# DS_Project

Anthéa von Borowski, Mengjiao li

2024-12-09

## Thema und Forschungsfrage

Nachhaltige Städte können nicht ohne städtische Wälder entwickelt werden. Stadtbäume, die Säulen der städtischen Wälder, spielen eine zentrale Rolle, indem sie unsere Gesundheit fördern, die Luft reinigen, Kohlendioxid speichern und die lokale Umgebung kühlen. Vor diesem Hintergrund haben wir uns für das Thema „Sozio-ökonomische Diskriminierung und Baumverteilung in US-Städten – am Beispiel New York City“ entschieden.

Unsere Forschungsfrage lautet: „Wie spiegeln sich sozio-ökonomische Diskriminierungen in der Baumverteilung in US-Städten wider?“

## Hypothesen

- Stadtviertel mit höherem Einkommen weisen eine höhere Baumdichte auf.
- Stadtviertel mit höherem Einkommen haben eine größere Anzahl exotischer Baumarten.
- Stadtviertel mit einem hohen Anteil nicht-weißer Bevölkerungsgruppen (z. B. Latino oder Afroamerikaner) haben eine geringere Baumdichte.
- Stadtviertel mit einem hohen Anteil nicht-weißer Bevölkerungsgruppen verfügen über weniger exotische Baumarten.

#Daten

Für die Untersuchung wurde ein umfassender und standardisierter Datensatz mit 5.660.237 Bäumen aus New York City zusammengestellt. Der Datensatz enthält detaillierte Informationen über:

- Baummerkmale: Standort, Baumart, Status der Einheimischen (einheimisch oder eingeführt), Gesundheitszustand, Größe.
- Umweltmerkmale: Unterscheidung zwischen Parks und urbanen Gebieten.

Zusätzlich wurden weitere Datensätze integriert, die sozio-ökonomische und demografische Merkmale abdecken:

- Einkommensdaten der Stadtviertel.
- Demografische Informationen wie ethnische Zusammensetzung.
- zur Stadtstruktur, z. B. Dichte von Wohn- und Geschäftsgebäuden.

Die zugrunde liegenden Informationen stammen aus Baumbestandsaufnahmen, die auf Stadt- und Stadtteilebene durchgeführt wurden.

#Methodik

Die Hypothesen werden mithilfe moderner Machine-Learning-Methoden überprüft:

- Supervised Learning: #Das ist kein Machine Learning
  - Regression: Zur Untersuchung der Beziehung zwischen Einkommen/Ethnizität und Baumdichte.
  - Logistische Regression: Für die Analyse kategorischer Merkmale wie „exotische vs. einheimische Baumarten“.
- Unsupervised Learning:
  - K-Means-Clustering: Zum Erkennen von Mustern und Clusterbildung in der Verteilung von Baumdichte und Baumarten auf Basis sozio-ökonomischer und ethnischer Variablen.

## Ressources

### Libraries

```
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("readr")
#install.packages("olsrr")
#install.packages("lmtest")
#install.packages("moments")
#install.packages("here")
#install.packages("Rtools")
library(dplyr)
```

```
## Warning: Paket 'dplyr' wurde unter R Version 4.4.2 erstellt
```

```
library(ggplot2)
library(readr)
```

```
## Warning: Paket 'readr' wurde unter R Version 4.4.2 erstellt
```

```
library(lmtest)
```

```
## Warning: Paket 'lmtest' wurde unter R Version 4.4.2 erstellt

## Warning: Paket 'zoo' wurde unter R Version 4.4.2 erstellt
```

```
library(olsrr)
```

```
## Warning: Paket 'olsrr' wurde unter R Version 4.4.2 erstellt
```

```
library(moments)
library(here)
```

```
## Warning: Paket 'here' wurde unter R Version 4.4.2 erstellt
```

### Data Import

Data has been previously selected and modified in “data_import.Rmd”.

```
baumnyc<-read.csv(here("Data", "Refined", "baumnyc2.csv"))
baumnyc_gp<-read.csv(here("Data", "Refined", "baumnyc_gp2.csv"))
zbp<-read.csv(here("Data", "Raw","zbp.csv"))
baumnyc$X<-NULL
baumnyc_gp$X<-NULL
```

### Multi-lineare Regression

**Bäumedensität** H1.1 - Viertel mit höheren Einkommen haben mehr Baümedensität. H2.1 - Viertel mit nicht-weissen/(Latino- Afroam?) Ethnien haben weniger Baümedensität.

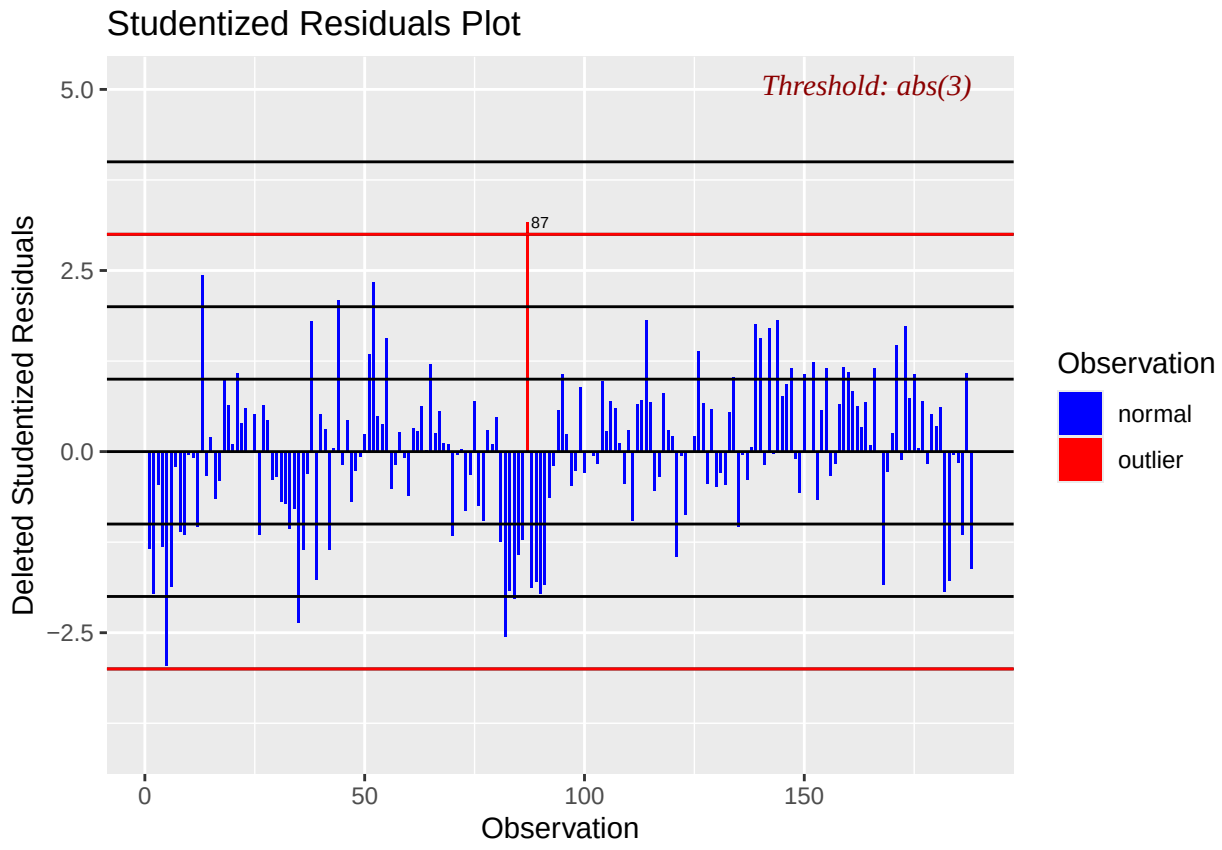
```
requiredh11<-c( "ZCTA", "mean_income", "eth_white_nonhisp", "eth_afroam_nonhisp", "eth_asian_nonhisp",


df_h11<-na.omit(baumnyc_gp[,requiredh11])
df_h11$income_category<-ifelse(df_h11$mean_income<=quantile(df_h11$mean_income, 0.25), 1,
                               ifelse(df_h11$mean_income<=quantile(df_h11$mean_income, 0.5), 2,
                                      ifelse(df_h11$mean_income<=quantile(df_h11$mean_income, 0.75), 3,
#df_h11_log<-log(df_h11[-188,])
#df_h11_log<-df_h11_log[-39,]
#index(df_h11$ZCTA == 11005,)

regh11<-lm(tree_density~mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + eth_hisp + eth_other + p
summary(regh11)
```

```
##
## Call:
## lm(formula = tree_density ~ mean_income + eth_afroam_nonhisp +
##     eth_asian_nonhisp + eth_hisp + eth_other + pop_density, data = df_h11)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -940.38 -170.12    5.43  207.78 1015.75
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.025e+02  1.310e+02   5.362 2.49e-07 ***
## mean_income        -2.123e-03  9.670e-04  -2.196   0.0294 *
## eth_afroam_nonhisp  4.097e+01  1.346e+02   0.304   0.7611
## eth_asian_nonhisp   2.834e+02  2.310e+02   1.227   0.2215
## eth_hisp           -3.117e+02  1.674e+02  -1.862   0.0642 .
## eth_other           1.286e+03  8.050e+02   1.598   0.1119
## pop_density         2.539e-02  2.029e-03  12.510  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 332.7 on 181 degrees of freedom
## Multiple R-squared:  0.4693, Adjusted R-squared:  0.4517
## F-statistic: 26.67 on 6 and 181 DF,  p-value: < 2.2e-16
```

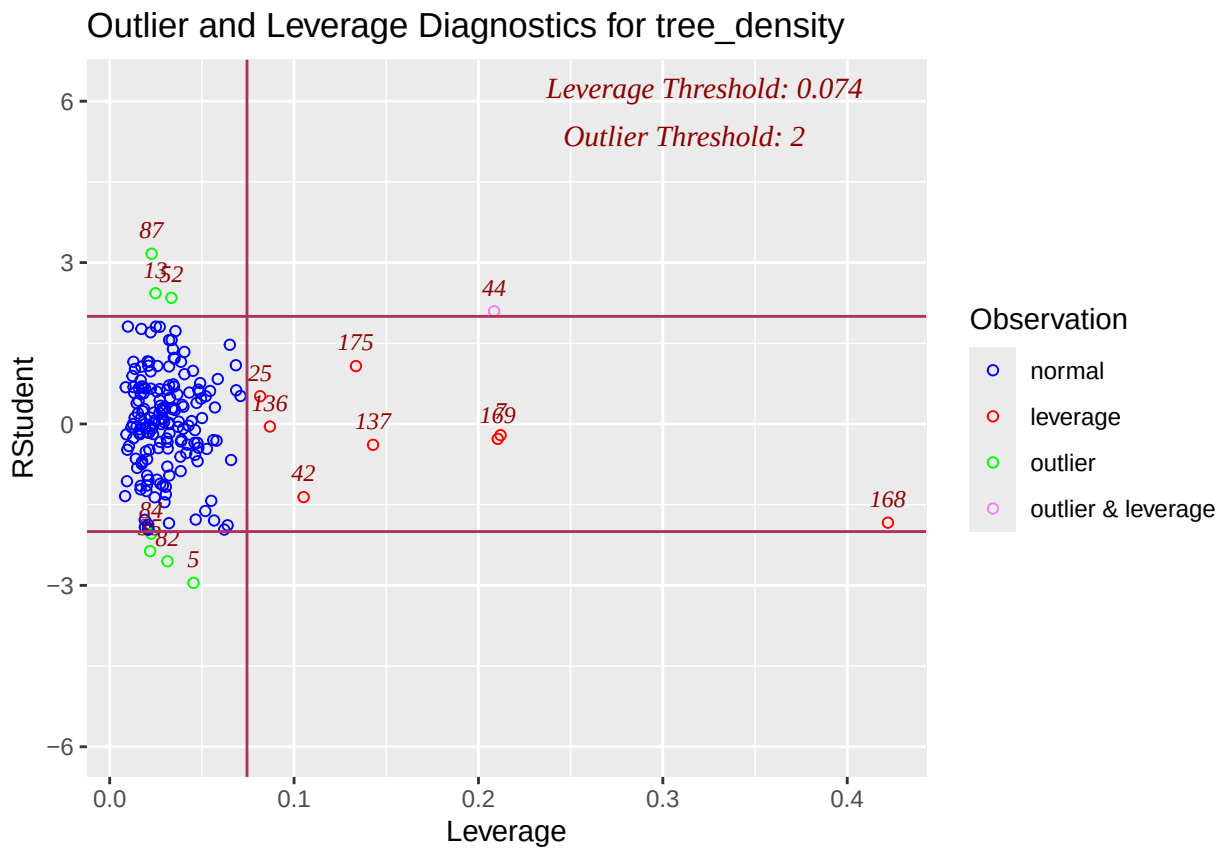
```r
summary(df_h11$tree_density)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##     0.1337  730.1606 1030.5843  987.1496 1275.7784 2126.2264
```

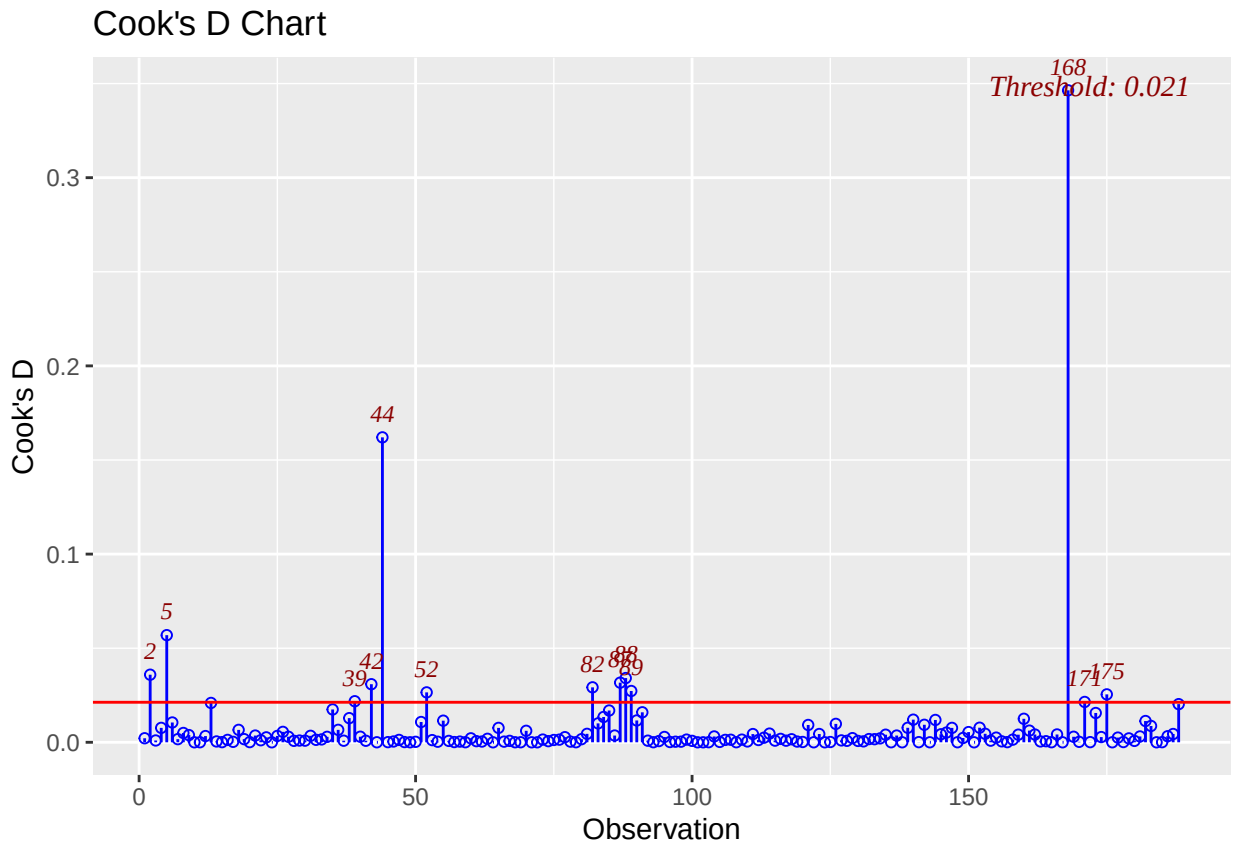
```r
#OUTLIERS

ols_plot_resid_stud(regh11) #no values out of bounds
```



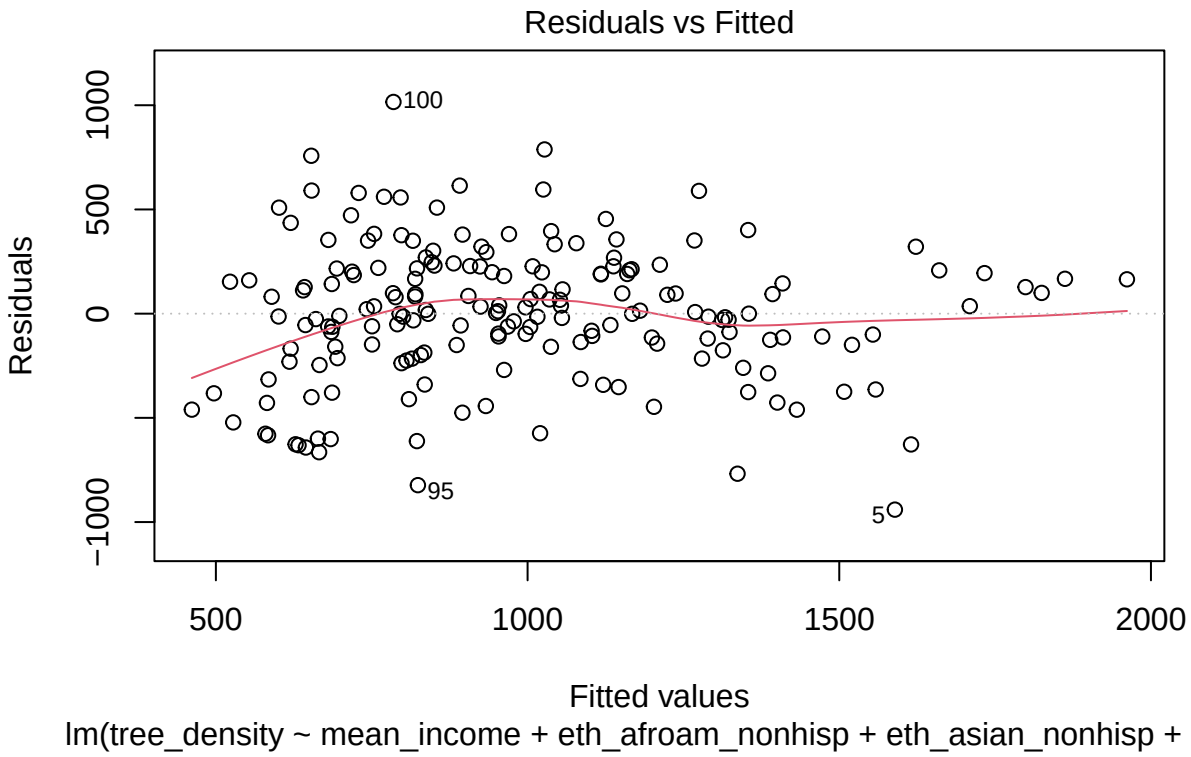

```r
ols_plot_resid_lev(regh11)
```

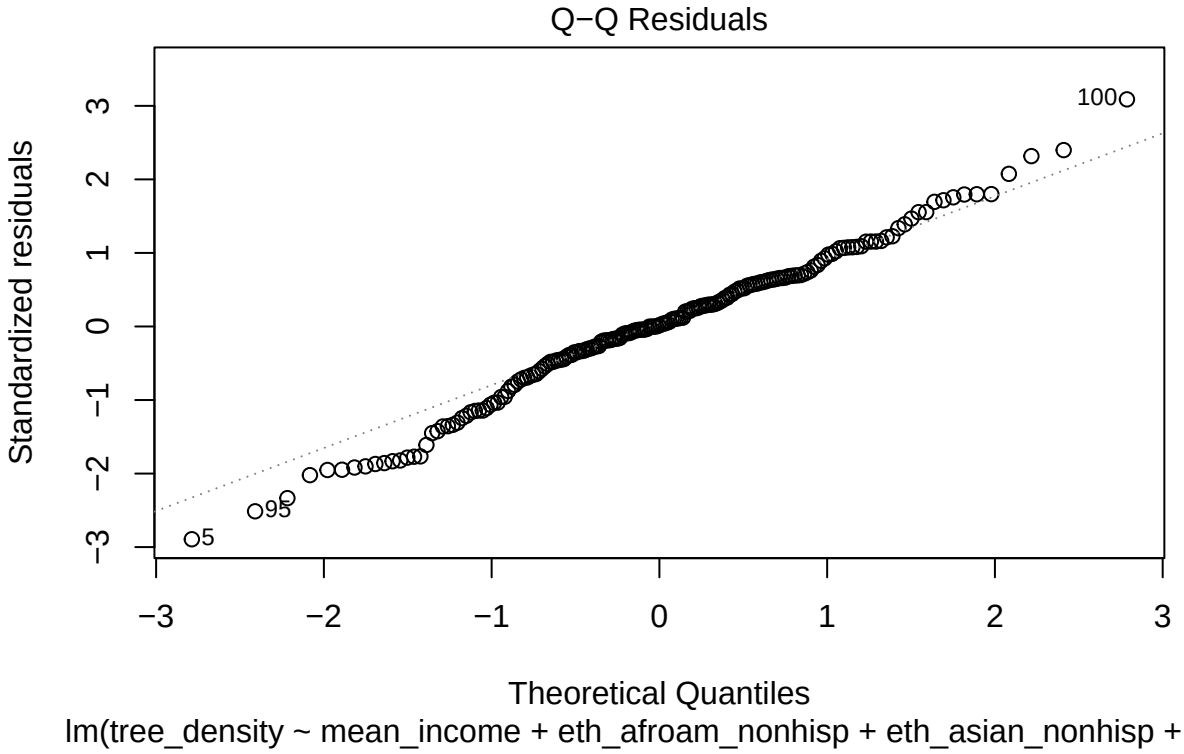


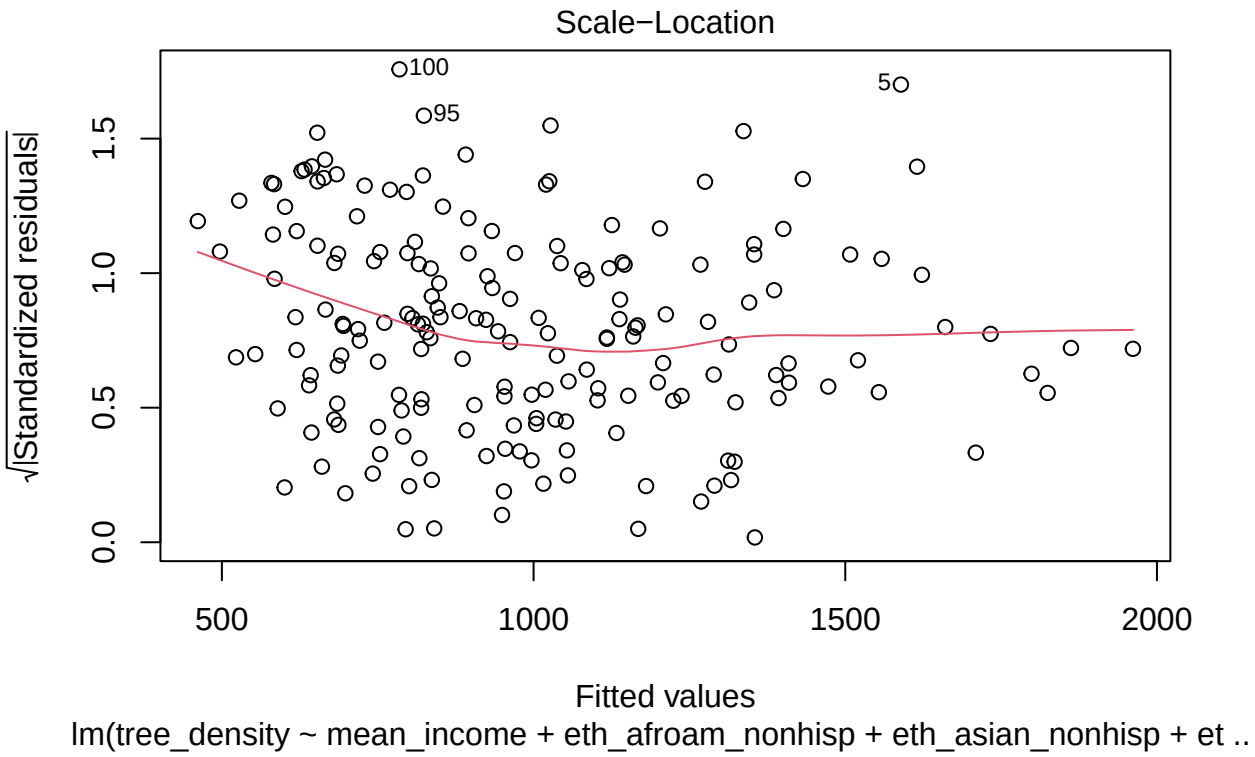

```
ols_plot_cooksd_chart(regh11)
```

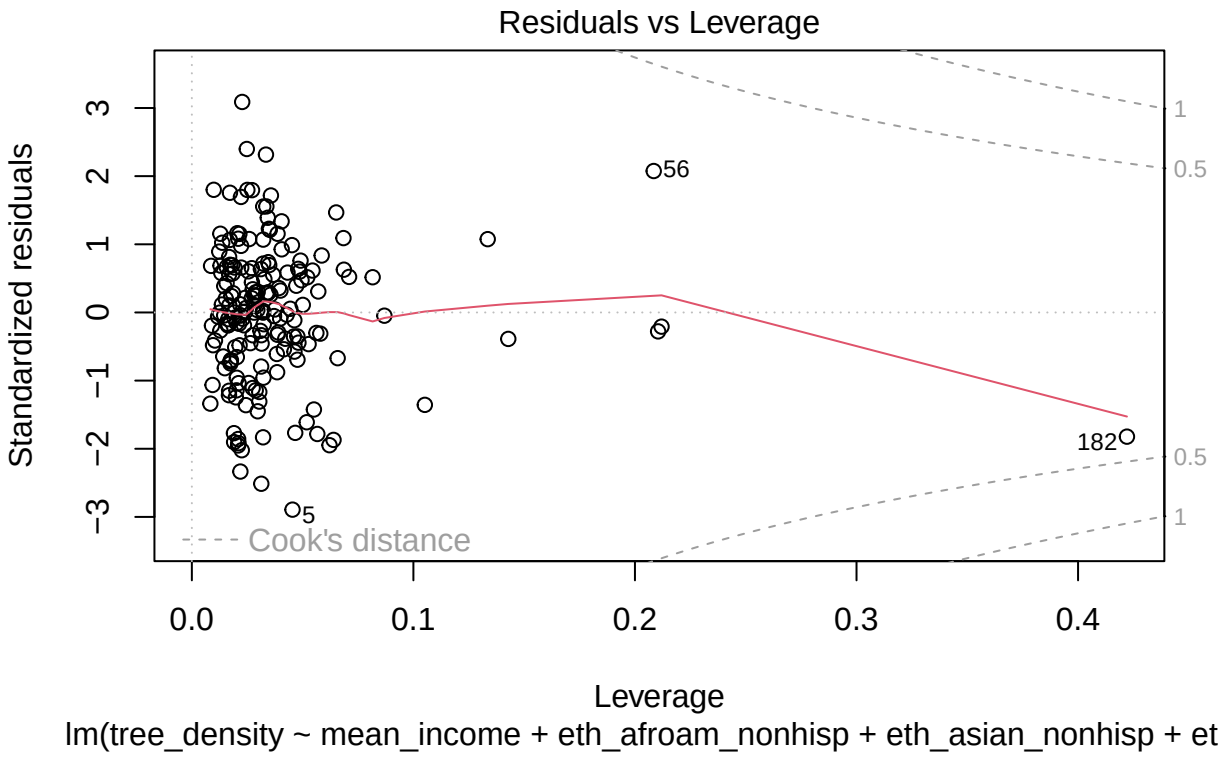



```
df_h11_no<-df_h11[-168,]
df_h11_no<-df_h11_no[-33,]

#LINEARITY - erfüllt

plot(regh11)
```

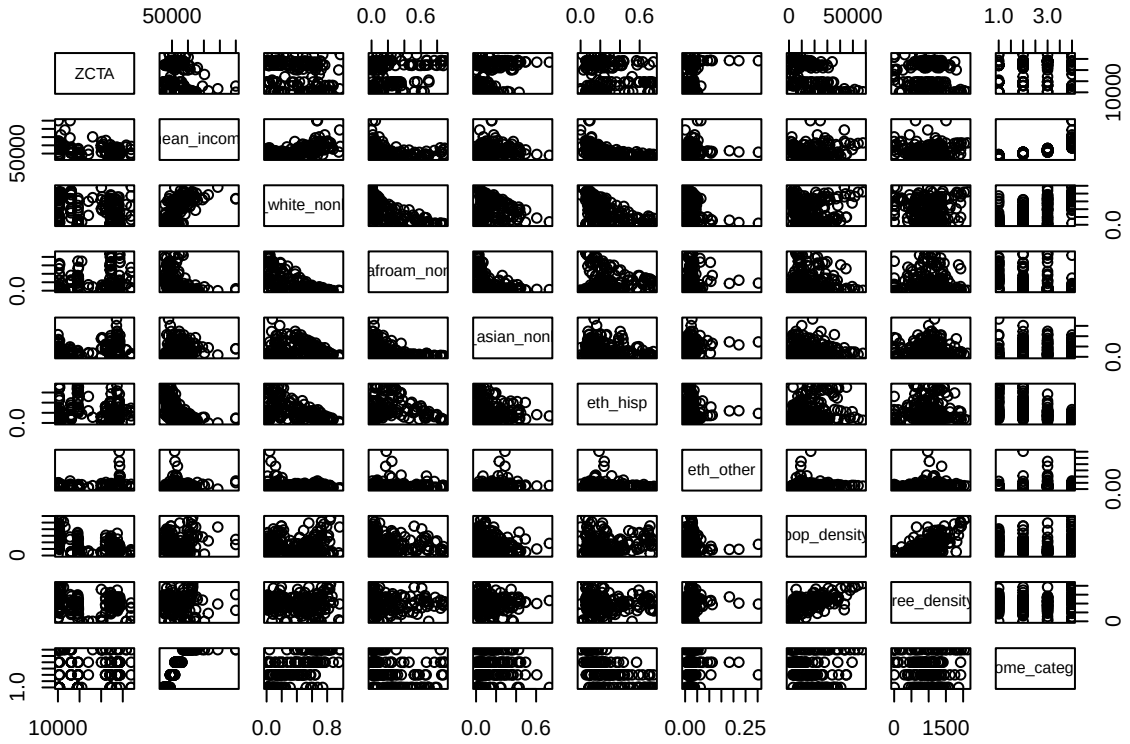

Residuals vs Fitted
Residuals
1000
500
0
−1000
100
95
5
500
1000
1500
2000
Fitted values
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..
100
95
5
−3
−2
−1
0
1
2
3


Scale−Location
100
95
5
√|Standardized residuals|
0.0
0.5
1.0
1.5
500
1000
1500
2000
Fitted values
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..


Residuals vs Leverage
Standardized residuals
3
2
1
0
−1
−2
−3
0.0
0.1
0.2
0.3
0.4
Leverage
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..
56
182
5
Cook's distance
1
0.5
0.5
1

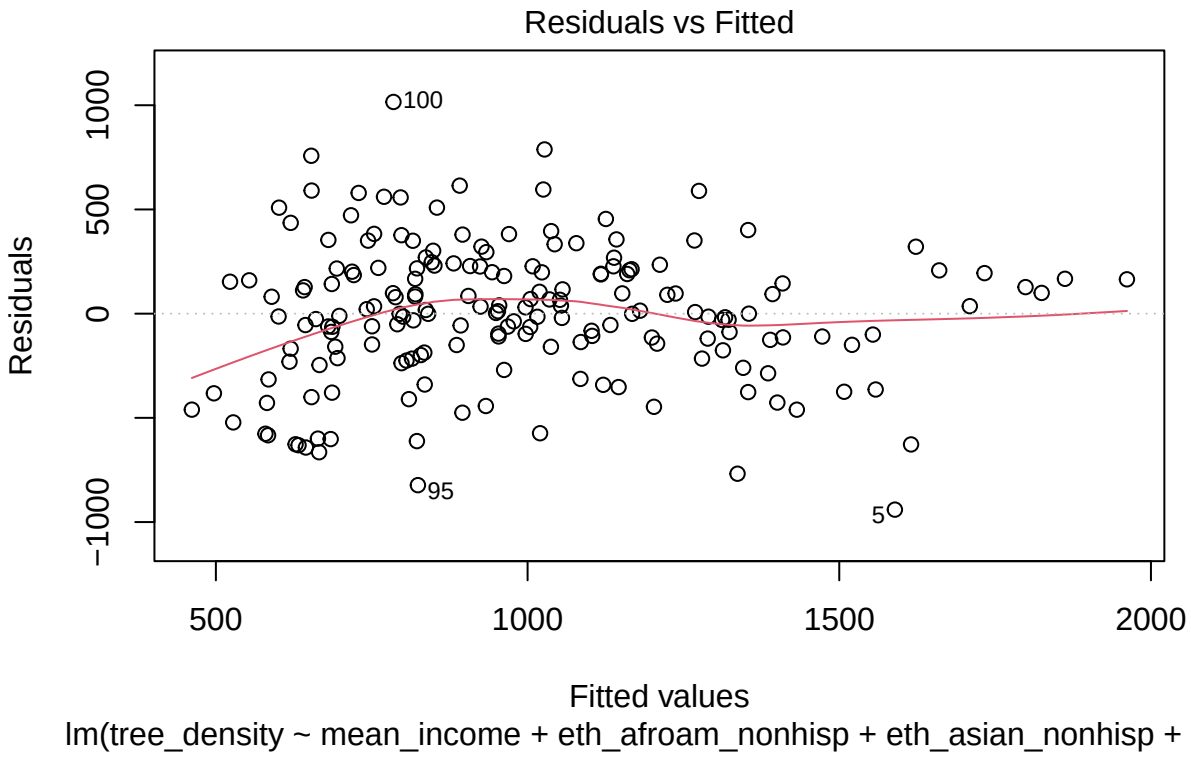
```
pairs(df_h11)
```

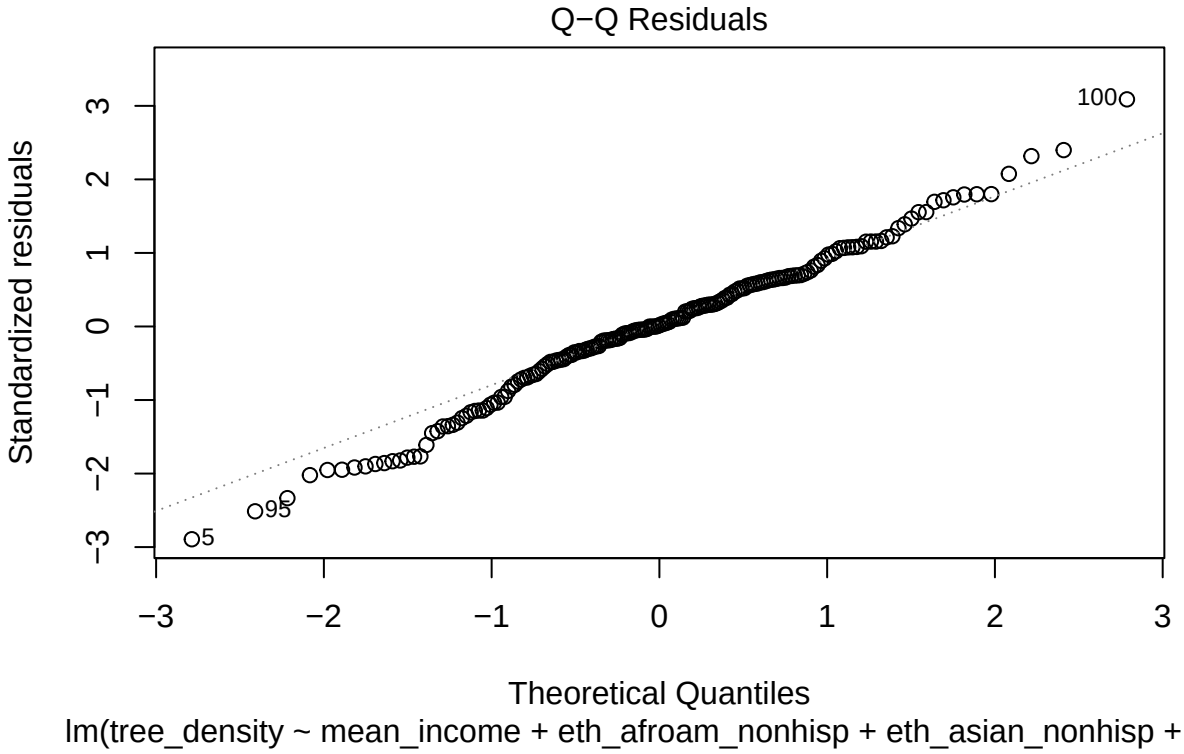


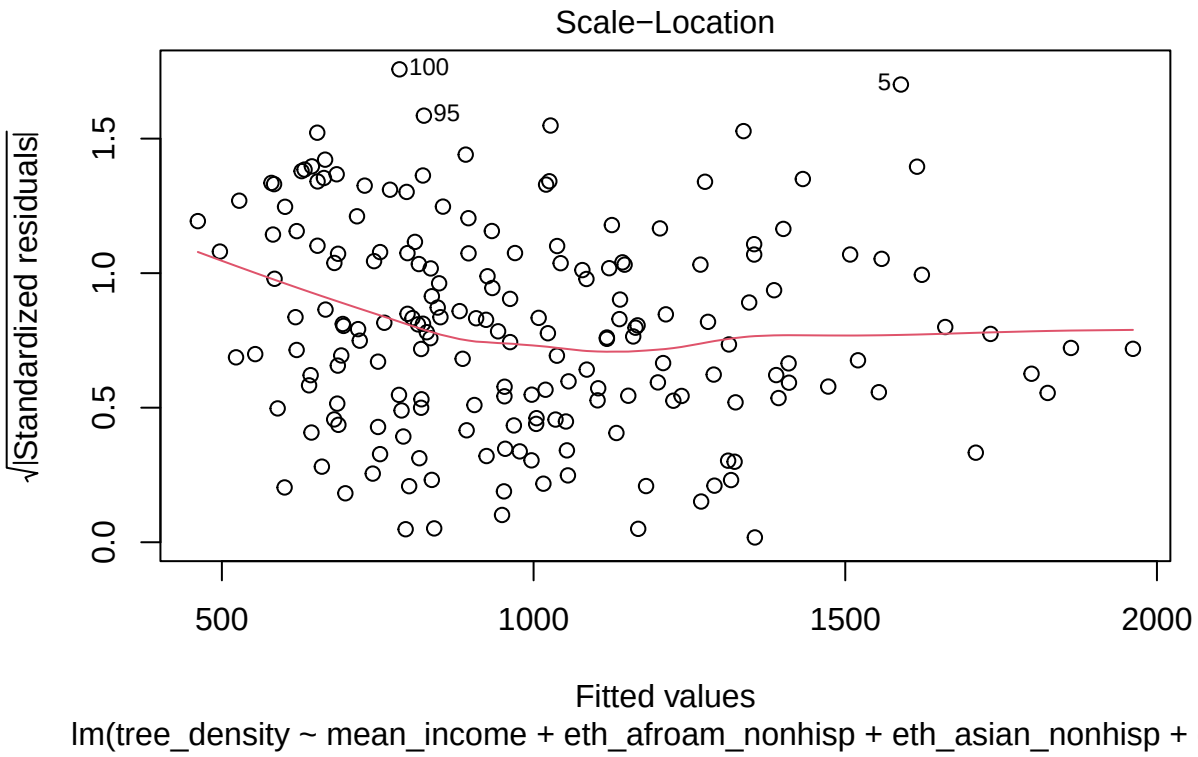

```
raintest(regh11)
```

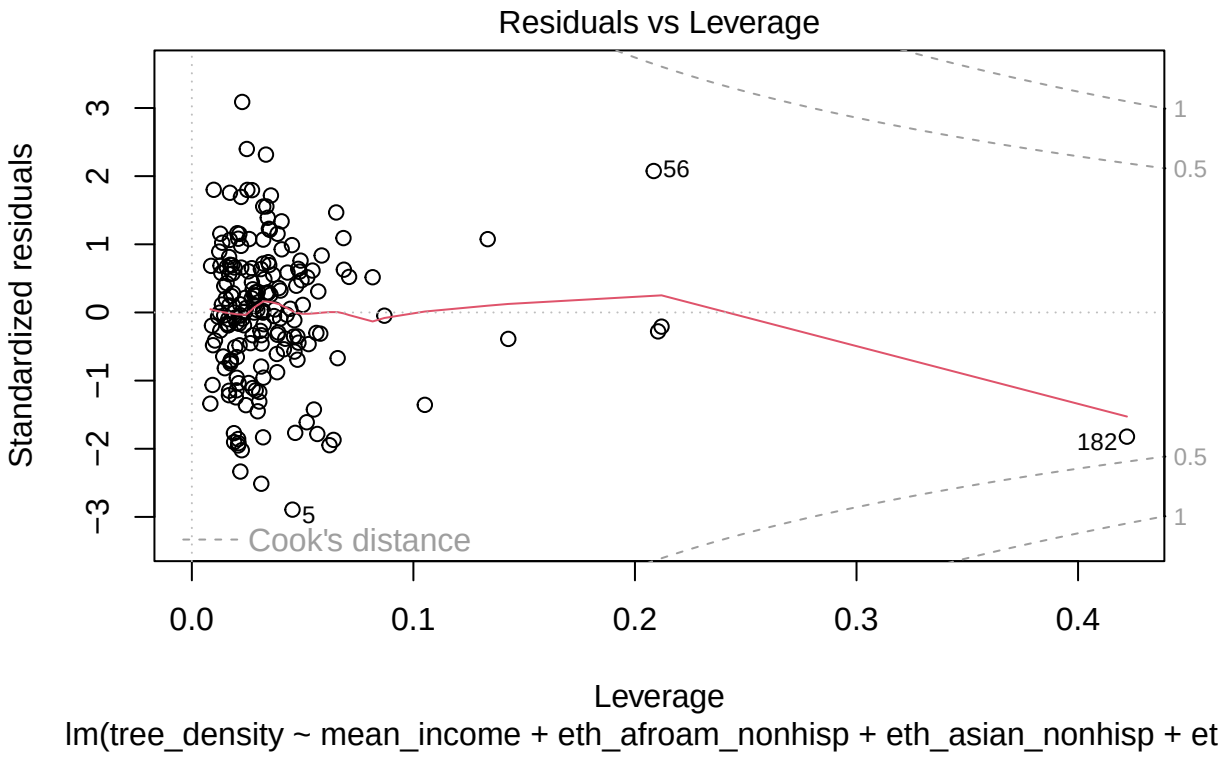
```
##
##  Rainbow test
##
## data:  regh11
## Rain = 1.2405, df1 = 94, df2 = 87, p-value = 0.1547
```

```
#MULTI-COLINEARITY NONE

#HETEROSKEDASTICITY

plot(regh11)
```

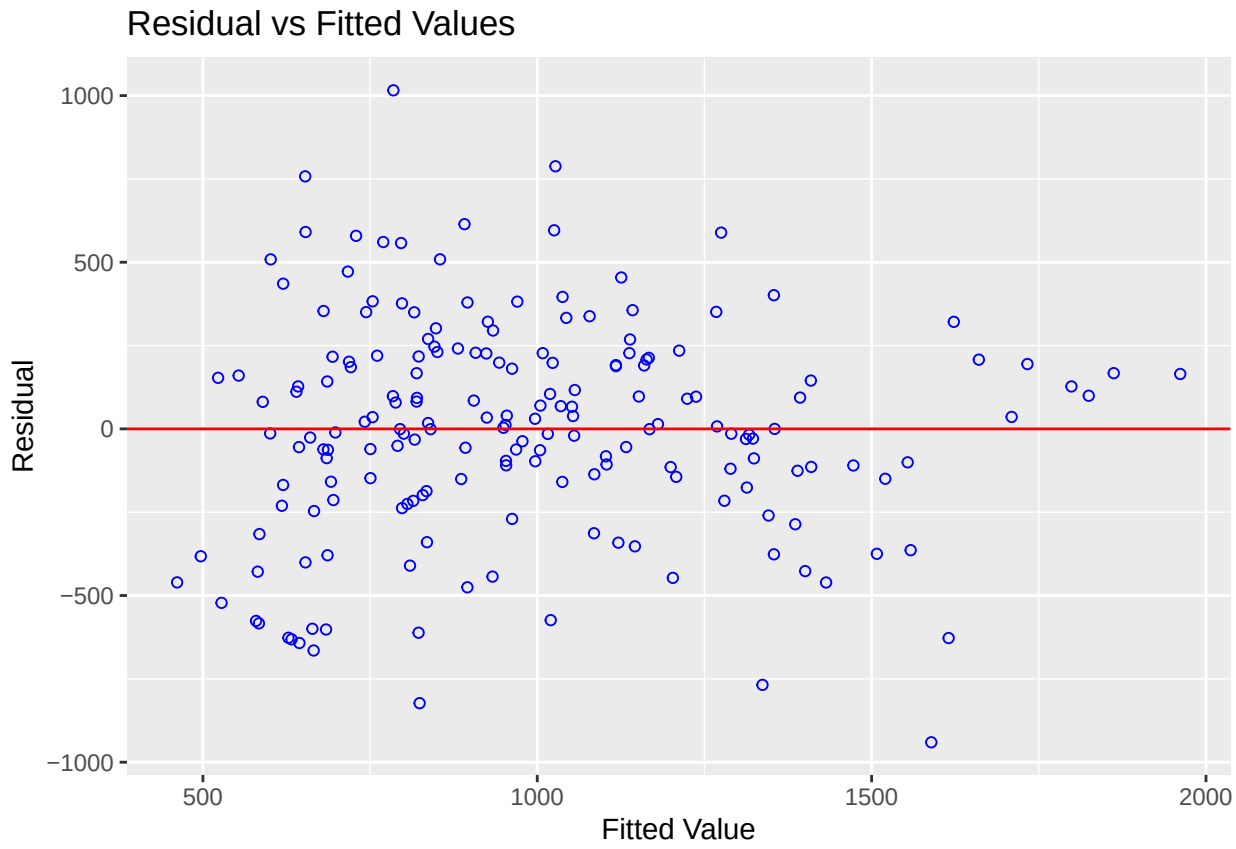

Residuals vs Fitted
Residuals
1000
500
0
−1000
100
95
5
500
1000
1500
2000
Fitted values
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..
100
95
5
−3
−2
−1
0
1
2
3


Scale−Location
100
95
5
√|Standardized residuals|
0.0
0.5
1.0
1.5
500
1000
1500
2000
Fitted values
lm(tree_density ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + et ..



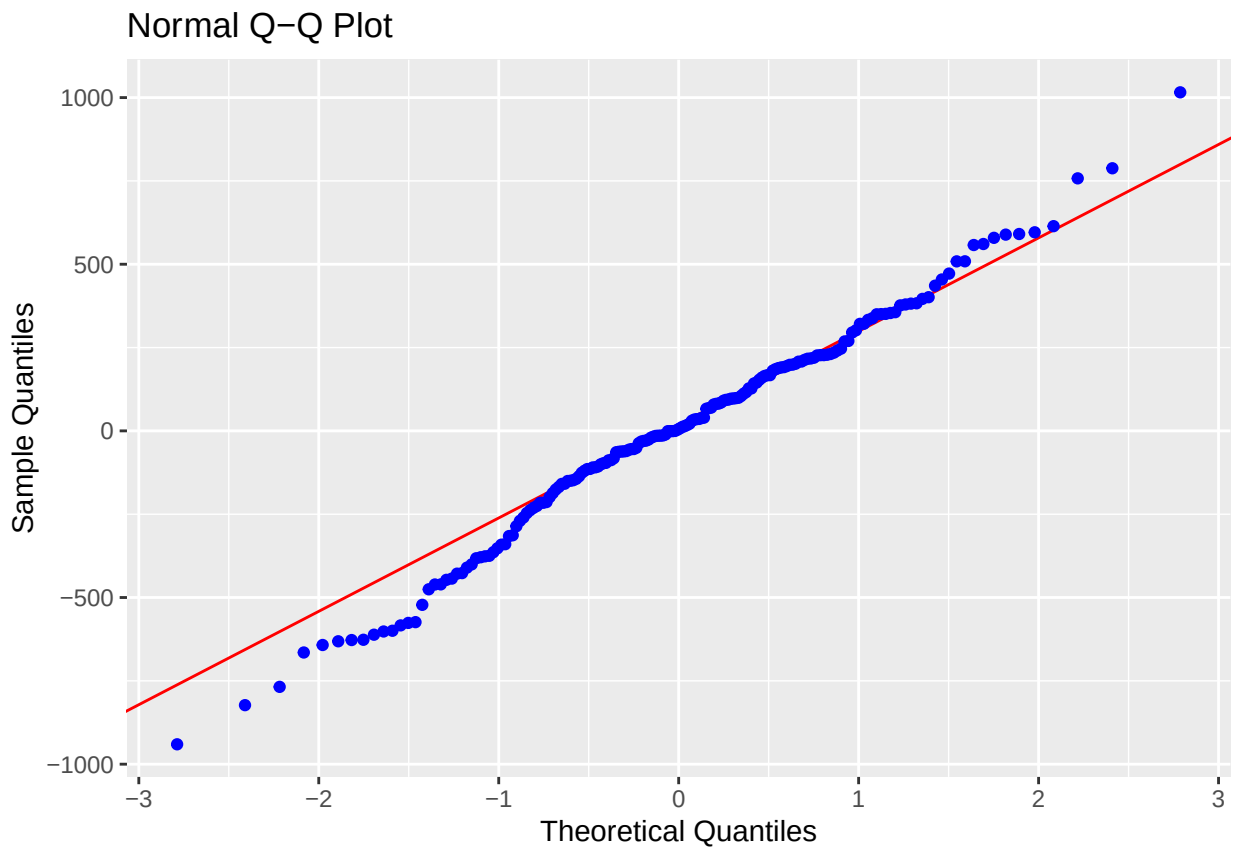

```
ols_plot_resid_fit(regh11)
```




```
ols_test_breusch_pagan(regh11)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##                  Data
##  -----------------------------------------
##  Response : tree_density
##  Variables: fitted values of tree_density
##
##         Test Summary
##  ----------------------------
##  DF            =    1
##  Chi2          =    3.305512
##  Prob > Chi2   =    0.06904783
```

```
#NORMALITY
ols_plot_resid_qq(regh11)
```




```
shapiro.test(regh11$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  regh11$residuals
## W = 0.99068, p-value = 0.2633
```

```
kurtosis(regh11$residuals)
```

```
## [1] 3.268726
```

```
skewness(regh11$residuals)
```

```
## [1] -0.1489246
```

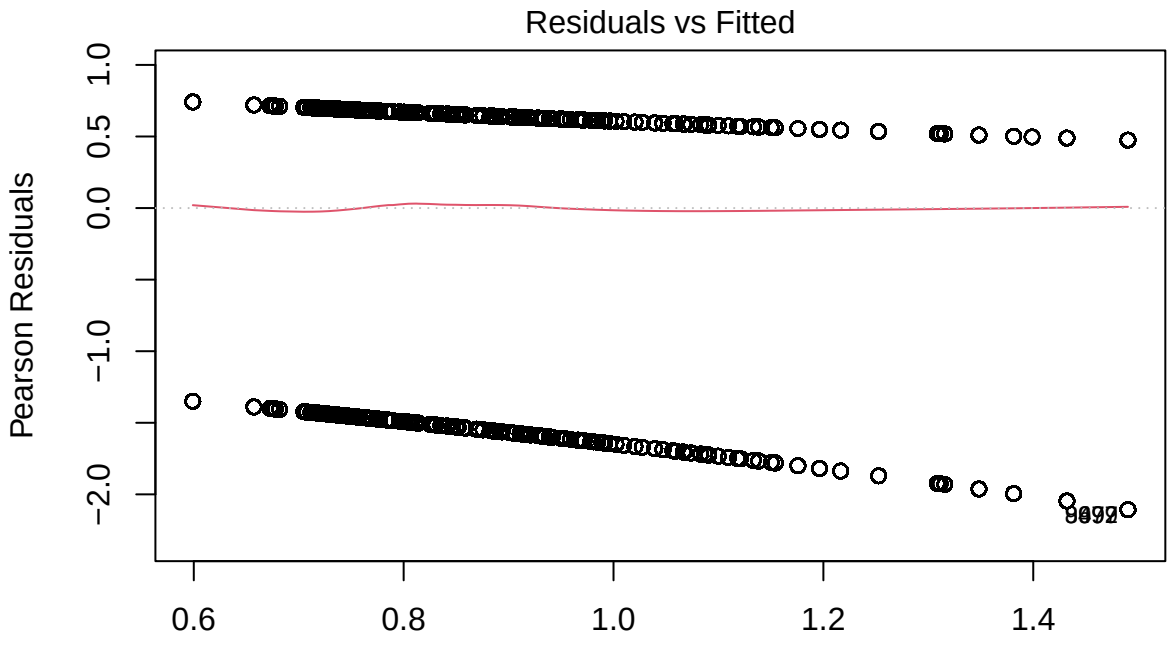
```
agostino.test(regh11$residuals)
```

```
##
##  D'Agostino skewness test
##
## data:  regh11$residuals
## skew = -0.14892, z = -0.85988, p-value = 0.3899
## alternative hypothesis: data have a skewness
```

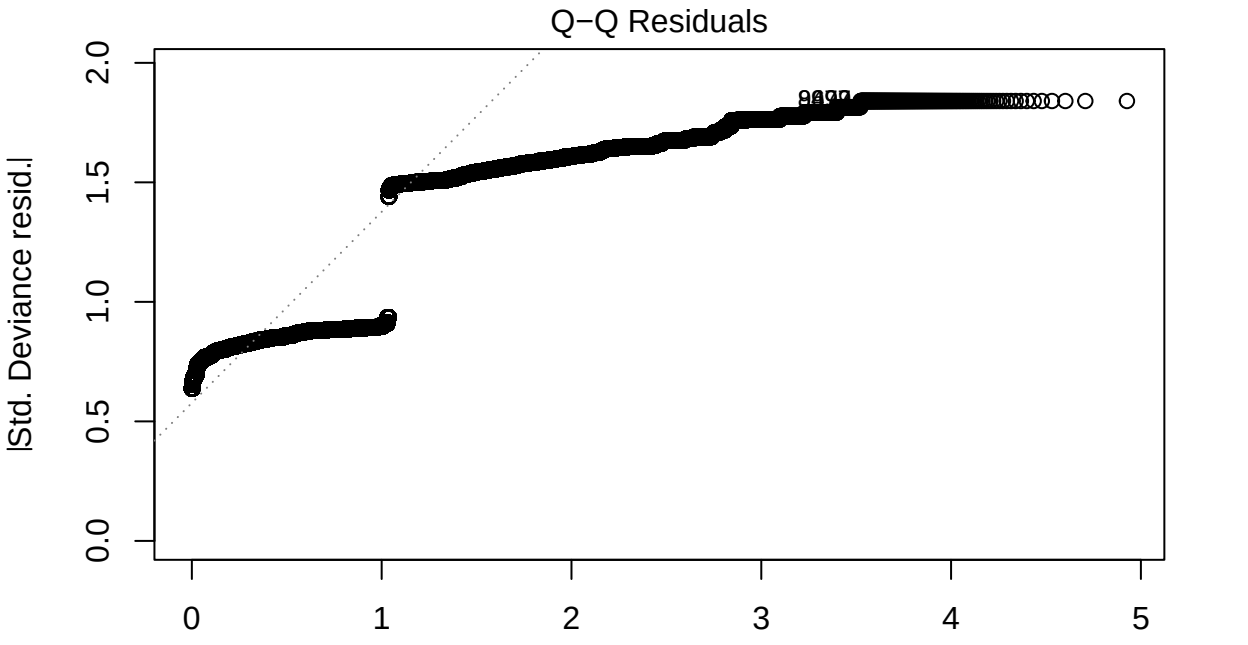
```r
anscombe.test(regh11$residuals)
```

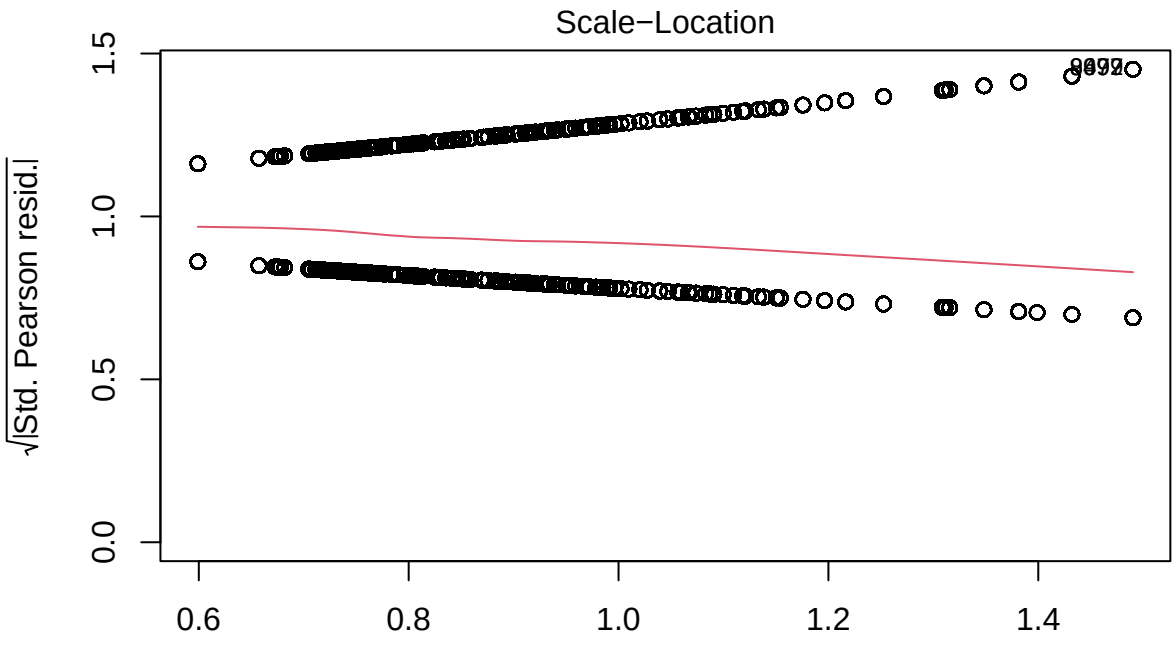
```
##
##  Anscombe-Glynn kurtosis test
##
## data:  regh11$residuals
## kurt = 3.26873, z = 0.95162, p-value = 0.3413
## alternative hypothesis: kurtosis is not equal to 3
```

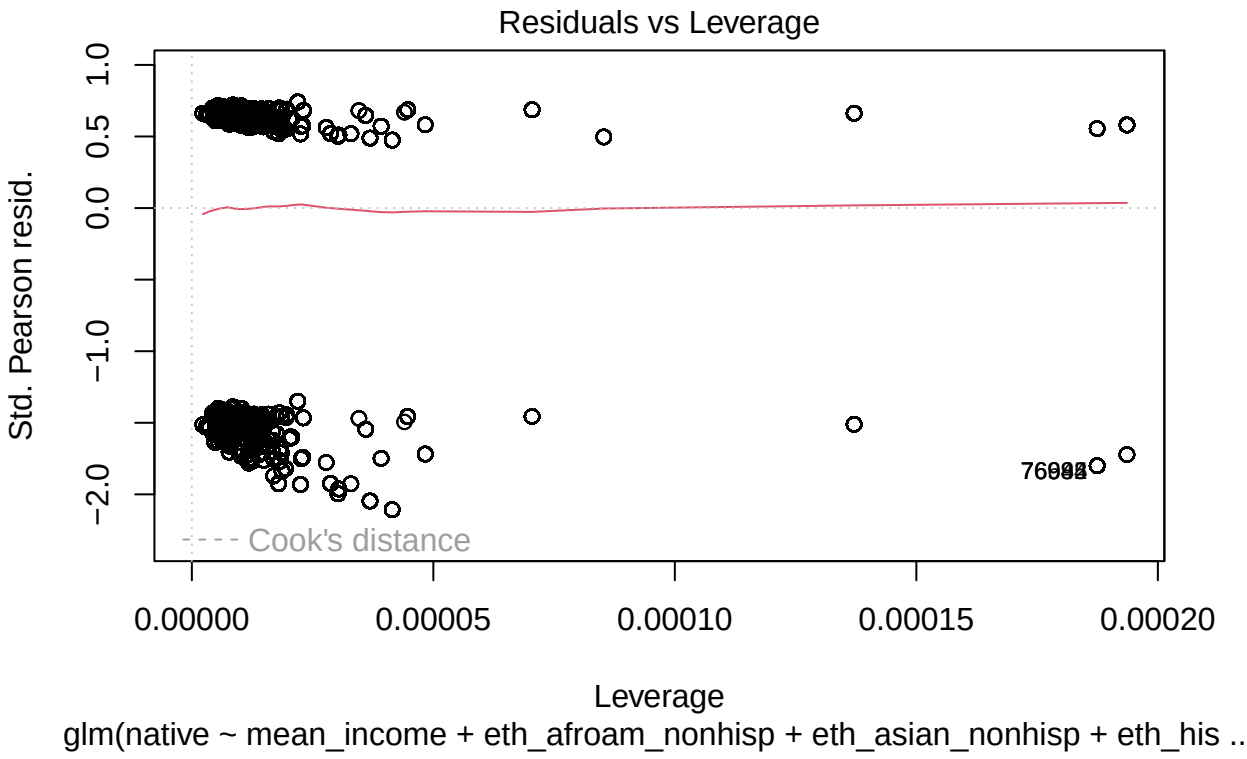
```r
requiredh12<-c( "ZCTA5CE10", "mean_income", "eth_white_nonhisp", "eth_afroam_nonhisp", "eth_asian_nonhis


df_h12<-na.omit(baumnyc[,requiredh12])
df_h12$income_category<-ifelse(df_h12$mean_income<=quantile(df_h12$mean_income, 0.25), 1,
                               ifelse(df_h12$mean_income<=quantile(df_h12$mean_income, 0.5), 2,
                                      ifelse(df_h12$mean_income<=quantile(df_h12$mean_income, 0.75), 3,
#df_h12_log<-log(df_h12[-188,])
#df_h12_log<-df_h11_log[-39,]
#index(df_h11$ZCTA == 11005,)

regh12<-glm(native~mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + eth_hisp + eth_other + pop_de

plot(regh12)
```


Residuals vs Fitted
Pearson Residuals
1.0
0.5
0.0
−1.0
−2.0
0.6
0.8
1.0
1.2
1.4
Predicted values
glm(native ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + eth_




Q−Q Residuals
|Std. Deviance resid.|
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
Theoretical Quantiles
glm(native ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + eth_his ..


Scale−Location
√|Std. Pearson resid.|
1.5
1.0
0.5
0.0
0.6
0.8
1.0
1.2
1.4
Predicted values
glm(native ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp + eth_his ..




```
summary(regh12)
```

```
##
## Call:
## glm(formula = native ~ mean_income + eth_afroam_nonhisp + eth_asian_nonhisp +
##     eth_hisp + eth_other + pop_density, family = "binomial",
##     data = df_h12)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          6.270e-01  1.929e-02  32.510  < 2e-16 ***
## mean_income          1.061e-06  1.867e-07   5.683 1.32e-08 ***
## eth_afroam_nonhisp   1.965e-01  1.475e-02  13.326  < 2e-16 ***
## eth_asian_nonhisp   -9.376e-03  2.472e-02  -0.379   0.7045
## eth_hisp            -2.147e-01  2.234e-02  -9.610  < 2e-16 ***
## eth_other           -2.578e-01  9.400e-02  -2.742   0.0061 **
## pop_density          1.310e-05  2.974e-07  44.037  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 729389  on 597296  degrees of freedom
## Residual deviance: 726993  on 597290  degrees of freedom
## AIC: 727007
```

```
##
## Number of Fisher Scoring iterations: 4
```

```
summary(df_h11$tree_density)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.1337  730.1606 1030.5843  987.1496 1275.7784 2126.2264
```

```
library(ggplot2)
library(factoextra)
```

```
## Warning: Paket 'factoextra' wurde unter R Version 4.4.2 erstellt
```

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

```
library(FactoMineR)
```

```
## Warning: Paket 'FactoMineR' wurde unter R Version 4.4.2 erstellt
```

```
data_cluster<-read.csv(here("Data","Refined","Baumnyc_gp2.csv"))
names(data_cluster)
```

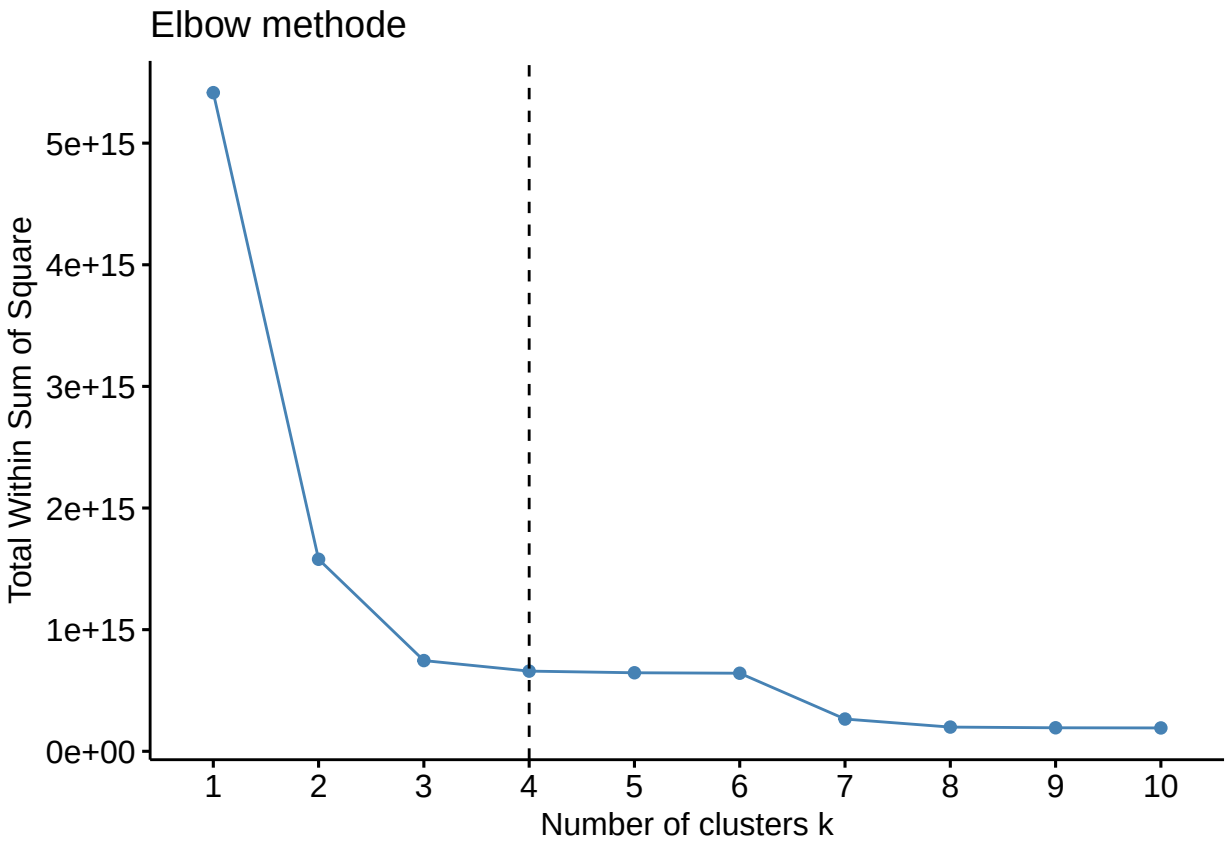
```
##  [1] "X"                  "ZCTA"               "mean_native"
##  [4] "mean_condition"     "mean_income"        "pop"
##  [7] "eth_white_nonhisp"  "eth_afroam_nonhisp" "eth_asian_nonhisp"
## [10] "eth_hisp"           "eth_other"          "count_tree"
## [13] "surface"            "pop_density"        "tree_density"
## [16] "housing_ttl"        "housing_own"        "housing_rent"
## [19] "housing_own_pct"    "housing_density"    "housing_pop"
## [22] "ESTAB"              "ESTAB_density"      "ESTAB_pop"
## [25] "ESTAB_housing"      "PAYANN"             "PAYANN_density"
## [28] "PAYANN_pop"
```

```
data_cluster$X<-NULL
rownames(data_cluster)<-data_cluster$ZCTA
for (col in names(data_cluster)) {
  if (!is.numeric(data_cluster[[col]])) {
    data_cluster[[col]] <- as.numeric(df[[col]])
  }
}
data_cluster <- na.omit(data_cluster)
df<-scale(data_cluster)

head(df)
```

```
##            ZCTA mean_native mean_condition mean_income        pop
## 10001 -1.448072   0.6796889      0.1978603   0.5514871 -0.8458835
## 10002 -1.446308   0.5096734     -0.2959430  -0.9500248  1.2881304
## 10003 -1.444545   1.3708390     -0.3459322   0.8725829  0.4067523
## 10004 -1.442781   1.9476870      0.2038518   1.4819577 -1.6038448
```

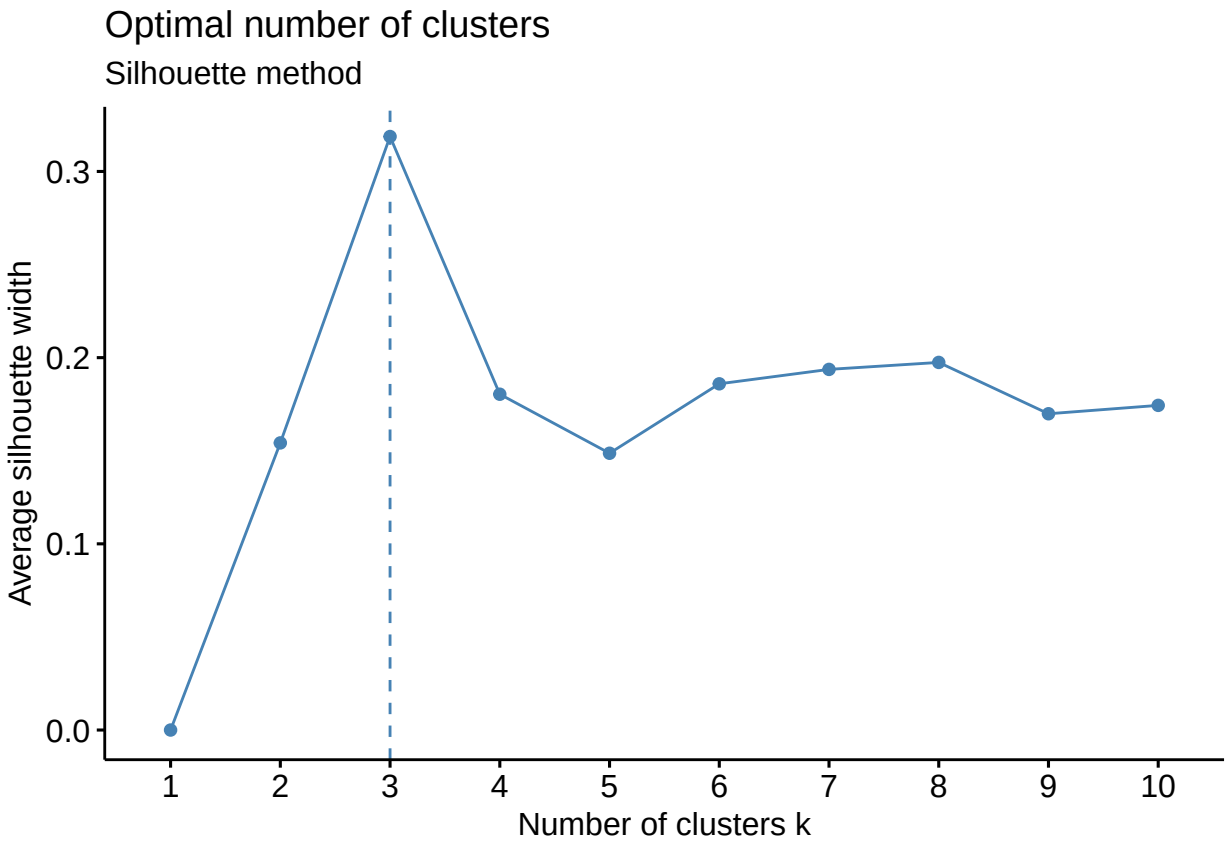
```
## 10005 -1.441018   1.2975746     -2.1400805   1.6093493 -1.4206596
## 10006 -1.439254   2.7026563      0.5400683   1.5951193 -1.6079860
##       eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp    eth_hisp
## 10001         0.4885354         -0.3508163        0.39556646 -0.48904068
## 10002        -0.6556779         -0.5101923        2.18923425  0.05524356
## 10003         1.2444335         -0.6526504       -0.02619475 -0.92293144
## 10004         0.9754581         -0.8154226        0.95408609 -0.95064298
## 10005         1.2170340         -0.7008395        0.28060136 -0.97215195
## 10006         1.1131489         -0.7073456        0.28460956 -0.83995422
##         eth_other count_tree    surface pop_density tree_density housing_ttl
## 10001 -0.21699606 -0.9599983 -0.6648012  -0.1438559 -1.106823946  -0.6074714
## 10002  0.04144264 -0.4648338 -0.4991691   1.5330950  0.000985157   1.6462012
## 10003 -0.03857446 -0.5327247 -0.6888061   1.7584590  0.853806006   1.1206499
## 10004 -0.40828614 -1.1494848 -0.7062093  -1.1584095 -1.855376381  -1.6182037
## 10005 -0.41722929 -1.1819103 -1.0048067   2.1587608 -0.752975964  -1.3525821
## 10006 -0.29864027 -1.2062294 -0.9929303  -0.2797960 -1.727645601  -1.6070308
##       housing_own housing_rent housing_own_pct housing_density housing_pop
## 10001  -0.6336907   -0.3989705      -0.4297716      0.05215902   1.0808455
## 10002  -0.3017980    2.0624936      -1.0185716      1.26334415   0.2487872
## 10003   0.7499109    0.9397098      -0.2767662      1.95498422   1.1753003
## 10004  -1.2881378   -1.2571586      -0.3499509     -0.92916445   1.2674129
## 10005  -1.2674419   -0.9580930      -1.0252180      2.50637631   1.4272885
## 10006  -1.3742028   -1.2023258      -1.2827539      0.06712530   1.8689377
##            ESTAB ESTAB_density   ESTAB_pop ESTAB_housing     PAYANN
## 10001  4.7248829     3.0987499  3.50805833    3.47468350  2.2434742
## 10002  1.3507796     0.5036173 -0.03770389   -0.01288174 -0.2027498
## 10003  2.3596040     1.7339504  0.45401000    0.38059199  1.6237148
## 10004  0.2157554     0.3193776  5.66263891    5.44275667  2.8684812
## 10005  0.0392305     4.9717217  1.57025605    1.39791847  1.4769524
## 10006 -0.3231101     2.2854283  3.00718075    2.52705864  0.2966241
##       PAYANN_density PAYANN_pop
## 10001      1.1197848  1.0270929
## 10002     -0.2208682 -0.2100989
## 10003      0.8602409  0.1612745
## 10004      1.6646186 11.1897435
## 10005      8.2083377  2.3633611
## 10006      2.2147918  2.2928277
```

```
set.seed(123)


fviz_nbclust(data_cluster,kmeans,method="wss")+geom_vline(xintercept=4,linetype=2)+labs(title="Elbow me
```



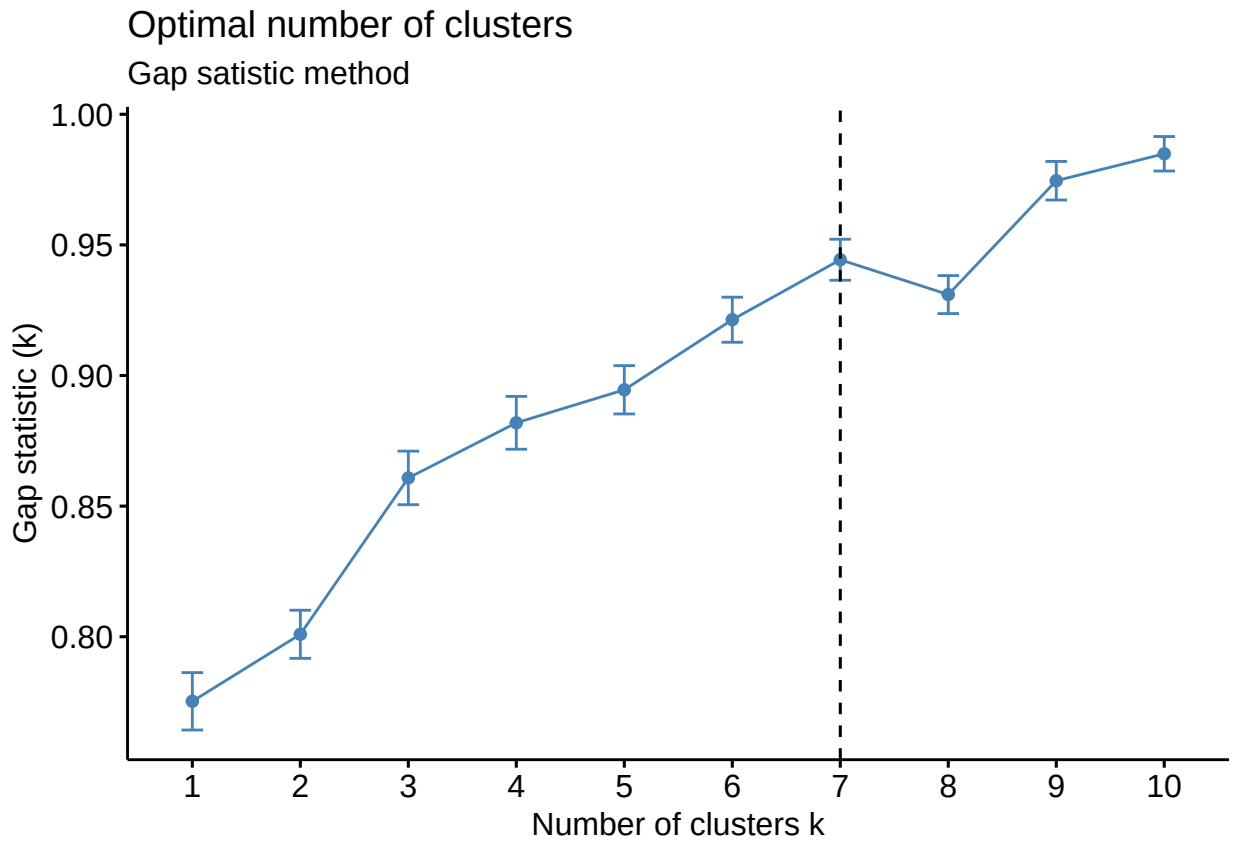

# Nach Elbow Methode soll k == 3 oder 4 sein.

```
fviz_nbclust(df,kmeans,method="silhouette")+labs(subtitle="Silhouette method")
```




#Nach Silouhette Methode ist 8 ein guter Wert für K.

```
fviz_nbclust(df,kmeans,nstart=2,method="gap_stat",nboot=20)+
  geom_vline(xintercept=7,linetype=2)+
  labs(subtitle="Gap satistic method")
```



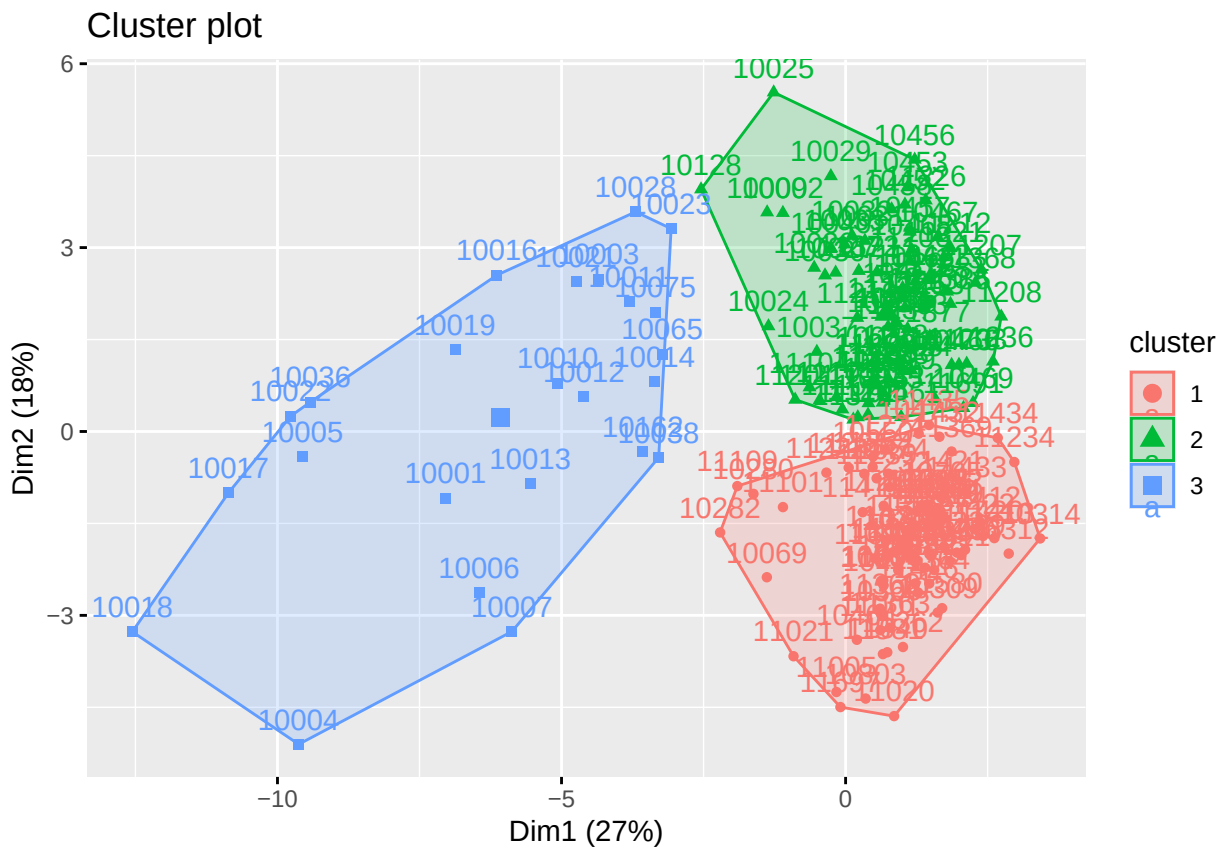

# Gap statistic methode emphehlt den Wert von 7. # Dann schauen wir welcher K-Wert Sinn macht.

```
df.k<-kmeans(df,3,iter.max=10,nstart=2)
print(df.k)
```

```
## K-means clustering with 3 clusters of sizes 85, 79, 24
##
## Cluster means:
##         ZCTA mean_native mean_condition mean_income        pop
## 1  0.4376052 -0.33557314    0.120044165   0.1657012 -0.5620788
## 2 -0.0443585 -0.02949793    0.003234914  -0.5934189  0.8123920
## 3 -1.4038383  1.28558554   -0.435804676   1.3664788 -0.6834279
##   eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp   eth_hisp   eth_other
## 1        0.08993953         -0.0420623         0.1845423 -0.2390721  0.23862822
## 2       -0.43986002          0.2513552        -0.2386365  0.5092503 -0.23152180
## 3        1.12933674         -0.6784068         0.1319244 -0.8295686 -0.08304901
##   count_tree    surface pop_density tree_density housing_ttl   housing_own
## 1  0.1266455  0.3800650  -0.7403406   -0.4708166  -0.6614787  0.0742600848
## 2  0.1156785 -0.1639171   0.4811033    0.4111990   0.7767495 -0.0003261594
## 3 -0.8293111 -0.8065028   1.0384079    0.3139453  -0.2140634 -0.2619308587
##   housing_rent housing_own_pct housing_density housing_pop       ESTAB
## 1   -0.8059063       0.7543811      -0.6639924  -0.2808871 -0.44828976
## 2    0.9041219      -0.6711724       0.2706775  -0.1967666 -0.01556357
## 3   -0.1218166      -0.4624907       1.4606595   1.6424986  1.63892295
##   ESTAB_density  ESTAB_pop ESTAB_housing     PAYANN PAYANN_density PAYANN_pop
## 1    -0.4161057 -0.2213778    -0.1846223 -0.3015824     -0.2769448 -0.1984784
```

```
## 2    -0.2158846 -0.2790873    -0.2722470 -0.2337340     -0.2476538 -0.2093330
## 3     2.1843278  1.7027090     1.5500171  1.8374786      1.7960399  1.3919987
##
## Clustering vector:
## 10001 10002 10003 10004 10005 10006 10007 10009 10010 10011 10012 10013 10014
##     3     2     3     3     3     3     3     2     3     3     3     3     3
## 10016 10017 10018 10019 10021 10022 10023 10024 10025 10026 10027 10028 10029
##     3     3     3     3     3     3     3     2     2     2     2     3     2
## 10030 10031 10032 10033 10034 10035 10036 10037 10038 10039 10040 10065 10069
##     2     2     2     2     2     2     3     2     3     2     2     3     1
## 10075 10128 10162 10280 10282 10301 10302 10303 10304 10305 10306 10307 10308
##     3     2     3     1     1     1     1     1     1     1     1     1     1
## 10309 10310 10312 10314 10451 10452 10453 10454 10455 10456 10457 10458 10459
##     1     1     1     1     2     2     2     2     2     2     2     2     2
## 10460 10461 10462 10463 10464 10465 10466 10467 10468 10469 10470 10471 10472
##     2     2     2     2     1     1     2     2     2     2     1     1     2
## 10473 10474 10475 10550 10704 10705 10803 11001 11004 11005 11020 11021 11040
##     2     1     1     1     1     1     1     1     1     1     1     1     1
## 11101 11102 11103 11104 11105 11106 11109 11201 11203 11204 11205 11206 11207
##     1     2     2     2     1     2     1     2     2     2     2     2     2
## 11208 11209 11210 11211 11212 11213 11214 11215 11216 11217 11218 11219 11220
##     2     2     2     2     2     2     2     2     2     2     2     2     2
## 11221 11222 11223 11224 11225 11226 11228 11229 11230 11231 11232 11233 11234
##     2     1     2     1     2     2     1     2     2     1     1     2     1
## 11235 11236 11237 11238 11239 11354 11355 11356 11357 11358 11360 11361 11362
##     2     2     2     2     1     1     2     1     1     1     1     1     1
## 11363 11364 11365 11366 11367 11368 11369 11370 11372 11373 11374 11375 11377
##     1     1     1     1     1     2     1     1     2     2     2     2     2
## 11378 11379 11385 11411 11412 11413 11414 11415 11416 11417 11418 11419 11420
##     1     1     2     1     1     1     1     1     1     1     1     1     1
## 11421 11422 11423 11426 11427 11428 11429 11432 11433 11434 11435 11436 11580
##     1     1     1     1     1     1     1     1     1     1     1     1     1
## 11581 11691 11692 11693 11694 11697
##     1     2     1     1     1     1
##
## Within cluster sum of squares by cluster:
## [1] 1582.6243  918.6894  905.5047
##  (between_SS / total_SS =  32.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(data=df,df.k)
```



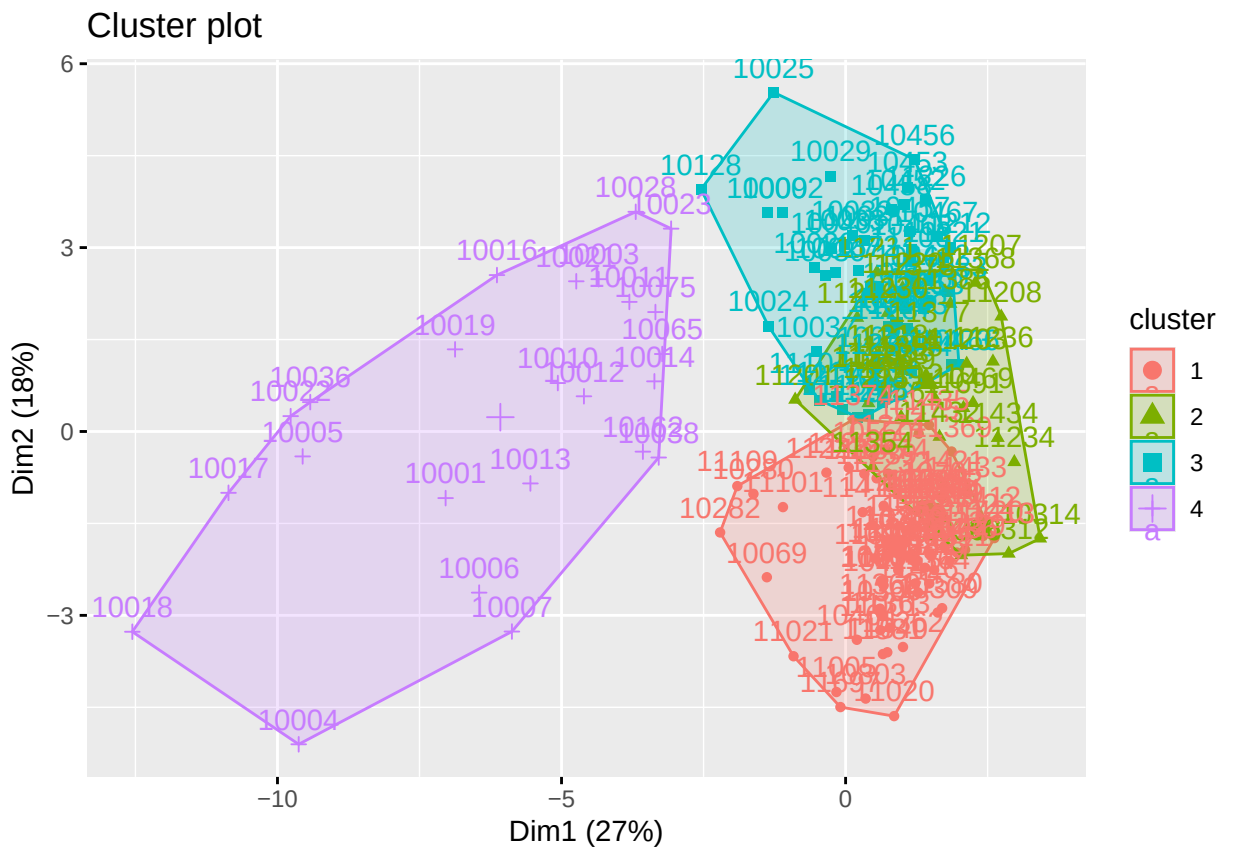

«««< HEAD

```
df.k<-kmeans(df,4,iter.max=10,nstart=2)
print(df.k)
```

```
## K-means clustering with 4 clusters of sizes 79, 34, 51, 24
##
## Cluster means:
##         ZCTA mean_native mean_condition mean_income        pop
## 1  0.4690759 -0.32611588     0.10545911   0.1746120 -0.6832388
## 2  0.5798022  0.02263665     0.14535985  -0.3335217  1.2767359
## 3 -0.4525128 -0.11491263    -0.05518024  -0.6911784  0.5288062
## 4 -1.4038383  1.28558554    -0.43580468   1.3664788 -0.6834279
##   eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp   eth_hisp   eth_other
## 1       0.087572201      -0.0632476516         0.1760700 -0.2101851  0.27744763
## 2       0.009681891      -0.0003059701         0.3443320 -0.2012384 -0.28451553
## 3      -0.673558231       0.4174260998        -0.5643726  0.8501251 -0.20101291
## 4       1.129336736      -0.6784068176         0.1319244 -0.8295686 -0.08304901
##   count_tree    surface pop_density tree_density housing_ttl housing_own
## 1 -0.1192183  0.1341041  -0.7193059  -0.47161353  -0.7686430  -0.1164064
## 2  1.2145668  0.9076172  -0.2612980   0.06351623   1.1033851   1.2452633
## 3 -0.2347757 -0.4332773   0.7997550   0.54045550   0.5557886  -0.5265982
## 4 -0.8293111 -0.8065028   1.0384079   0.31394525  -0.2140634  -0.2619309
##   housing_rent housing_own_pct housing_density housing_pop      ESTAB
## 1   -0.8379525      0.72484567      -0.6417943  -0.2453450 -0.5089160
## 2    0.6788626      0.06461355      -0.3600239  -0.4380686  0.4059205
```

```
## 3    0.9027553     -0.94823317       0.5467986  -0.1008505 -0.2535507
## 4   -0.1218166     -0.46249071       1.4606595   1.6424986  1.6389230
##   ESTAB_density  ESTAB_pop ESTAB_housing     PAYANN PAYANN_density PAYANN_pop
## 1    -0.4141359 -0.2225078    -0.1894110 -0.3077494     -0.2763011 -0.1976041
## 2    -0.2863251 -0.2220607    -0.1762797 -0.1924996     -0.2629664 -0.2068401
## 3    -0.1955309 -0.3085654    -0.3184987 -0.2596529     -0.2418885 -0.2110722
## 4     2.1843278  1.7027090     1.5500171  1.8374786      1.7960399  1.3919987
##
## Clustering vector:
## 10001 10002 10003 10004 10005 10006 10007 10009 10010 10011 10012 10013 10014
##     4     3     4     4     4     4     4     3     4     4     4     4     4
## 10016 10017 10018 10019 10021 10022 10023 10024 10025 10026 10027 10028 10029
##     4     4     4     4     4     4     4     3     3     3     3     4     3
## 10030 10031 10032 10033 10034 10035 10036 10037 10038 10039 10040 10065 10069
##     3     3     3     3     3     3     4     3     4     3     3     4     1
## 10075 10128 10162 10280 10282 10301 10302 10303 10304 10305 10306 10307 10308
##     4     3     4     1     1     1     1     1     1     1     2     1     1
## 10309 10310 10312 10314 10451 10452 10453 10454 10455 10456 10457 10458 10459
##     1     1     2     2     3     3     3     3     3     3     3     3     3
## 10460 10461 10462 10463 10464 10465 10466 10467 10468 10469 10470 10471 10472
##     3     2     3     3     1     1     3     3     3     2     1     1     3
## 10473 10474 10475 10550 10704 10705 10803 11001 11004 11005 11020 11021 11040
##     3     1     1     1     1     1     1     1     1     1     1     1     1
## 11101 11102 11103 11104 11105 11106 11109 11201 11203 11204 11205 11206 11207
##     1     3     3     3     1     3     1     2     2     2     3     3     2
## 11208 11209 11210 11211 11212 11213 11214 11215 11216 11217 11218 11219 11220
##     2     2     2     2     3     3     2     2     3     3     2     2     2
## 11221 11222 11223 11224 11225 11226 11228 11229 11230 11231 11232 11233 11234
##     3     1     2     1     3     3     1     2     2     1     1     3     2
## 11235 11236 11237 11238 11239 11354 11355 11356 11357 11358 11360 11361 11362
##     2     2     3     3     1     2     2     1     1     1     1     1     1
## 11363 11364 11365 11366 11367 11368 11369 11370 11372 11373 11374 11375 11377
##     1     1     1     1     1     2     1     1     3     2     1     2     2
## 11378 11379 11385 11411 11412 11413 11414 11415 11416 11417 11418 11419 11420
##     1     1     2     1     1     1     1     1     1     1     1     1     1
## 11421 11422 11423 11426 11427 11428 11429 11432 11433 11434 11435 11436 11580
##     1     1     1     1     1     1     1     2     1     2     1     1     1
## 11581 11691 11692 11693 11694 11697
##     1     2     1     1     1     1
##
## Within cluster sum of squares by cluster:
## [1] 1252.2860  476.6075  482.0997  905.5047
##  (between_SS / total_SS =  38.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(data=df,df.k)
```



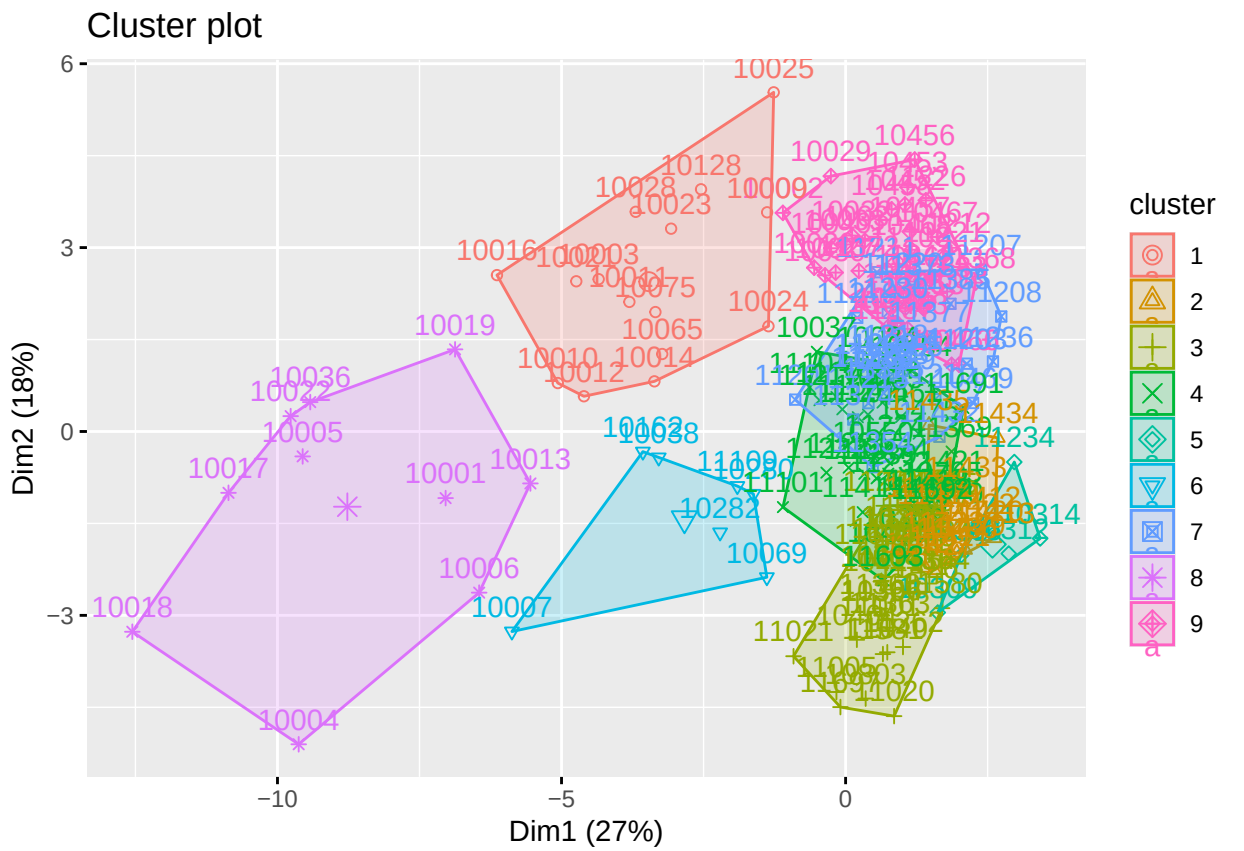

```
df.k<-kmeans(df,9,iter.max=10,nstart=2)
print(df.k)
```

```
## K-means clustering with 9 clusters of sizes 15, 15, 38, 34, 5, 7, 28, 10, 36
##
## Cluster means:
##         ZCTA mean_native mean_condition mean_income        pop
## 1 -1.3952856  0.97396421    -0.37018468  1.00522144  0.0934721
## 2  1.0612127 -0.08902333     0.31591201 -0.05285388 -0.3764470
## 3  0.3828626 -0.45686360     0.11946539  0.30971752 -0.7358507
## 4  0.2468197 -0.06766315    -0.05837407 -0.50075113 -0.4964173
## 5 -0.5769231 -0.46409753     0.18623947  0.36035730  0.7657206
## 6 -0.9593417  0.17776308     0.02588788  2.98029307 -1.4301360
## 7  0.7308077  0.06381571     0.13437383 -0.38734837  1.2555346
## 8 -1.4249705  1.33845595    -0.45328664  1.09274150 -1.0289940
## 9 -0.4039572 -0.21411119    -0.05785805 -0.88263165  0.8445126
##   eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp   eth_hisp
## 1       1.248555478         -0.6696105       -0.24851911 -0.7627209
## 2      -1.135931207          1.3752100        0.02580747 -0.4493296
## 3       0.590918771         -0.5314817        0.40742259 -0.4108367
## 4      -0.309513547          0.1118301       -0.29064743  0.5100355
## 5       1.137047950         -0.3716257       -0.51065691 -0.7052721
## 6       1.095239418         -0.6980781        0.29510216 -0.9382456
## 7      -0.006918063         -0.0586609        0.58202388 -0.2830507
## 8       0.859143799         -0.6462789        0.50917122 -0.7281103
```

```
## 9      -0.982512907          0.5738905       -0.64333839  1.1597764
##       eth_other   count_tree     surface pop_density tree_density housing_ttl
## 1 -0.0008670576 -0.530923287 -0.73756005  1.82799602    1.4139456   0.8797072
## 2  2.0657836984  0.271810560 -0.02116913 -0.60995014    0.1780910  -0.6567590
## 3 -0.2646187944 -0.007641333  0.43368415 -0.96955977   -0.6834061  -0.7960278
## 4 -0.0895184165 -0.441488241 -0.24244605 -0.37140755   -0.4501767  -0.5068155
## 5 -0.5505401585  3.919364347  4.55749915 -1.10501516   -0.6486909   0.6571824
## 6  0.3468529738 -1.137083959 -0.97444657  0.89246127   -0.1166578  -1.4086578
## 7 -0.1899497182  0.764295397  0.27290581 -0.08081842    0.2800999   1.1543119
## 8 -0.1904762487 -0.959255169 -0.79496232  0.24440374   -0.6117866  -0.7316917
## 9 -0.2868479081 -0.118258103 -0.34761121  0.84158791    0.5480560   0.7140955
##   housing_own housing_rent housing_own_pct housing_density housing_pop
## 1   0.4773971    0.7917553      -0.3283453       2.2912637  1.70788451
## 2   0.1843892   -0.8539394       1.0164543      -0.6721632 -0.99201271
## 3   0.1129916   -0.9813161       1.2709595      -0.8350387 -0.14425201
## 4  -0.6149842   -0.2909213      -0.4608214      -0.3538270 -0.09971638
## 5   2.9020399   -0.6456566       1.5751025      -0.9455398 -0.40542637
## 6  -1.1975298   -1.0573318      -0.7657894       1.0593322  0.98162413
## 7   0.9954322    0.8595547      -0.1253318      -0.2038212 -0.31488451
## 8  -0.7625863   -0.4808884      -0.6366694       0.6171832  1.64916668
## 9  -0.5467985    1.0968071      -0.9885881       0.4534060 -0.39959114
##         ESTAB ESTAB_density  ESTAB_pop ESTAB_housing     PAYANN PAYANN_density
## 1  1.05621011     1.1767465  0.2611457     0.1147603  0.5682683      0.3902387
## 2 -0.64864886    -0.4514053 -0.3716034    -0.3459465 -0.3586422     -0.2873400
## 3 -0.47614317    -0.4477959 -0.1858640    -0.1426596 -0.3171821     -0.2844589
## 4 -0.39153855    -0.3336913 -0.2072018    -0.1843372 -0.2578512     -0.2592223
## 5  0.03257211    -0.4913624 -0.2602003    -0.2270848 -0.2646938     -0.2907824
## 6 -0.55980389     0.4267177  0.1965280     0.2361964 -0.1104946      0.1827224
## 7  0.49135437    -0.2318790 -0.1992285    -0.1544795 -0.1841069     -0.2569056
## 8  2.52892850     3.2260330  3.3716973     3.1430093  3.4118828      3.4336577
## 9 -0.27775300    -0.2448973 -0.3457975    -0.3462860 -0.2553176     -0.2469125
##    PAYANN_pop
## 1  0.02063358
## 2 -0.22534451
## 3 -0.20226297
## 4 -0.18217130
## 5 -0.21465872
## 6  0.07509272
## 7 -0.20535480
## 8  3.09591468
## 9 -0.21419686
##
## Clustering vector:
## 10001 10002 10003 10004 10005 10006 10007 10009 10010 10011 10012 10013 10014
##     8     9     1     8     8     8     6     1     1     1     1     8     1
## 10016 10017 10018 10019 10021 10022 10023 10024 10025 10026 10027 10028 10029
##     1     8     8     8     1     8     1     1     1     9     9     1     9
## 10030 10031 10032 10033 10034 10035 10036 10037 10038 10039 10040 10065 10069
##     9     9     9     9     4     4     8     4     6     4     9     1     6
## 10075 10128 10162 10280 10282 10301 10302 10303 10304 10305 10306 10307 10308
##     1     1     6     6     6     3     4     3     3     3     5     3     3
## 10309 10310 10312 10314 10451 10452 10453 10454 10455 10456 10457 10458 10459
##     5     3     5     5     9     9     9     4     9     9     9     9     9
## 10460 10461 10462 10463 10464 10465 10466 10467 10468 10469 10470 10471 10472
```

```
##     9     4     9     9     3     3     9     9     9     7     4     3     9
## 10473 10474 10475 10550 10704 10705 10803 11001 11004 11005 11020 11021 11040
##     9     4     4     4     3     4     3     3     3     3     3     3     3
## 11101 11102 11103 11104 11105 11106 11109 11201 11203 11204 11205 11206 11207
##     4     4     4     4     4     4     6     7     7     7     4     9     7
## 11208 11209 11210 11211 11212 11213 11214 11215 11216 11217 11218 11219 11220
##     7     7     7     7     9     9     7     7     9     4     7     7     7
## 11221 11222 11223 11224 11225 11226 11228 11229 11230 11231 11232 11233 11234
##     9     4     7     4     9     9     3     7     7     4     4     9     5
## 11235 11236 11237 11238 11239 11354 11355 11356 11357 11358 11360 11361 11362
##     7     7     9     7     4     7     7     3     3     3     3     3     3
## 11363 11364 11365 11366 11367 11368 11369 11370 11372 11373 11374 11375 11377
##     3     3     3     3     3     9     4     4     9     7     7     7     7
## 11378 11379 11385 11411 11412 11413 11414 11415 11416 11417 11418 11419 11420
##     3     3     7     2     2     2     3     4     4     2     4     2     2
## 11421 11422 11423 11426 11427 11428 11429 11432 11433 11434 11435 11436 11580
##     4     2     2     3     2     2     2     7     2     2     2     2     3
## 11581 11691 11692 11693 11694 11697
##     3     4     4     4     3     3
##
## Within cluster sum of squares by cluster:
## [1] 137.33798 171.84259 494.75717 293.38036  52.84965 110.11355 228.14211
## [8] 398.83990 263.07448
##  (between_SS / total_SS =  57.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(data=df,df.k)
```




# k=3 oder k= 4 sieht gut aus. K=7 ist schon schlecht. # Deshalb 3 und 4 sind die besten Werte für k. Da wenn k=3, es weniger Überlappung gibt, ist 3 der beste Wert für K.Jetzt checken wir, mean von allen Clusters to

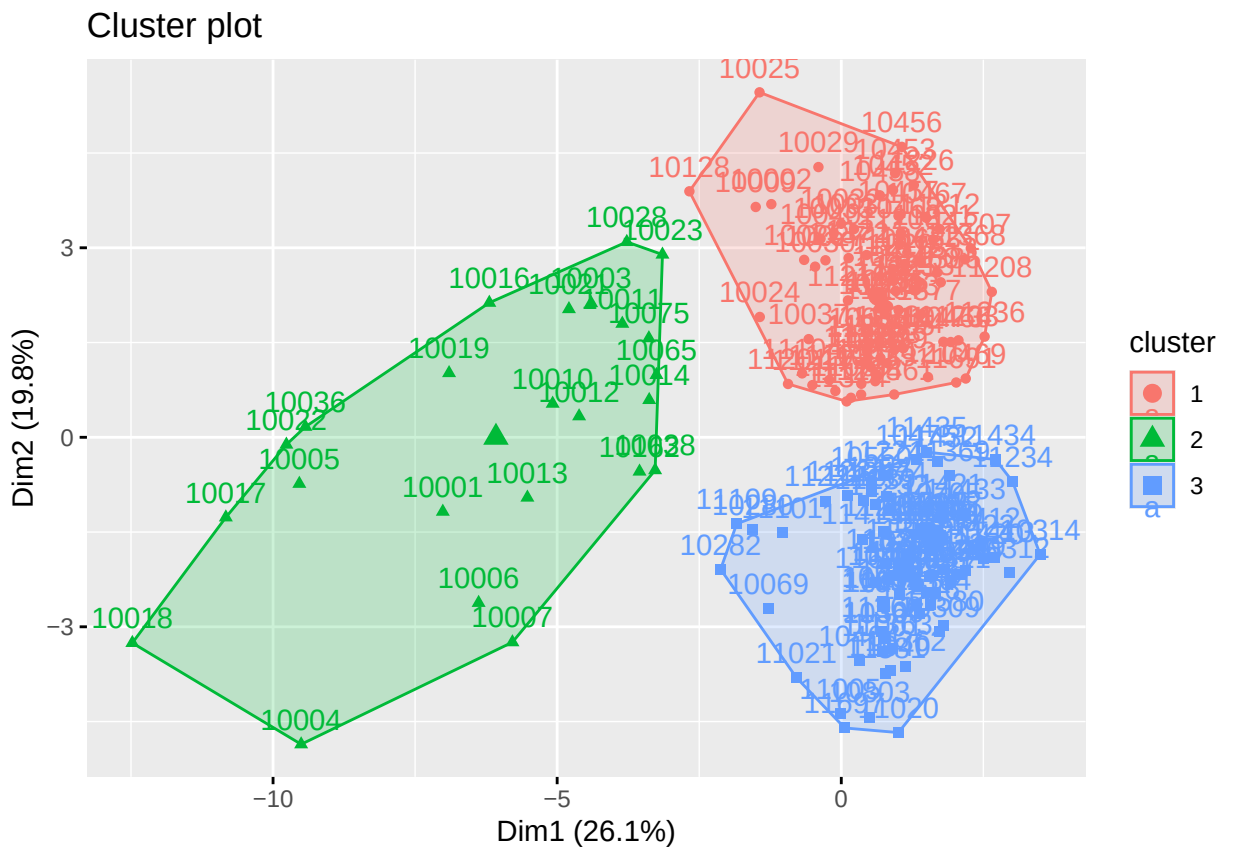
```
df.k<-kmeans(df,3,iter.max=10,nstart=2)
print(df.k)
```

```
## K-means clustering with 3 clusters of sizes 79, 24, 85
##
## Cluster means:
##         ZCTA mean_native mean_condition mean_income        pop
## 1 -0.0443585 -0.02949793    0.003234914  -0.5934189  0.8123920
## 2 -1.4038383  1.28558554   -0.435804676   1.3664788 -0.6834279
## 3  0.4376052 -0.33557314    0.120044165   0.1657012 -0.5620788
##   eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp   eth_hisp   eth_other
## 1       -0.43986002          0.2513552        -0.2386365  0.5092503 -0.23152180
## 2        1.12933674         -0.6784068         0.1319244 -0.8295686 -0.08304901
## 3        0.08993953         -0.0420623         0.1845423 -0.2390721  0.23862822
##   count_tree    surface pop_density tree_density housing_ttl   housing_own
## 1  0.1156785 -0.1639171   0.4811033    0.4111990   0.7767495 -0.0003261594
## 2 -0.8293111 -0.8065028   1.0384079    0.3139453  -0.2140634 -0.2619308587
## 3  0.1266455  0.3800650  -0.7403406   -0.4708166  -0.6614787  0.0742600848
##   housing_rent housing_own_pct housing_density housing_pop       ESTAB
## 1    0.9041219      -0.6711724       0.2706775  -0.1967666 -0.01556357
## 2   -0.1218166      -0.4624907       1.4606595   1.6424986  1.63892295
## 3   -0.8059063       0.7543811      -0.6639924  -0.2808871 -0.44828976
```

```
##   ESTAB_density  ESTAB_pop ESTAB_housing     PAYANN PAYANN_density PAYANN_pop
## 1    -0.2158846 -0.2790873    -0.2722470 -0.2337340     -0.2476538 -0.2093330
## 2     2.1843278  1.7027090     1.5500171  1.8374786      1.7960399  1.3919987
## 3    -0.4161057 -0.2213778    -0.1846223 -0.3015824     -0.2769448 -0.1984784
##
## Clustering vector:
## 10001 10002 10003 10004 10005 10006 10007 10009 10010 10011 10012 10013 10014
##     2     1     2     2     2     2     2     1     2     2     2     2     2
## 10016 10017 10018 10019 10021 10022 10023 10024 10025 10026 10027 10028 10029
##     2     2     2     2     2     2     2     1     1     1     1     2     1
## 10030 10031 10032 10033 10034 10035 10036 10037 10038 10039 10040 10065 10069
##     1     1     1     1     1     1     2     1     2     1     1     2     3
## 10075 10128 10162 10280 10282 10301 10302 10303 10304 10305 10306 10307 10308
##     2     1     2     3     3     3     3     3     3     3     3     3     3
## 10309 10310 10312 10314 10451 10452 10453 10454 10455 10456 10457 10458 10459
##     3     3     3     3     1     1     1     1     1     1     1     1     1
## 10460 10461 10462 10463 10464 10465 10466 10467 10468 10469 10470 10471 10472
##     1     1     1     1     3     3     1     1     1     1     3     3     1
## 10473 10474 10475 10550 10704 10705 10803 11001 11004 11005 11020 11021 11040
##     1     3     3     3     3     3     3     3     3     3     3     3     3
## 11101 11102 11103 11104 11105 11106 11109 11201 11203 11204 11205 11206 11207
##     3     1     1     1     3     1     3     1     1     1     1     1     1
## 11208 11209 11210 11211 11212 11213 11214 11215 11216 11217 11218 11219 11220
##     1     1     1     1     1     1     1     1     1     1     1     1     1
## 11221 11222 11223 11224 11225 11226 11228 11229 11230 11231 11232 11233 11234
##     1     3     1     3     1     1     3     1     1     3     3     1     3
## 11235 11236 11237 11238 11239 11354 11355 11356 11357 11358 11360 11361 11362
##     1     1     1     1     3     3     1     3     3     3     3     3     3
## 11363 11364 11365 11366 11367 11368 11369 11370 11372 11373 11374 11375 11377
##     3     3     3     3     3     1     3     3     1     1     1     1     1
## 11378 11379 11385 11411 11412 11413 11414 11415 11416 11417 11418 11419 11420
##     3     3     1     3     3     3     3     3     3     3     3     3     3
## 11421 11422 11423 11426 11427 11428 11429 11432 11433 11434 11435 11436 11580
##     3     3     3     3     3     3     3     3     3     3     3     3     3
## 11581 11691 11692 11693 11694 11697
##     3     1     3     3     3     3
##
## Within cluster sum of squares by cluster:
## [1]  918.6894  905.5047 1582.6243
##  (between_SS / total_SS =  32.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
df<-data.frame(df)
df$cluster<-df.k$cluster
```

```
fviz_cluster(data=df,df.k)
```



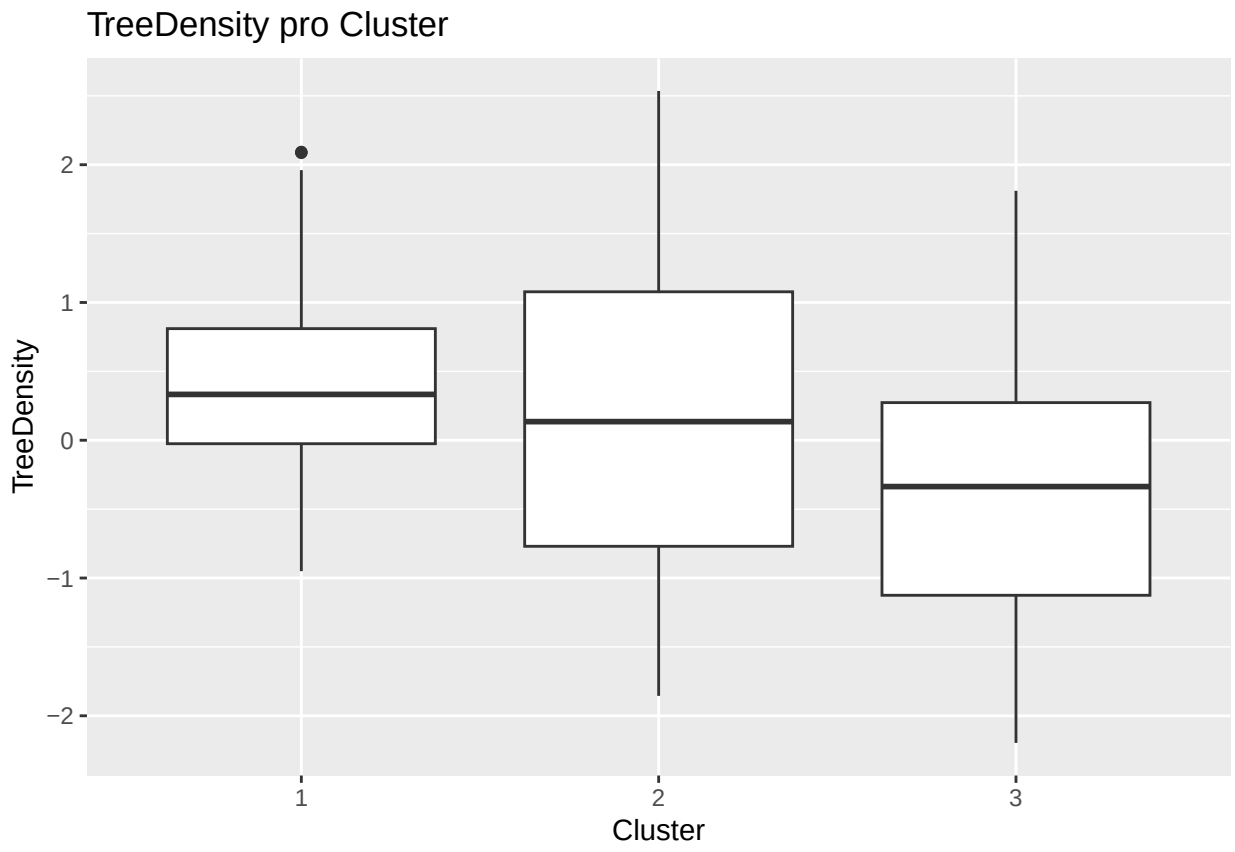

#Wir checken jetzt Mittelwert aller Cluster, um wesentliche Unterschiede zwischen den Clustern zu untersuchen

```
aggregate(df,by=list(cluster=df$cluster),mean)
```

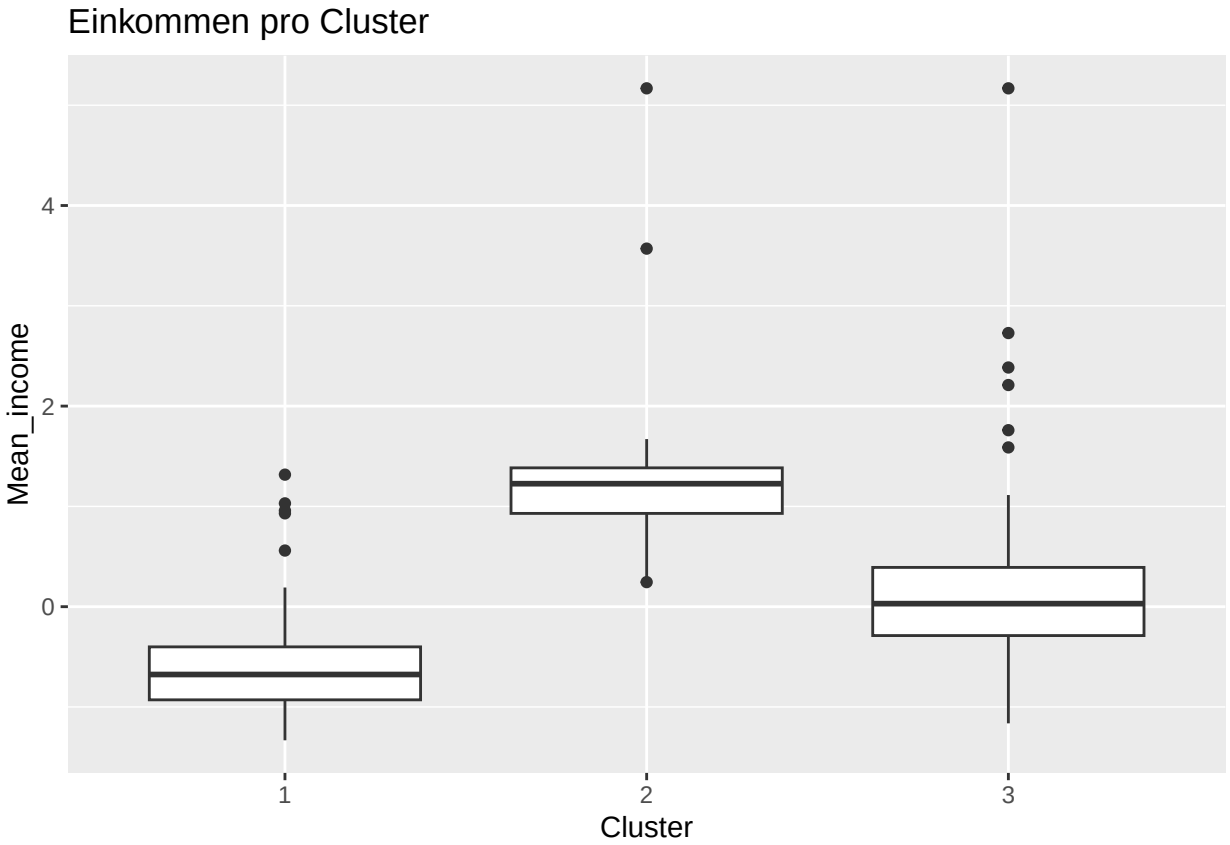
```
##   cluster       ZCTA mean_native mean_condition mean_income        pop
## 1       1 -0.0443585 -0.02949793    0.003234914  -0.5934189  0.8123920
## 2       2 -1.4038383  1.28558554   -0.435804676   1.3664788 -0.6834279
## 3       3  0.4376052 -0.33557314    0.120044165   0.1657012 -0.5620788
##   eth_white_nonhisp eth_afroam_nonhisp eth_asian_nonhisp   eth_hisp   eth_other
## 1       -0.43986002          0.2513552        -0.2386365  0.5092503 -0.23152180
## 2        1.12933674         -0.6784068         0.1319244 -0.8295686 -0.08304901
## 3        0.08993953         -0.0420623         0.1845423 -0.2390721  0.23862822
##   count_tree    surface pop_density tree_density housing_ttl   housing_own
## 1  0.1156785 -0.1639171   0.4811033    0.4111990   0.7767495 -0.0003261594
## 2 -0.8293111 -0.8065028   1.0384079    0.3139453  -0.2140634 -0.2619308587
## 3  0.1266455  0.3800650  -0.7403406   -0.4708166  -0.6614787  0.0742600848
##   housing_rent housing_own_pct housing_density housing_pop       ESTAB
## 1    0.9041219      -0.6711724       0.2706775  -0.1967666 -0.01556357
## 2   -0.1218166      -0.4624907       1.4606595   1.6424986  1.63892295
## 3   -0.8059063       0.7543811      -0.6639924  -0.2808871 -0.44828976
##   ESTAB_density  ESTAB_pop ESTAB_housing     PAYANN PAYANN_density PAYANN_pop
## 1    -0.2158846 -0.2790873    -0.2722470 -0.2337340     -0.2476538 -0.2093330
## 2     2.1843278  1.7027090     1.5500171  1.8374786      1.7960399  1.3919987
## 3    -0.4161057 -0.2213778    -0.1846223 -0.3015824     -0.2769448 -0.1984784
##   cluster
```

```
## 1       1
## 2       2
## 3       3
```

```r
library(ggplot2)
ggplot(df, aes(x = as.factor(cluster), y = tree_density)) +
  geom_boxplot() +
  labs(title = "TreeDensity pro Cluster", x = "Cluster", y = "TreeDensity")
```




```r
ggplot(df, aes(x = as.factor(cluster), y = mean_income)) +
  geom_boxplot() +
  labs(title = "Einkommen pro Cluster", x = "Cluster", y = "Mean_income")
```




======= »»»> 1c37c08bb3ca1918284436b4e833dde27a8e386f #Literatur #McCoy, Dakota; Goulet-Scott, Benjamin; Meng, Weilin et al. (2022). A dataset of 5 million city trees from 63 US cities: species, location, nativity status, health, and more. [Dataset]. Dryad. https://doi.org/10.5061/dryad.2jm63xsrf #Burghardt KT, Avolio ML, Locke DH, Grove JM, Sonti NF, Swan CM. Current street tree communities reflect race-based housing policy and modern attempts to remedy environmental injustice. Ecology. 2023 Feb;104(2):e3881. doi: 10.1002/ecy.3881. Epub 2022 Dec 1. PMID: 36196604; PMCID: PMC10078568.