6840	6859

```
## 5084.9670 9128.6466 4511.0754 6002.6285 3325.2133 10067.2739 3201.5761
##      6873      6875      6879      6896      6900      6903      6938
## 2664.3392 3986.5419 5052.2122 2887.8672 2609.7815 6035.5355 4028.6812
##      6965      6975      7004      7076      7116      7119      7145
## 4347.9573 2431.3057 2235.6944 4390.5274 3324.4011 5435.8113 3139.1294
##      7164      7173      7174      7183      7228      7242      7251
## 8088.8379 2854.7814 4055.1486 2684.8254 1422.6764 3006.7924 1632.1109
##      7283      7289      7300      7882      7920      7929      7964
## 2744.1999 3946.5629 2358.5194 1998.3419 2742.2960 2961.1095 2205.8737
##      8000      8017      8025      8059      8061      8069      8083
## 1917.0923 1457.9921 5949.5150 6183.0569 1226.5802 3945.9270 3258.8680
##      8084      8115      8120      8142      8148      8152      8228
## 2172.2820 2404.8370 1379.0472 2050.6407 3397.6434 2186.4807 3020.0255
##      8243      8284      8312      8313      8332      8360      8383
## 3675.7291 2437.7428 1994.4143 3397.6434 5694.3253 1431.5247 2815.7169
##      8400      8406      8411      8420      8428      8441      8455
## 5007.7960 2484.6964 2417.8926 2437.7428 2173.0689 2411.2755 1670.5447
##      8458      8500      8503      8509      8522      8536      8540
## 2470.8274 2814.9035 3864.1149 902.1769 1563.8617 2040.7320 4213.7742
##      8590      8621      8645      8649      8653      8673      8680
## 1742.5163 2007.1902 1696.1986 1742.5163 1087.4486 3125.8950 1888.5448
##      8715      8747      8774      8783      8879      8882      8893
## 2841.8281 4946.1639 540.6869 1704.8959 4833.2199 2514.2779 5694.3253
##      8907      9002      9095      9132      9133
## 2755.3515 1849.0218 1888.2650 963.8089 2086.5923
```

Vamos a calcular el RMSE. Primero hacemos la diferencias entre los valores reales y los predichos, luego el residual de la suma y el valor del acierto

```
dif <- data_val_rg$price-data_val_regression
rmse <- sqrt(mean(dif ^ 2))
rmse
```

```
## [1] 1255.776
```

Vemos que el RSME es una diferencia bastante grande. Pasamos a calcular el acierto

```
residual <- sum(dif^2)
total <- sum((data_val_rg$price - mean(data_val_rg$price)) ^ 2)
acierto_rg <- 1 - (residual / total)
acierto_rg
```

```
## [1] 0.5832167
```

No supervisado: Clustering con k-means

Primero pasamos a ver que las variables sean numericas

```
lapply(data,class)
```

```
## $price
## [1] "numeric"
```

```
##
## $floor_built
## [1] "numeric"
##
## $bathrooms
## [1] "numeric"
##
## $terrace
## [1] "numeric"
##
## $bedrooms
## [1] "numeric"
##
## $postalcode
## [1] "numeric"
##
## $garage_included
## [1] "numeric"
```

Comprobamos si necesitamos escalarlas.

```
summary(data)
```

```
##      price      floor_built      bathrooms      terrace
## Min.   : 725   Min.   : 30.0   Min.   : 1.000   Min.   :0.0000
## 1st Qu.:1450   1st Qu.: 80.0   1st Qu.: 1.000   1st Qu.:0.0000
## Median :2200   Median :101.0   Median : 2.000   Median :0.0000
## Mean   :2698   Mean   :119.6   Mean   : 2.016   Mean   :0.2765
## 3rd Qu.:3350   3rd Qu.:140.0   3rd Qu.: 2.000   3rd Qu.:1.0000
## Max.   :25000   Max.   :512.0   Max.   :20.000   Max.   :1.0000
##      bedrooms      postalcode      garage_included
## Min.   :2.000   Min.   :28001   Min.   :0.0000
## 1st Qu.:2.000   1st Qu.:28003   1st Qu.:0.0000
## Median :2.000   Median :28006   Median :0.0000
## Mean   :2.643   Mean   :28006   Mean   :0.1841
## 3rd Qu.:3.000   3rd Qu.:28008   3rd Qu.:0.0000
## Max.   :6.000   Max.   :28011   Max.   :1.0000
```

Vemos que hay mucha diferencia entre el maximo y el minimo por lo cual sería necesario escalarlo, para tener una mejor vision global

```
data_scaled = scale(data)
summary(data_scaled)
```

```
##      price      floor_built      bathrooms      terrace
## Min.   :-1.0237   Min.   :-1.4362   Min.   :-0.97777   Min.   :-0.618
## 1st Qu.: -0.6476   1st Qu.: -0.6350   1st Qu.: -0.97777   1st Qu.: -0.618
## Median : -0.2584   Median : -0.2985   Median : -0.01565   Median : -0.618
## Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.000
## 3rd Qu.: 0.3382   3rd Qu.: 0.3265   3rd Qu.: -0.01565   3rd Qu.: 1.617
## Max.   :11.5707   Max.   : 6.2877   Max.   :17.30258   Max.   : 1.617
##      bedrooms      postalcode      garage_included
```

```
## Min.    :-0.7999   Min.    :-1.6429   Min.    :-0.4748
## 1st Qu.: -0.7999   1st Qu.: -0.9334   1st Qu.: -0.4748
## Median : -0.7999   Median :  0.1309   Median : -0.4748
## Mean    :  0.0000   Mean     :  0.0000   Mean     :  0.0000
## 3rd Qu.:  0.4441   3rd Qu.:  0.8405   3rd Qu.: -0.4748
## Max.    :  4.1761   Max.     :  1.9048   Max.     :  2.1045
```

Una vez los datos bien estructurados, pasamos a calcular el mejor valor de k. Para ello iremos iterando el valor de 1 a 20 para guardar el valor de compactación en cada iteración, luego lo graficaremos y veremos a qué k que produce un cambio considerable. Inicializamos la variable nstart a 10 para controlar la aleatoriedad.

```
v_compac <- 0

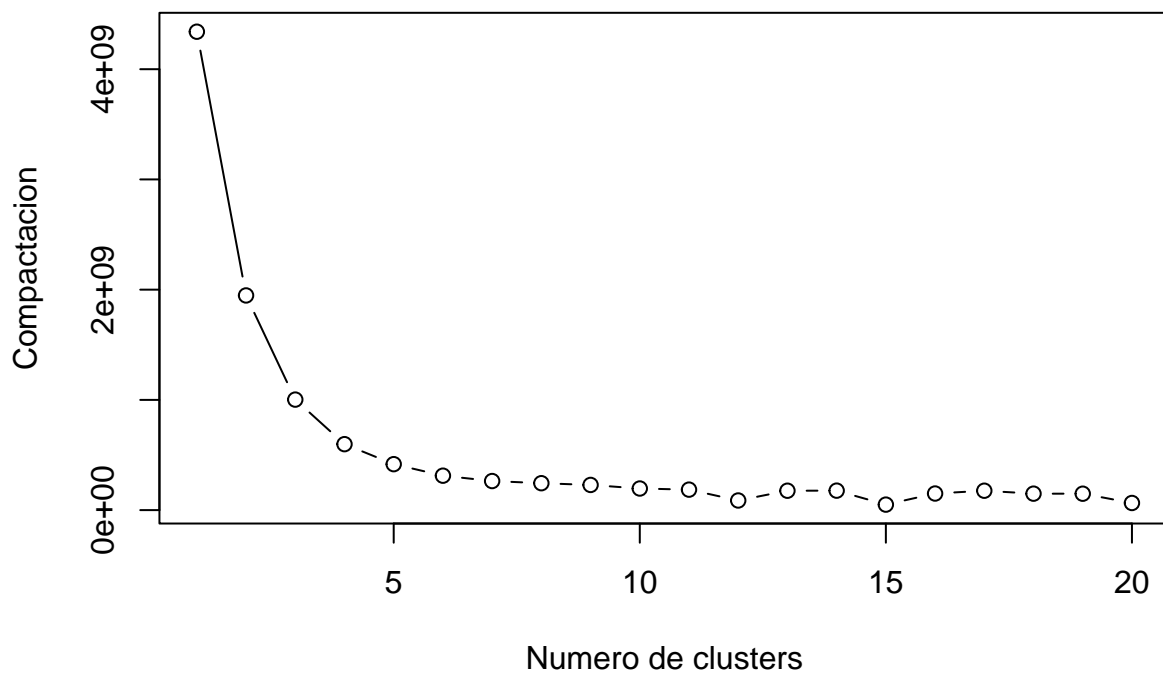
for(i in 1:20){

  km1<-kmeans(data,center=i,nstart=10)
  v_compac[i] <- km1$tot.withinss

}

par(mfrow = c(1,1))

plot(1:20, v_compac, type = "b",
     xlab = "Numero de clusters",
     ylab = "Compactacion")
```





Observamos que en k=3 y k=4 empieza a ver un cambio significativo, por lo cual escogemos estos valores para K. Pasamos a ver los resultados y los comparamos

K=3

```
kmeans3 <- kmeans(data, center =3,nstart= 10)
kmeans3

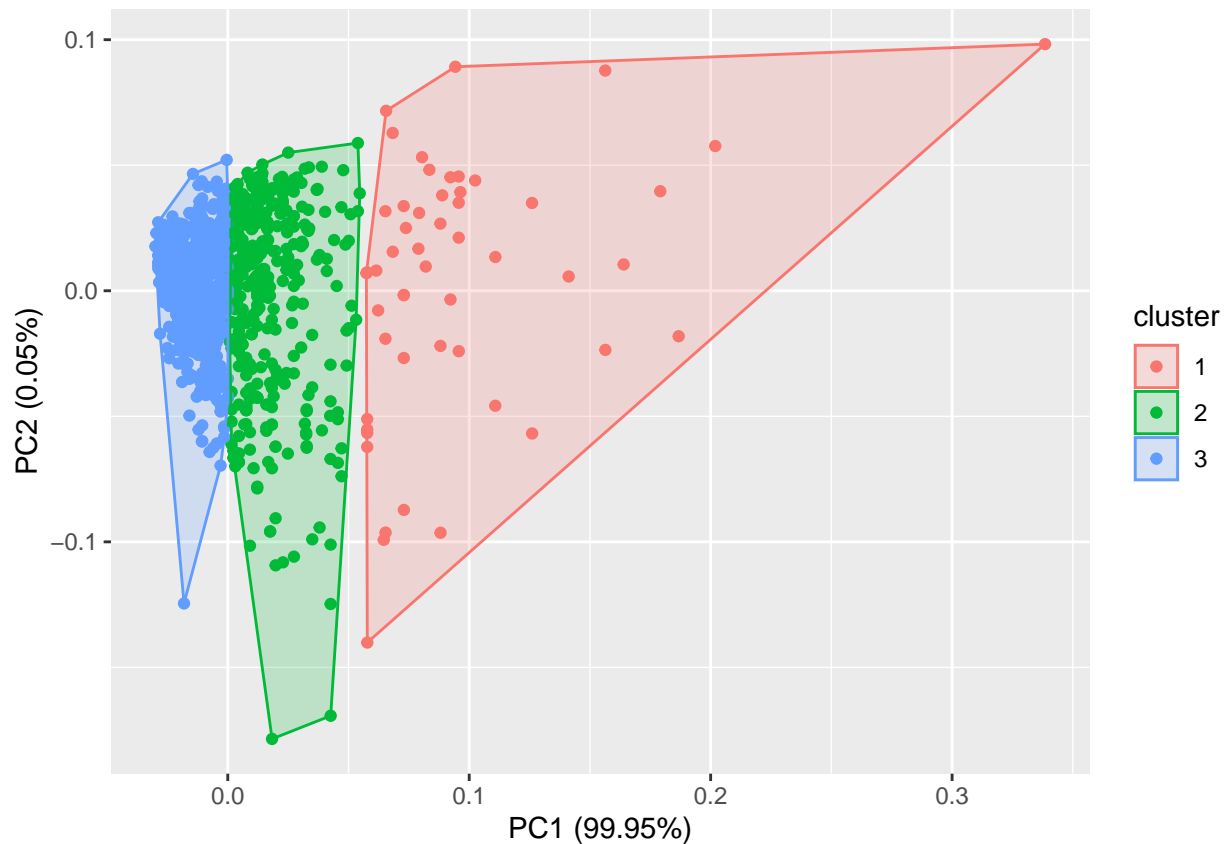
## K-means clustering with 3 clusters of sizes 52, 377, 739
##
## Cluster means:
##      price floor_built bathrooms   terrace bedrooms postalcode garage_included
## 1 9088.096   268.13462  3.846154 0.3846154 3.557692   28003.69    0.3269231
## 2 3796.255   149.38992  2.541114 0.2758621 2.811671   28005.33    0.1962865
## 3 1688.304    93.99323  1.619756 0.2692828 2.492558   28005.92    0.1677943
##
## Clustering vector:
##      7   15   16   17   18   24   28   29   39   42   45   47   50   58   60   66
##      3    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
##     71   73   76   82   83   95   96   98  107  108  120  122  129  133  136  137
##      3    3    3    3    3    3    3    3    3    3    3    3    3    2    3    3
##    140  144  160  162  164  172  173  189  193  204  207  209  210  222  224  225
##      3    3    3    3    3    2    2    3    3    3    3    2    3    3    1    1
##    226  227  237  240  246  248  250  252  257  266  268  270  284  286  288  315
##      1    2    3    2    3    3    3    3    3    3    3    3    2    1    1    3
##    320  321  330  331  332  343  347  356  366  367  368  373  377  378  382  401
##      3    3    3    3    1    3    3    3    2    3    3    3    3    3    3    3
##    403  412  418  428  429  432  441  442  443  448  449  485  489  499  501  502
##      3    2    3    2    3    1    3    2    3    3    3    3    3    3    3    3
##    509  517  524  533  535  545  550  552  555  566  568  569  572  575  579  589
##      3    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
##    590  599  600  603  606  613  628  636  642  643  664  668  673  682  690  693
##      3    3    3    3    3    3    3    3    3    3    3    3    3    2    3    3
##    702  704  705  709  710  713  718  721  725  733  742  1098  1173  1175  1176  1177
##      3    3    3    3    3    3    3    2    3    3    3    3    2    3    3    3
##   1179 1181 1182 1184 1188 1193 1199 1200 1203 1204 1205 1212 1214 1218 1220 1223
##      3    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
##   1228 1230 1231 1236 1239 1244 1245 1252 1264 1268 1269 1271 1272 1273 1277 1279
##      3    2    2    3    3    3    3    3    2    3    3    3    2    2    3    3
##   1282 1285 1296 1301 1305 1307 1317 1318 1319 1321 1325 1327 1328 1333 1335 1340
##      3    3    3    3    2    3    2    3    3    3    3    2    2    3    3    3
##   1345 1348 1349 1356 1362 1366 1367 1374 1376 1378 1379 1384 1385 1391 1392 1397
##      3    3    3    3    2    2    3    3    3    3    3    3    2    3    3    3
##   1400 1401 1402 1403 1409 1414 1415 1418 1420 1421 1422 1427 1428 1429 1432 1436
##      3    3    3    2    3    3    3    3    3    2    3    3    2    3    3    3
##   1437 1438 1441 1450 1453 1454 1456 1461 1463 1468 1469 1473 1474 1476 1478 1481
##      3    3    3    3    3    3    3    3    3    2    3    3    3    3    3    3
##   1495 1496 1507 1508 1512 1515 1529 1530 1537 1540 1541 1545 1546 1547 1548 1550
##      2    3    3    3    3    3    3    3    3    3    3    3    2    3    3    3
##   1554 1560 1564 1566 1571 1575 1584 1586 1592 1594 1596 1598 1599 1600 1602 1603
##      3    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
##   1605 1610 1621 1622 1623 1625 1627 1629 1630 1631 1639 1643 1656 1665 1670 1671
##      3    3    3    3    3    2    3    3    3    3    3    2    3    3    3    3
##   1672 1674 1687 1692 1693 1695 1700 1710 1713 1716 1722 1724 1725 1727 1729 1733
```

##	3	3	3	3	3	3	2	3	3	3	2	3	3	3	3	2
##	1737	1741	1748	1749	1754	1756	1761	1763	1767	1768	1771	1773	1777	1780	1787	1793
##	3	3	3	2	2	2	2	3	2	3	3	2	3	3	2	2
##	1797	1801	1806	1807	1808	1812	1814	1816	1821	1822	1823	1829	1833	1834	1835	1836
##	3	3	3	2	2	3	2	2	2	2	2	2	2	3	2	2
##	1838	1840	1841	1853	1855	1857	1858	1861	1864	1867	1871	1874	1879	1882	1883	1885
##	3	3	3	1	2	3	2	2	2	2	3	2	2	2	3	2
##	1888	1889	1891	1892	1904	1905	1906	1908	1911	1913	1915	1918	1920	1923	1924	1930
##	2	3	3	2	2	2	2	2	2	2	2	2	2	3	3	2
##	1934	1938	1939	1942	1945	1946	1948	1949	1953	1954	1956	1957	1960	1962	1965	1968
##	3	3	3	2	3	2	2	1	3	2	3	2	3	3	3	2
##	1969	1970	1971	1972	1975	1976	1977	1978	1983	1990	1991	1993	2003	2007	2008	2012
##	2	1	2	3	2	3	1	3	3	3	3	3	2	3	2	2
##	2013	2016	2020	2021	2041	2045	2049	2056	2057	2063	2069	2070	2079	2082	2084	2086
##	3	2	2	3	2	3	3	3	3	2	3	2	3	3	2	3
##	2088	2090	2091	2093	2094	2099	2101	2103	2107	2108	2115	2116	2144	2147	2151	2164
##	3	3	3	2	2	2	3	3	3	2	2	2	3	2	3	3
##	2165	2167	2172	2175	2178	2184	2187	2190	2197	2204	2205	2208	2209	2213	2228	2230
##	3	2	3	2	2	3	2	2	2	3	3	2	3	3	2	3
##	2232	2241	2251	2269	2270	2272	2273	2283	2286	2287	2289	2300	2301	2305	2307	2317
##	3	2	3	3	3	3	2	3	3	2	3	3	3	2	3	3
##	2335	2337	2341	2346	2350	2351	2354	2358	2364	2368	2473	2482	2483	2485	2495	2496
##	2	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	2497	2499	2503	2507	2509	2511	2513	2519	2521	2530	2538	2549	2552	2557	2558	2563
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	2568	2570	2572	2579	2589	2599	2602	2605	2610	2612	2614	2620	2621	2623	2641	2645
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	2661	2662	2670	2682	2689	2710	2712	2727	2739	2744	2746	2747	2751	2764	2766	2768
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	2771	3046	3595	4239	4646	4649	4652	4653	4655	4659	4661	4663	4664	4665	4670	4672
##	3	3	3	2	3	3	3	3	3	3	3	3	3	3	2	3
##	4678	4679	4680	4682	4697	4699	4700	4704	4706	4724	4731	4743	4745	4763	4768	4769
##	3	3	3	3	3	3	3	3	3	2	3	3	3	3	2	2
##	4770	4771	4772	4779	4783	4784	4792	4793	4794	4796	4797	4804	4805	4806	4809	4810
##	2	2	2	2	3	3	3	2	2	3	3	3	3	1	3	3
##	4826	4831	4833	4838	4843	4845	4850	4853	4859	4872	4873	4874	4875	4878	4879	4890
##	3	3	3	3	2	2	2	3	3	2	3	3	3	2	3	3
##	4898	4900	4905	4923	4930	4931	4935	4937	4938	4945	4952	4958	4959	4961	4965	4968
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	4977	4982	4983	4987	4994	5000	5007	5014	5016	5026	5031	5032	5033	5042	5043	5046
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1
##	5059	5069	5070	5071	5074	5075	5606	5609	5610	5613	5615	5619	5622	5624	5630	5635
##	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3
##	5637	5642	5643	5644	5669	5677	5680	5684	5687	5690	5693	5694	5695	5700	5701	5706
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	5712	5715	5717	5719	5722	5724	5726	5736	5737	5742	5743	5749	6229	6235	6240	6246
##	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
##	6273	6277	6278	6279	6282	6288	6289	6291	6299	6305	6308	6316	6317	6319	6327	6329
##	2	2	1	2	3	2	3	2	3	3	3	2	3	3	2	2
##	6330	6331	6332	6336	6349	6350	6354	6363	6367	6376	6377	6378	6379	6384	6385	6387
##	3	2	2	1	3	3	3	2	2	2	2	2	1	3	2	2
##	6389	6397	6398	6400	6402	6408	6413	6418	6419	6426	6430	6432	6434	6436	6442	6448
##	3	3	3	3	3	3	3	3	2	2	3	1	2	1	3	2
##	6449	6452	6454	6458	6461	6463	6465	6468	6471	6475	6481	6482	6484	6486	6488	6496

##	2	3	2	2	2	3	2	2	1	2	2	3	3	2	2	2
##	6497	6500	6502	6503	6505	6506	6510	6511	6512	6513	6514	6515	6516	6517	6518	6519
##	2	2	2	2	2	2	1	1	1	2	1	3	2	2	3	2
##	6520	6522	6524	6525	6526	6527	6530	6532	6533	6534	6536	6537	6538	6542	6543	6546
##	2	2	2	3	3	2	2	2	1	2	3	2	2	2	2	3
##	6549	6551	6552	6553	6554	6556	6557	6558	6566	6568	6569	6570	6573	6575	6576	6578
##	2	2	2	2	2	2	2	3	2	2	2	2	1	2	2	2
##	6580	6581	6582	6587	6589	6590	6591	6595	6596	6597	6598	6599	6600	6602	6603	6607
##	2	2	2	2	2	2	1	2	2	2	2	2	2	3	3	2
##	6612	6613	6617	6619	6620	6624	6629	6630	6631	6633	6635	6636	6645	6646	6648	6656
##	3	3	3	2	2	3	1	2	2	2	2	2	2	2	2	2
##	6657	6658	6660	6661	6662	6663	6665	6668	6670	6671	6672	6673	6675	6677	6678	6680
##	2	2	2	2	1	1	2	2	1	2	1	2	2	1	2	3
##	6681	6682	6683	6684	6685	6686	6687	6688	6693	6694	6697	6699	6700	6701	6703	6704
##	2	2	2	2	3	2	2	2	2	2	2	2	3	2	2	3
##	6708	6713	6714	6717	6718	6720	6740	6742	6743	6748	6749	6750	6752	6764	6773	6774
##	3	3	1	2	2	1	2	2	2	2	1	2	3	2	2	2
##	6775	6779	6780	6781	6786	6788	6789	6794	6795	6800	6801	6806	6817	6818	6819	6820
##	3	3	2	2	1	3	3	3	2	3	2	2	2	2	2	2
##	6823	6826	6831	6833	6839	6840	6841	6847	6859	6870	6873	6875	6877	6879	6880	6882
##	2	2	2	3	3	1	2	2	3	3	3	1	3	2	3	2
##	6884	6896	6900	6902	6903	6913	6916	6921	6938	6943	6960	6965	6966	6974	6975	6976
##	2	3	2	2	2	3	2	3	2	1	1	2	2	3	2	1
##	6981	6993	7004	7014	7015	7018	7043	7045	7049	7051	7064	7075	7076	7081	7086	7087
##	3	1	3	3	3	3	3	2	3	3	3	3	2	3	2	2
##	7088	7091	7099	7110	7116	7119	7125	7134	7145	7156	7158	7164	7172	7173	7174	7183
##	2	3	3	3	3	2	1	2	3	3	3	1	2	2	2	2
##	7184	7213	7218	7228	7233	7242	7250	7251	7260	7264	7267	7268	7283	7287	7289	7297
##	3	3	3	3	3	3	3	3	3	3	2	3	3	2	3	2
##	7300	7305	7306	7310	7325	7327	7331	7334	7880	7882	7883	7888	7895	7905	7916	7918
##	3	2	2	3	3	3	3	3	3	3	3	3	3	3	2	3
##	7920	7923	7929	7946	7949	7957	7964	7978	7999	8000	8016	8017	8022	8025	8028	8050
##	3	3	3	2	3	1	3	3	3	3	3	3	3	1	2	3
##	8051	8055	8059	8061	8069	8072	8082	8083	8084	8103	8111	8114	8115	8118	8120	8126
##	3	3	2	3	2	3	3	2	3	2	3	3	3	2	3	2
##	8142	8145	8148	8152	8173	8175	8176	8181	8184	8200	8201	8205	8206	8214	8228	8230
##	3	3	2	3	2	2	3	2	3	3	2	2	2	2	2	2
##	8232	8236	8242	8243	8260	8261	8262	8272	8280	8284	8290	8292	8293	8301	8303	8311
##	2	2	2	2	2	2	2	3	2	2	2	2	2	2	1	3
##	8312	8313	8314	8324	8332	8334	8339	8342	8345	8346	8352	8360	8365	8382	8383	8395
##	3	2	3	2	1	2	2	2	1	3	3	3	2	2	2	2
##	8397	8400	8405	8406	8407	8411	8412	8413	8420	8422	8423	8428	8437	8441	8442	8448
##	2	2	2	2	2	3	3	2	2	3	2	3	2	2	2	2
##	8453	8455	8457	8458	8468	8489	8500	8503	8509	8522	8535	8536	8540	8565	8566	8577
##	3	3	3	3	3	3	3	3	3	3	2	2	2	3	3	2
##	8578	8583	8590	8604	8621	8622	8645	8647	8649	8650	8653	8660	8672	8673	8680	8691
##	3	3	3	3	3	2	3	2	3	1	3	3	3	2	2	3
##	8692	8693	8699	8702	8715	8735	8743	8745	8747	8750	8755	8766	8768	8774	8776	8783
##	3	3	3	3	2	2	3	3	2	3	2	3	2	3	3	3
##	8789	8791	8796	8805	8814	8815	8817	8819	8831	8838	8840	8842	8843	8856	8870	8879
##	2	3	2	3	3	3	3	2	3	3	3	3	2	3	3	2
##	8882	8886	8891	8893	8901	8907	8911	8915	8918	8921	8924	8934	8938	8965	8968	8972
##	3	2	3	1	2	3	3	3	3	3	2	3	2	3	2	3
##	8989	8991	8998	9002	9004	9008	9030	9044	9057	9076	9095	9112	9121	9125	9132	9133



```
autoplot(kmeans3, data, frame=TRUE)
```



K=4

```
kmeans4 <- kmeans(data, center =4,nstart= 10)
kmeans4
```

```
## K-means clustering with 4 clusters of sizes 401, 25, 112, 630
```

```
##
```

```
## Cluster means:
```

```
##      price floor_built bathrooms  terrace bedrooms postcode garage_included
## 1  3173.379   133.52618  2.331671  0.2992519  2.743142   28005.34    0.1845387
## 2 11160.000   296.68000  4.000000  0.5200000  3.680000   28003.68    0.5200000
## 3  5609.545   200.62500  3.089286  0.2232143  3.133929   28005.04    0.2321429
## 4  1542.273    89.35397  1.546032  0.2619048  2.450794   28006.00    0.1619048
```

```
##
```

```
## Clustering vector:
```

```
##      7  15  16  17  18  24  28  29  39  42  45  47  50  58  60  66
##      4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4
##     71  73  76  82  83  95  96  98 107 108 120 122 129 133 136 137
##      1   4   4   4   4   4   4   1   4   4   4   4   4   1   4   4
##    140 144 160 162 164 172 173 189 193 204 207 209 210 222 224 225
##      4   1   4   4   1   3   1   4   4   4   4   1   1   4   3   2
##    226 227 237 240 246 248 250 252 257 266 268 270 284 286 288 315
##      2   3   4   1   4   4   1   4   4   4   4   1   1   2   3   4
```

##	320	321	330	331	332	343	347	356	366	367	368	373	377	378	382	401
##	4	4	4	4	3	1	4	4	1	4	4	4	1	1	1	4
##	403	412	418	428	429	432	441	442	443	448	449	485	489	499	501	502
##	4	1	4	3	4	3	4	3	4	4	4	1	4	4	4	4
##	509	517	524	533	535	545	550	552	555	566	568	569	572	575	579	589
##	4	4	4	4	4	4	4	1	4	1	4	4	4	4	4	4
##	590	599	600	603	606	613	628	636	642	643	664	668	673	682	690	693
##	4	4	4	4	4	4	4	4	4	4	4	4	4	1	1	1
##	702	704	705	709	710	713	718	721	725	733	742	1098	1173	1175	1176	1177
##	1	4	4	4	4	4	4	1	1	4	4	4	1	4	4	4
##	1179	1181	1182	1184	1188	1193	1199	1200	1203	1204	1205	1212	1214	1218	1220	1223
##	4	4	4	4	4	1	4	4	4	4	4	4	4	4	4	4
##	1228	1230	1231	1236	1239	1244	1245	1252	1264	1268	1269	1271	1272	1273	1277	1279
##	4	1	1	1	4	4	4	4	1	4	4	4	1	1	4	4
##	1282	1285	1296	1301	1305	1307	1317	1318	1319	1321	1325	1327	1328	1333	1335	1340
##	1	4	4	4	1	4	1	1	4	1	4	1	1	4	1	1
##	1345	1348	1349	1356	1362	1366	1367	1374	1376	1378	1379	1384	1385	1391	1392	1397
##	4	4	4	4	1	1	4	4	4	4	4	4	1	4	4	4
##	1400	1401	1402	1403	1409	1414	1415	1418	1420	1421	1422	1427	1428	1429	1432	1436
##	4	4	4	1	4	4	4	4	4	1	4	4	1	4	4	4
##	1437	1438	1441	1450	1453	1454	1456	1461	1463	1468	1469	1473	1474	1476	1478	1481
##	4	4	4	4	4	4	4	4	4	1	4	4	4	4	4	4
##	1495	1496	1507	1508	1512	1515	1529	1530	1537	1540	1541	1545	1546	1547	1548	1550
##	1	4	4	4	4	4	1	4	4	4	4	4	1	4	4	4
##	1554	1560	1564	1566	1571	1575	1584	1586	1592	1594	1596	1598	1599	1600	1602	1603
##	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
##	1605	1610	1621	1622	1623	1625	1627	1629	1630	1631	1639	1643	1656	1665	1670	1671
##	4	1	4	4	4	3	4	4	4	4	4	1	4	4	1	4
##	1672	1674	1687	1692	1693	1695	1700	1710	1713	1716	1722	1724	1725	1727	1729	1733
##	4	4	4	4	4	1	1	4	4	4	1	4	4	4	4	1
##	1737	1741	1748	1749	1754	1756	1761	1763	1767	1768	1771	1773	1777	1780	1787	1793
##	1	1	4	1	1	1	1	1	1	1	4	3	4	1	3	1
##	1797	1801	1806	1807	1808	1812	1814	1816	1821	1822	1823	1829	1833	1834	1835	1836
##	4	1	4	1	3	4	1	3	1	1	3	3	3	1	1	3
##	1838	1840	1841	1853	1855	1857	1858	1861	1864	1867	1871	1874	1879	1882	1883	1885
##	1	1	1	2	1	4	1	1	1	1	4	3	1	3	4	1
##	1888	1889	1891	1892	1904	1905	1906	1908	1911	1913	1915	1918	1920	1923	1924	1930
##	1	1	4	1	1	1	1	3	1	1	3	1	1	4	4	3
##	1934	1938	1939	1942	1945	1946	1948	1949	1953	1954	1956	1957	1960	1962	1965	1968
##	4	4	4	1	1	1	1	3	4	1	1	1	4	4	4	1
##	1969	1970	1971	1972	1975	1976	1977	1978	1983	1990	1991	1993	2003	2007	2008	2012
##	1	3	3	4	3	1	3	4	4	4	4	4	1	4	1	1
##	2013	2016	2020	2021	2041	2045	2049	2056	2057	2063	2069	2070	2079	2082	2084	2086
##	4	1	3	4	1	4	1	4	4	1	4	1	4	4	3	4
##	2088	2090	2091	2093	2094	2099	2101	2103	2107	2108	2115	2116	2144	2147	2151	2164
##	4	4	4	3	1	1	4	4	4	1	1	1	4	1	4	4
##	2165	2167	2172	2175	2178	2184	2187	2190	2197	2204	2205	2208	2209	2213	2228	2230
##	4	1	4	1	1	4	1	1	1	4	1	1	4	4	1	4
##	2232	2241	2251	2269	2270	2272	2273	2283	2286	2287	2289	2300	2301	2305	2307	2317
##	4	1	1	1	4	4	1	4	4	1	4	4	4	1	4	4
##	2335	2337	2341	2346	2350	2351	2354	2358	2364	2368	2473	2482	2483	2485	2495	2496
##	1	2	4	4	1	4	4	4	4	4	4	4	4	4	4	4
##	2497	2499	2503	2507	2509	2511	2513	2519	2521	2530	2538	2549	2552	2557	2558	2563
##	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

##	2568	2570	2572	2579	2589	2599	2602	2605	2610	2612	2614	2620	2621	2623	2641	2645
##	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
##	2661	2662	2670	2682	2689	2710	2712	2727	2739	2744	2746	2747	2751	2764	2766	2768
##	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
##	2771	3046	3595	4239	4646	4649	4652	4653	4655	4659	4661	4663	4664	4665	4670	4672
##	4	4	4	1	1	4	4	4	4	4	4	4	4	4	1	1
##	4678	4679	4680	4682	4697	4699	4700	4704	4706	4724	4731	4743	4745	4763	4768	4769
##	1	4	4	4	4	4	4	4	4	1	4	4	4	4	1	1
##	4770	4771	4772	4779	4783	4784	4792	4793	4794	4796	4797	4804	4805	4806	4809	4810
##	1	1	1	1	4	4	4	1	1	4	4	4	4	3	4	4
##	4826	4831	4833	4838	4843	4845	4850	4853	4859	4872	4873	4874	4875	4878	4879	4890
##	4	1	4	4	3	1	1	4	4	1	1	4	4	1	1	4
##	4898	4900	4905	4923	4930	4931	4935	4937	4938	4945	4952	4958	4959	4961	4965	4968
##	4	4	4	4	4	4	4	4	4	4	4	4	1	1	4	4
##	4977	4982	4983	4987	4994	5000	5007	5014	5016	5026	5031	5032	5033	5042	5043	5046
##	4	4	1	4	4	4	4	4	4	4	4	4	4	4	4	2
##	5059	5069	5070	5071	5074	5075	5606	5609	5610	5613	5615	5619	5622	5624	5630	5635
##	4	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4
##	5637	5642	5643	5644	5669	5677	5680	5684	5687	5690	5693	5694	5695	5700	5701	5706
##	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
##	5712	5715	5717	5719	5722	5724	5726	5736	5737	5742	5743	5749	6229	6235	6240	6246
##	4	4	4	4	4	4	4	4	4	4	4	4	1	4	4	4
##	6273	6277	6278	6279	6282	6288	6289	6291	6299	6305	6308	6316	6317	6319	6327	6329
##	1	1	2	1	4	3	4	1	1	1	4	1	1	4	1	1
##	6330	6331	6332	6336	6349	6350	6354	6363	6367	6376	6377	6378	6379	6384	6385	6387
##	1	3	1	2	4	4	4	3	3	1	1	1	2	4	3	3
##	6389	6397	6398	6400	6402	6408	6413	6418	6419	6426	6430	6432	6434	6436	6442	6448
##	1	4	4	4	4	4	4	1	1	1	4	3	1	2	4	1
##	6449	6452	6454	6458	6461	6463	6465	6468	6471	6475	6481	6482	6484	6486	6488	6496
##	1	4	1	1	3	4	1	1	2	1	1	4	1	1	1	1
##	6497	6500	6502	6503	6505	6506	6510	6511	6512	6513	6514	6515	6516	6517	6518	6519
##	1	1	1	1	3	1	3	3	2	1	2	1	1	1	1	1
##	6520	6522	6524	6525	6526	6527	6530	6532	6533	6534	6536	6537	6538	6542	6543	6546
##	1	1	1	4	4	1	1	1	2	1	1	1	1	3	3	4
##	6549	6551	6552	6553	6554	6556	6557	6558	6566	6568	6569	6570	6573	6575	6576	6578
##	1	3	1	1	3	1	3	4	3	3	1	1	3	1	3	1
##	6580	6581	6582	6587	6589	6590	6591	6595	6596	6597	6598	6599	6600	6602	6603	6607
##	3	1	1	1	1	3	3	1	3	3	3	3	3	4	4	3
##	6612	6613	6617	6619	6620	6624	6629	6630	6631	6633	6635	6636	6645	6646	6648	6656
##	1	4	4	1	3	4	3	1	1	1	1	1	1	1	3	1
##	6657	6658	6660	6661	6662	6663	6665	6668	6670	6671	6672	6673	6675	6677	6678	6680
##	3	3	3	1	3	3	1	1	3	3	2	3	1	2	1	1
##	6681	6682	6683	6684	6685	6686	6687	6688	6693	6694	6697	6699	6700	6701	6703	6704
##	1	1	1	1	1	1	3	1	1	1	1	1	4	1	1	4
##	6708	6713	6714	6717	6718	6720	6740	6742	6743	6748	6749	6750	6752	6764	6773	6774
##	4	1	3	1	3	2	1	3	1	1	3	1	4	1	1	3
##	6775	6779	6780	6781	6786	6788	6789	6794	6795	6800	6801	6806	6817	6818	6819	6820
##	4	1	1	1	3	1	1	1	1	1	3	1	3	1	1	1
##	6823	6826	6831	6833	6839	6840	6841	6847	6859	6870	6873	6875	6877	6879	6880	6882
##	3	1	1	4	4	2	1	3	1	1	1	3	4	1	4	3
##	6884	6896	6900	6902	6903	6913	6916	6921	6938	6943	6960	6965	6966	6974	6975	6976
##	1	1	1	3	1	1	1	4	1	3	3	1	1	4	1	3
##	6981	6993	7004	7014	7015	7018	7043	7045	7049	7051	7064	7075	7076	7081	7086	7087
##	4	2	4	4	4	4	4	1	4	4	4	4	1	4	1	1

```

## 7088 7091 7099 7110 7116 7119 7125 7134 7145 7156 7158 7164 7172 7173 7174 7183
##      3      4      4      1      4      1      2      3      4      1      4      2      1      1      1      1
## 7184 7213 7218 7228 7233 7242 7250 7251 7260 7264 7267 7268 7283 7287 7289 7297
##      1      1      1      4      4      1      4      4      4      1      3      4      4      1      1      1
## 7300 7305 7306 7310 7325 7327 7331 7334 7880 7882 7883 7888 7895 7905 7916 7918
##      4      1      1      1      4      4      4      4      1      4      4      4      4      4      1      4
## 7920 7923 7929 7946 7949 7957 7964 7978 7999 8000 8016 8017 8022 8025 8028 8050
##      4      4      4      1      4      3      4      4      4      4      4      4      4      2      1      1
## 8051 8055 8059 8061 8069 8072 8082 8083 8084 8103 8111 8114 8115 8118 8120 8126
##      4      4      1      4      1      4      4      3      4      1      4      4      1      3      1      1
## 8142 8145 8148 8152 8173 8175 8176 8181 8184 8200 8201 8205 8206 8214 8228 8230
##      4      4      3      1      1      1      4      3      1      4      1      1      1      1      1      3
## 8232 8236 8242 8243 8260 8261 8262 8272 8280 8284 8290 8292 8293 8301 8303 8311
##      1      3      1      1      3      1      1      4      1      1      1      3      1      1      3      4
## 8312 8313 8314 8324 8332 8334 8339 8342 8345 8346 8352 8360 8365 8382 8383 8395
##      4      1      1      1      2      3      3      1      2      4      4      4      1      3      3      1
## 8397 8400 8405 8406 8407 8411 8412 8413 8420 8422 8423 8428 8437 8441 8442 8448
##      3      1      3      1      1      1      4      1      1      1      1      1      1      1      3      1
## 8453 8455 8457 8458 8468 8489 8500 8503 8509 8522 8535 8536 8540 8565 8566 8577
##      4      4      4      1      4      4      1      1      4      4      1      1      1      4      4      1
## 8578 8583 8590 8604 8621 8622 8645 8647 8649 8650 8653 8660 8672 8673 8680 8691
##      4      4      4      4      4      1      4      1      4      3      4      1      4      1      1      4
## 8692 8693 8699 8702 8715 8735 8743 8745 8747 8750 8755 8766 8768 8774 8776 8783
##      4      4      4      4      3      1      4      1      1      4      1      4      1      4      4      4
## 8789 8791 8796 8805 8814 8815 8817 8819 8831 8838 8840 8842 8843 8856 8870 8879
##      1      4      1      4      1      4      4      1      4      4      4      4      1      4      4      1
## 8882 8886 8891 8893 8901 8907 8911 8915 8918 8921 8924 8934 8938 8965 8968 8972
##      4      1      4      2      1      4      1      4      4      4      3      4      1      4      1      4
## 8989 8991 8998 9002 9004 9008 9030 9044 9057 9076 9095 9112 9121 9125 9132 9133
##      4      4      4      4      1      1      4      1      1      4      4      4      4      4      4      4
##
## Within cluster sum of squares by cluster:
## [1] 119938239 327314347 120966510 97929284
## (between_SS / total_SS = 84.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

```

Ahora pasamos a graficarlo para ver los resultados, como hemos hecho anteriormente

```

fviz_cluster(object = kmeans4, data = data, show.clust.cent = TRUE,
              ellipse.type = "euclid", star.plot = TRUE, repel = TRUE) +
  labs(title = "Resultados clustering K-means") +
  theme_bw() +
  theme(legend.position = "none")

```

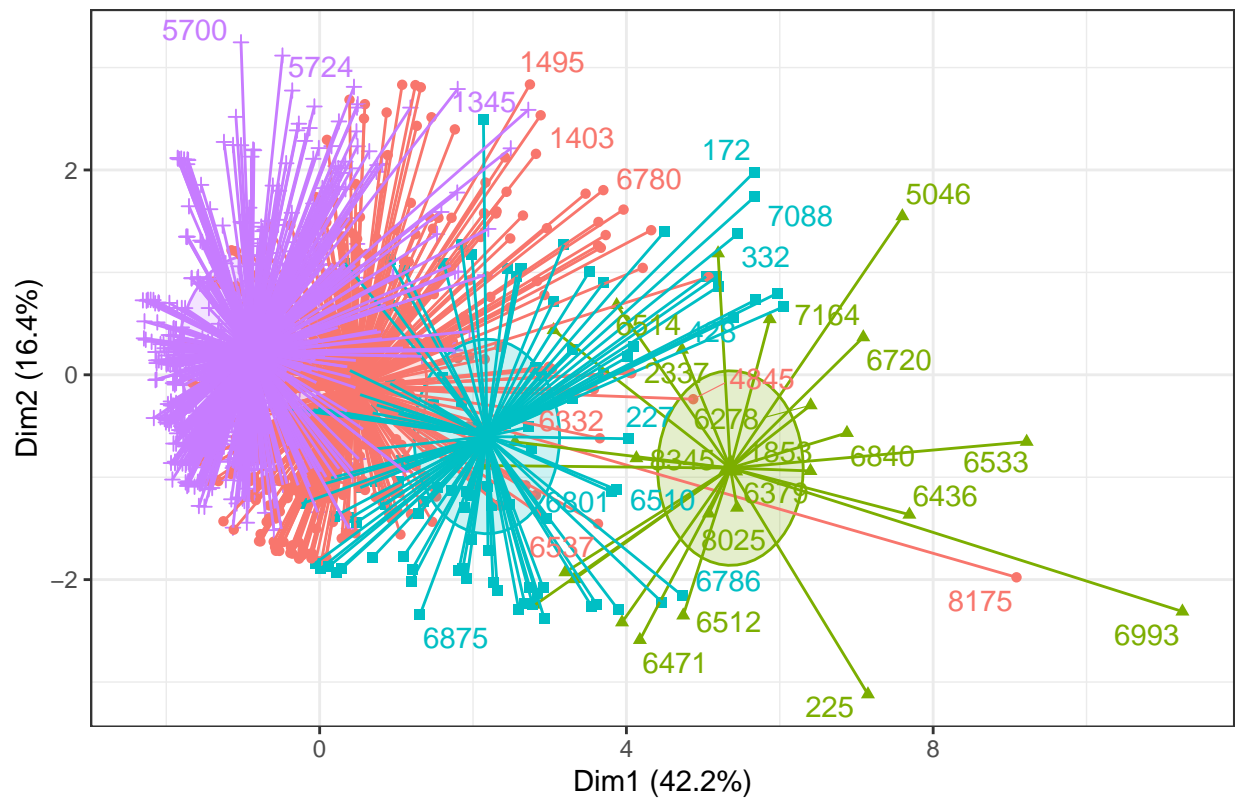
```

## Warning: ggrepel: 1132 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps

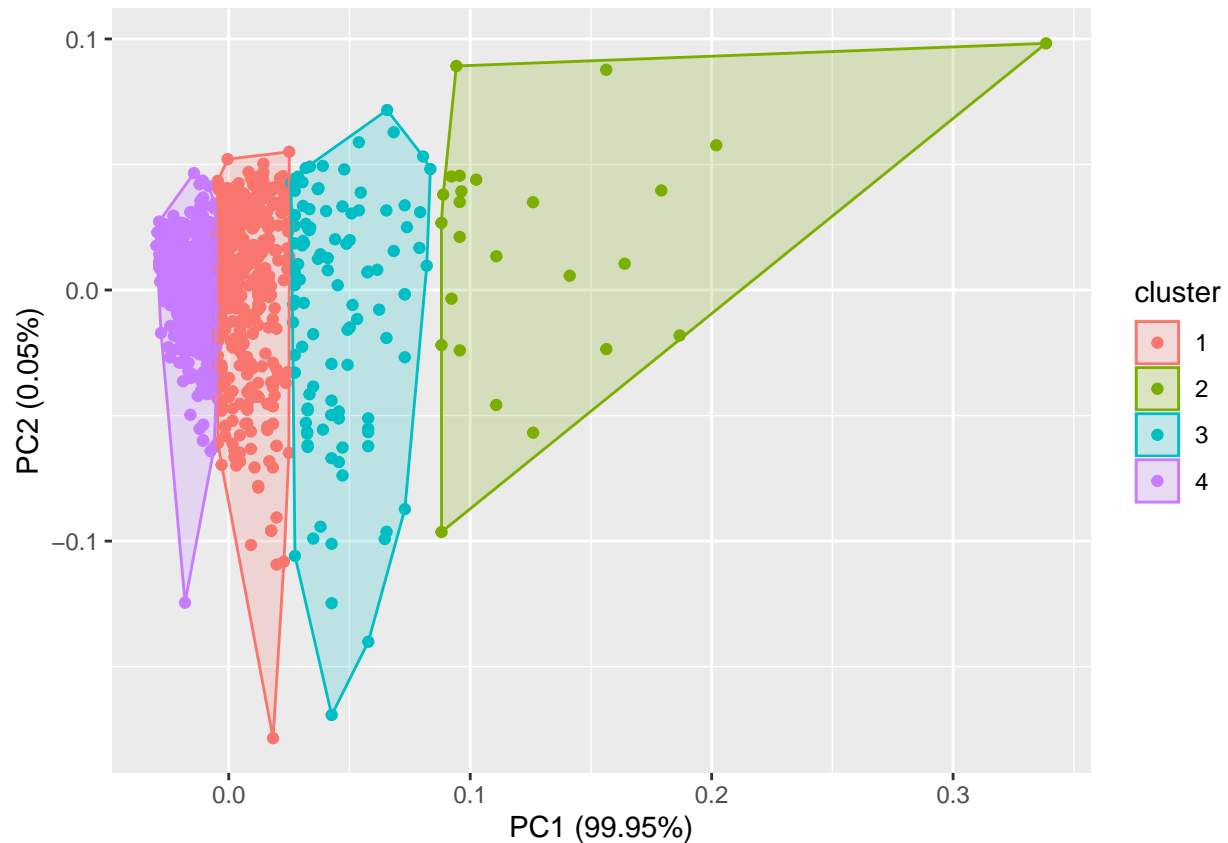
```



## Resultados clustering K-means



```
autoplot(kmeans4, data, frame=TRUE)
```



Obsevarmos que con K=4 tenemos todos los cluster con un gran numero de valores aunque al graficarlos vemos que están muy juntos por la zona media.

Pasamos el Clustering jerarquico

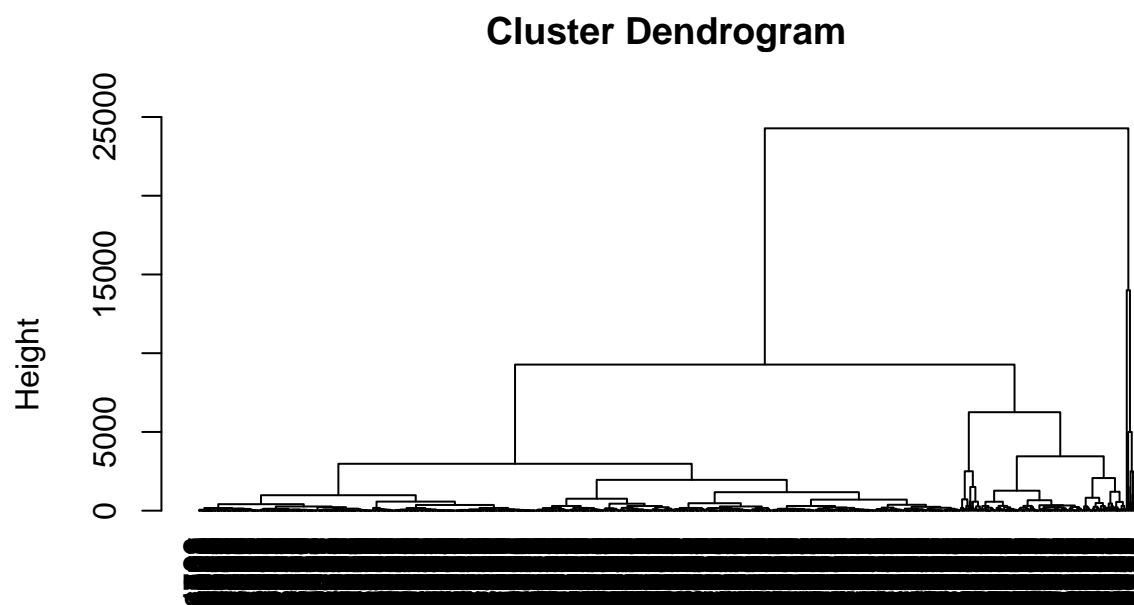
Construimos el dendograma

```
matriz_distancias <- dist(data)
den1 = hclust(matriz_distancias, method = "complete")
summary(den1)
```

```
##          Length Class  Mode
## merge      2334  -none- numeric
## height     1167  -none- numeric
## order      1168  -none- numeric
## labels     1168  -none- character
## method        1  -none- character
## call         3  -none- call
## dist.method   1  -none- character
```

Pasamos a visualizarlo

```
plot(den1, hang = -10)
```



```
matriz_distancias
hclust (*, "complete")
```

Decidimos cortar según el número de clusters

```
cortado_den1 <- cutree(den1, k=4)
```

Y comparamos con k-means

```
table(kmeans4$cluster, cortado_den1)
```

```
##      cortado_den1
##      1  2  3  4
## 1 321 80 0 0
## 2  0 15 9 1
## 3  0 112 0 0
## 4 630  0 0 0
```