Stat 420 Final Project - Modeling

Shashank Thakur, Net ID: sthakur5

27th July 2022

Modeling California Housing Project Data Set

Initialization

Read the data file to be used for this project.

```
ca_housing_data = read.csv('../000_Data/california-housing-prices/housing.csv')
# head(ca_housing_data)
```

Initialize global variables.

```
# build different models to test the performance
n_models = 9
# create a vector list of models to pass to various functions
cahoudta_models = vector("list", length = n_models)
cahoudta_model_names = vector("character", length = n_models)
# Train-Test Split Ration
tr_pct = 0.70
```

Helper Functions

Function : `model_performance_metrics` will generate model performance metrics such as RMSE and R-squared, etc.

Inputs:

models : A list of models to generate the performance metrics

model_names : Names of the models passed to the function

data : The actual data which will be used to calculate predicted values and RMSE for each model

Output : Table with following performance metrics as columns and a row for each model provided in `model_names` as input.

```
Adjusted R-Squared
Root Mean Square Error on Training Data
Root Mean Square Error on Test Data
Leave One Out Cross Validated RMSE
Total number of coefficients in the model
```

```
model_performance_metrics = function(models, model_names, data){
  # Calculate model names and total numbers
  n_mod = length(models)
  # Initialize the data frame to be be built by the function
  perf_metrics_df = data.frame(
    row.names = model_names,
    "ADJR2" = rep(0L, n_mod),
    "Train.RMSE" = rep(0L, n_mod),
    "Test.RMSE"  = rep(0L, n_mod),
    "LOOCV_RMSE" = rep(0L, n_mod),
    "Coefs" = rep(0L, n_mod)
  )
  # Extract actual response values
  y_i = log(data[["median_house_value"]])

  for (idx in 1:n_mod){
    perf_metrics_df[model_names[idx], "ADJR2"] = summary(models[[idx]])$adj.r.squared
    perf_metrics_df[model_names[idx], "Train.RMSE"] = sqrt(mean(resid(models[[idx]]) ^ 2))
    y_hat = predict(models[[idx]], newdata = data)
    perf_metrics_df[idx, "Test.RMSE"] = sqrt(mean((y_i - y_hat) ^ 2))
    perf_metrics_df[model_names[idx], "LOOCV_RMSE"] = sqrt(mean((resid(models[[idx]]) /
                                                      (1 - hatvalues(models[[idx]]))) ^ 2))
    perf_metrics_df[model_names[idx], "Coefs"] = length(coef(models[[idx]]))
  }

  # Print Model performance on train data
  kable(perf_metrics_df,
        col.names = c("Adjusted R-Squared",
                      "Train RMSE",
                      "Test RMSE",
                      "LOOCV RMSE",
                      "Coefficients"),
        caption = "Comparative Model Performance Metrics")

}
```

Data cleaning

During initial data analysis phase, we determined that some of the columns have high multicollinearity and we decided to drop those columns. One example of such column is `total_bedrooms` which not only has multicollinearity with `total_rooms` but also has some missing data. We will drop this column to allow for modeling **response** variable `median_house_value`.

```
ca_housing_data = subset(ca_housing_data, select = -total_bedrooms)
head(ca_housing_data)
```

```
##   longitude latitude housing_median_age total_rooms population households
## 1    -122.2    37.88                41         880        322        126
## 2    -122.2    37.86                21        7099       2401       1138
## 3    -122.2    37.85                52        1467        496        177
## 4    -122.2    37.85                52        1274        558        219
## 5    -122.2    37.85                52        1627        565        259
```

```
## 6    -122.2    37.85                52         919        413        193
##   median_income median_house_value ocean_proximity
## 1         8.325             452600        NEAR BAY
## 2         8.301             358500        NEAR BAY
## 3         7.257             352100        NEAR BAY
## 4         5.643             341300        NEAR BAY
## 5         3.846             342200        NEAR BAY
## 6         4.037             269700        NEAR BAY
```

We have another column `ocean_proximity` which is a categorical variable indicating how close the block is from ocean.

```
table(ca_housing_data$ocean_proximity)
```

```
##
##  <1H OCEAN     INLAND     ISLAND   NEAR BAY NEAR OCEAN
##       9136       6551          5       2290       2658
```

As can be seen, this column has 5 different categories. The `ISLAND` category has only 5 observations. During initial analysis, our team determined that `ISLAND` has high leverage on the predicting response variable `median_house_value` which is problematic for our purpose. We will be dropping the rows from data set with `ISLAND` category and coercing the predictor `ocean_proximity` to be categorical.

```
ca_housing_data = subset(ca_housing_data,
                         subset = ca_housing_data$ocean_proximity != "ISLAND")
table(ca_housing_data$ocean_proximity)
```

```
##
##  <1H OCEAN     INLAND   NEAR BAY NEAR OCEAN
##       9136       6551       2290       2658
```

```
# Coerce ocean_proximity to a factor variable
ca_housing_data$ocean_proximity = as.factor(ca_housing_data$ocean_proximity)
is.factor(ca_housing_data$ocean_proximity)
```

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

```
levels(ca_housing_data$ocean_proximity)
```

```
## [1] "<1H OCEAN"  "INLAND"     "NEAR BAY"   "NEAR OCEAN"
```

Model Building

Now that we have data cleaned, we will proceed with building a model. The very first step we will take is to divide data into train and test. Based on total number of observations team has decided to use 70%-30% split for train-test data.

```
set.seed(420072022)
# Randomly select train indexes
cahoudta_trn_idx = sample(nrow(ca_housing_data),
                          size = trunc(tr_pct * nrow(ca_housing_data)))
# Split into train and test data sets based on the index
ca_housing_data.tr = ca_housing_data[cahoudta_trn_idx, ]
ca_housing_data.te = ca_housing_data[-cahoudta_trn_idx, ]

n_tr_rows = nrow(ca_housing_data.tr)
n_te_rows = nrow(ca_housing_data.te)
```

Next we will build models and start running tests. While recursively analyzing different model predictors and transformations, it was identified that there are few outlier observations which result in issues with the model training. So we circled back to the very first full additive model and identified these outliers.

```
cahoudta_full_add_model_cd = lm(log(median_house_value) ~ ., data = ca_housing_data.tr)
cd_full_add_mod = cooks.distance(cahoudta_full_add_model_cd)
(count_outliers = sum(cd_full_add_mod > 4 / length(cd_full_add_mod)))
```

```
## [1] 841
```

We will now adjust our full additive model by ignoring these **841** outliers while training. This fitting gives best Test RMSE.

```
cahoudta_full_add_model = lm(log(median_house_value) ~ .,
                             data = ca_housing_data.tr,
                             subset = (cd_full_add_mod <= 4 / length(cd_full_add_mod)))
# summary(cahoudta_full_add_model)
```

Checking if there is multi-collinearity in the full fitted additive model. Following code snippet outputs all the predictors which has high variation inflation factors.

```
high_vif_predictors = vif(cahoudta_full_add_model) > 5
high_vif_pred_vals = vif(cahoudta_full_add_model)[high_vif_predictors]
sort(high_vif_pred_vals)
```

```
##  population total_rooms  households   longitude    latitude
##        7.35       11.23       13.55       19.84       22.32
```

Based on variable inflation factors the predictors `households` and `latitude` can be dropped.

Lets try backward BIC search from the full additive model.

```
cahoudta_add_bckwd_bic_model = step(cahoudta_full_add_model,
                                    direction = "backward",
                                    k = log(n_tr_rows),
                                    trace = 0)
```

Lets try two way interaction full additive model.

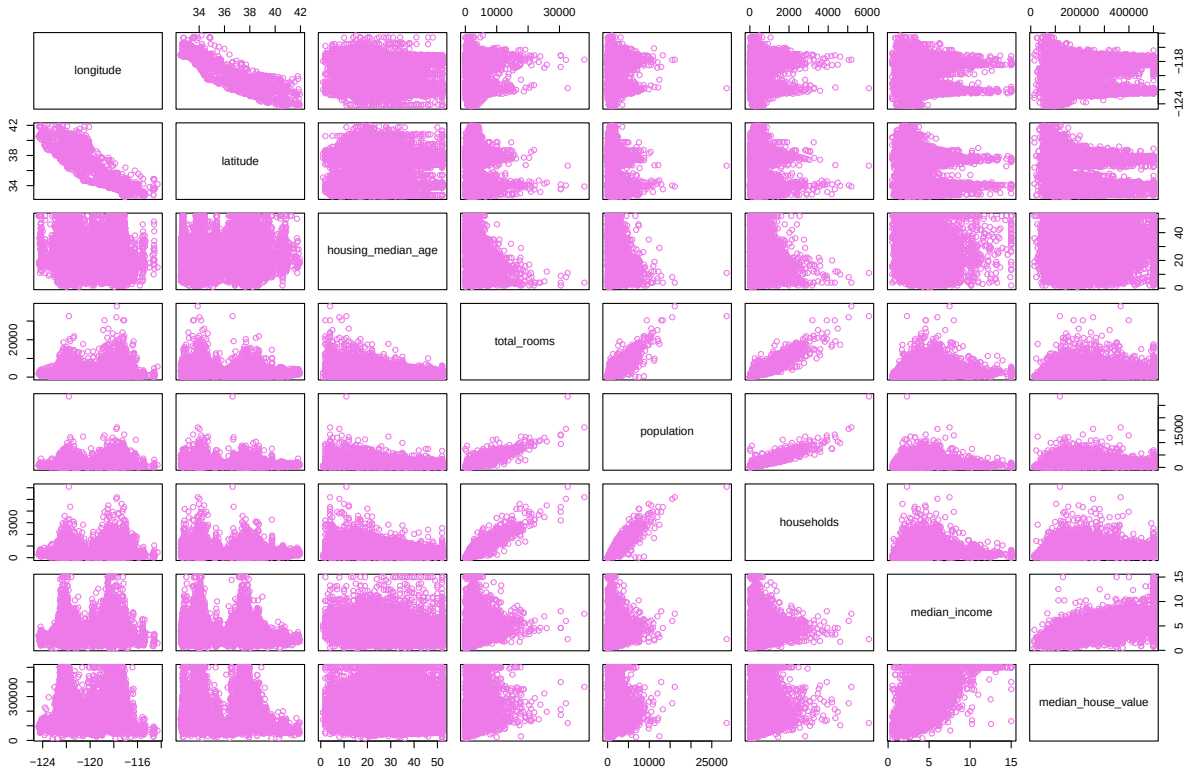
```
cahoudta_2int_add_model = lm(log(median_house_value) ~ . ^ 2,
                             data = ca_housing_data.tr)
```

Lets try backward BIC search from two way interaction full additive model.

```
cahoudta_2int_bckwd_bic_model = step(cahoudta_2int_add_model,
                                     direction = "backward",
                                     k = log(n_tr_rows),
                                     trace = 0)
```

We will run pairs plot to check any obvious trends in predictors compared to the response variable `median_house_value`.

```
pairs(subset(ca_housing_data.tr,
             select = -ocean_proximity),
      col = "orchid2")
```



Based on pairs plot and for response variable `median_house_value`

- Variables `median_income`, `households`, `total_rooms` have some polynomial relationship.
- Variable `housing_median_age` has lot of variance and needs variable transformation.
- Either one of `latitude` or `longitude` can be eliminated due to high partial collinearity.

Lets drop the predictors `households` and `latitude` with high VIF.

```
predictor_variables = colnames(ca_housing_data.tr)[-which(colnames(ca_housing_data.tr)
                                                          == "median_house_value")]
drop_cols = which(colnames(ca_housing_data.tr) == "median_house_value" |
                    colnames(ca_housing_data.tr) == "households" |
                    colnames(ca_housing_data.tr) == "latitude")
smaller_predictors = colnames(ca_housing_data.tr)[-drop_cols]
less_preds = paste0("log(median_house_value) ~ ",
                    paste(smaller_predictors, collapse = " + "))
cahoudta_lowvif_add_mod = lm(formula = less_preds, data = ca_housing_data.tr)
```

Lets add 2 way interaction for above model to check the performance.

```
less_preds_2int = paste0("log(median_house_value) ~ (",
                         paste(smaller_predictors, collapse = " + "), ") ^ 2")
cahoudta_lowvif_2int_mod = lm(formula = less_preds_2int, data = ca_housing_data.tr)
```

To increase complexity, lets try a degree 2 polynomial for reduced predictors based on vif above. Also we have to exclude the categorical predictor `ocean_proximity` from polynomial formula.

```
drop_cols = which(colnames(ca_housing_data.tr) == "median_house_value" |
                    colnames(ca_housing_data.tr) == "households" |
                    colnames(ca_housing_data.tr) == "latitude")
reduced_predictors = colnames(ca_housing_data.tr)[-drop_cols]
less_preds = paste0("log(median_house_value) ~ ", paste(smaller_predictors,
                                                        collapse = " + "))
further_reduced_predictors = reduced_predictors[-which(reduced_predictors
                                                       == "ocean_proximity")]
less_poly_preds = paste0(" + I(", paste(further_reduced_predictors,
                                        collapse = " ^ 2) + I("), " ^ 2)")
poly_formula = paste0(less_preds, less_poly_preds)
message("Polynomial Formula: ", poly_formula)
```

```
## Polynomial Formula: log(median_house_value) ~ longitude + housing_median_age + total_rooms + populati
```

```
cahoudta_lowvif_2poly_mod = lm(formula = poly_formula, data = ca_housing_data.tr)
# summary(cahoudta_lowvif_2poly_mod)$coef
```

I will add back `latitude` and test if that gives better results.

```
drop_cols = which(colnames(ca_housing_data.tr) == "median_house_value" |
                    colnames(ca_housing_data.tr) == "households")
lesser_predictors = colnames(ca_housing_data.tr)[-drop_cols]
less_preds = paste0("log(median_house_value) ~ ",
                    paste(lesser_predictors,
                          collapse = " + "))
message("Model 8: Formula: ", less_preds)
```

```
## Model 8: Formula: log(median_house_value) ~ longitude + latitude + housing_median_age + total_rooms +
```

```
cahoudta_lola_add_mod = lm(formula = less_preds, data = ca_housing_data.tr)
```

Next we will transform predictors `population` and `total_rooms` to be normalized with predictor `households` and build a new model.

```
ca_housing_data.tr$pop_per_hh = (ca_housing_data.tr$population /
                                   ca_housing_data.tr$households)
ca_housing_data.tr$rooms_per_hh = (ca_housing_data.tr$total_rooms /
                                     ca_housing_data.tr$households)
ca_housing_data.te$pop_per_hh = (ca_housing_data.te$population /
                                   ca_housing_data.te$households)
ca_housing_data.te$rooms_per_hh = (ca_housing_data.te$total_rooms /
                                     ca_housing_data.te$households)
```

Fit a additive model with transformed variables and predictors withou high variation inflation factors or collinearity.

```
cahoudta_2int_txfd_model_cd = lm(log(median_house_value) ~
                                   (longitude +
                                      housing_median_age +
                                      median_income +
                                      pop_per_hh +
                                      rooms_per_hh +
                                      ocean_proximity ) ^ 2,
                                 data = ca_housing_data.tr)
cd_2int_txfd = cooks.distance(cahoudta_2int_txfd_model_cd)
cahoudta_2int_txfd_model = lm(log(median_house_value) ~
                                   (longitude +
                                      housing_median_age +
                                      median_income +
                                      pop_per_hh +
                                      rooms_per_hh +
                                      ocean_proximity ) ^ 2,
                                 data = ca_housing_data.tr,
                                 subset = (cd_2int_txfd <= 4 / length(cd_2int_txfd)))
```

Comparative Model Performance Metrics

We will now compare all the models to calculate and display the performance metrics.

```
cahoudta_models[[1]] = cahoudta_full_add_model
cahoudta_model_names[[1]] = "cahoudta_full_add_model"

cahoudta_models[[2]] = cahoudta_add_bckwd_bic_model
cahoudta_model_names[[2]] = "cahoudta_add_bckwd_bic_model"

cahoudta_models[[3]] = cahoudta_2int_add_model
cahoudta_model_names[[3]] = "cahoudta_2int_add_model"

cahoudta_models[[4]] = cahoudta_2int_bckwd_bic_model
cahoudta_model_names[[4]] = "cahoudta_2int_bckwd_bic_model"
```

```r
cahoudta_models[[5]] = cahoudta_lowvif_add_mod
cahoudta_model_names[[5]] = "cahoudta_lowvif_add_mod"

cahoudta_models[[6]] = cahoudta_lowvif_2int_mod
cahoudta_model_names[[6]] = "cahoudta_lowvif_2int_mod"

cahoudta_models[[7]] = cahoudta_lowvif_2poly_mod
cahoudta_model_names[[7]] = "cahoudta_lowvif_2poly_mod"

cahoudta_models[[8]] = cahoudta_lola_add_mod
cahoudta_model_names[[8]] = "cahoudta_lola_add_mod"

cahoudta_models[[9]] = cahoudta_2int_txfd_model
cahoudta_model_names[[9]] = "cahoudta_2int_txfd_model"

# Print Comparative Model performance on train data
model_performance_metrics(cahoudta_models,
                          cahoudta_model_names,
                          ca_housing_data.te)
```

Table 1: Comparative Model Performance Metrics

	Adjusted R-Squared	Train RMSE	Test RMSE	LOOCV RMSE	Coefficients
cahoudta_full_add_model	0.7597	0.2687	0.3407	0.2690	11
cahoudta_add_bckwd_bic_model	0.7596	0.2688	0.3407	0.2690	10
cahoudta_2int_add_model	0.7150	0.3038	0.3090	0.3069	53
cahoudta_2int_bckwd_bic_model	0.7144	0.3043	0.3094	0.3065	44
cahoudta_lowvif_add_mod	0.6310	0.3463	0.3456	0.3465	9
cahoudta_lowvif_2int_mod	0.6645	0.3299	0.3321	0.3311	34
cahoudta_lowvif_2poly_mod	0.6667	0.3290	0.3335	0.3321	14
cahoudta_lola_add_mod	0.6515	0.3365	0.3379	0.3368	10
cahoudta_2int_txfd_model	0.7702	0.2635	1.6615	0.2664	34

As we can see from the results that the backward BIC search does help in reducing number of beta parameters by 1. Compared to the full additive model `cahoudta_full_add_model` and backward step wise search model `cahoudta_add_bckwd_bic_model` has one less parameter i.e. 10. To check if we can get as simpler model (less coefficients) we tried to change the train-test split to 50%-50%. But that did not help with reducing number of beta parameters.

We do however note, that having both `latitude` and `longitude` reduces the adjusted r-squared further and is not a good predictor combination.

Model Selection

During the model building process, we explored **9** different models for our housing data set. The main goal we had for this project was to effectively model the California Housing Data and identify the predictors which help explain the `median_house_value` response variable. Based on table Comparative Model Performance Metrics we select model `cahoudta_add_bckwd_bic_model` as our best model for purpose of this project.

This model has following features:

- High `adjusted r-squared` of **0.7596**

- Lowest `Test RMSE` of **0.3407**
- Low `Train RMSE` of **0.2688**
- Lowest number of coefficients **(10)** makes this model easy to interpret and computationally efficient.

Following are all the predictors for the model `cahoudta_add_bckwd_bic_model` with test statistics and p-values.

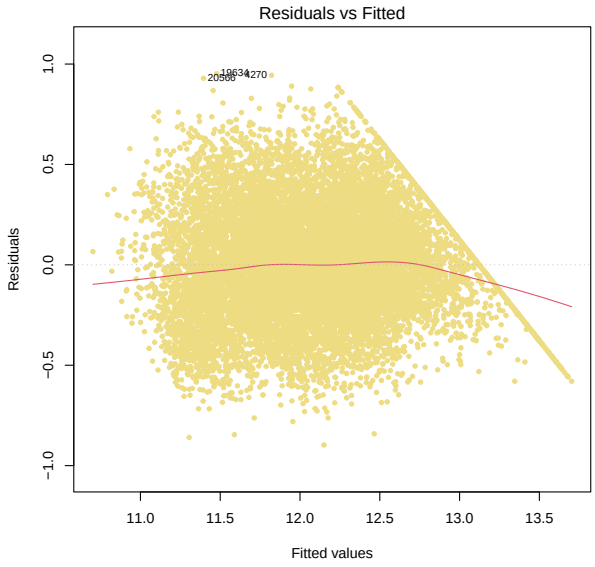
```
tidy(cahoudta_add_bckwd_bic_model)
```

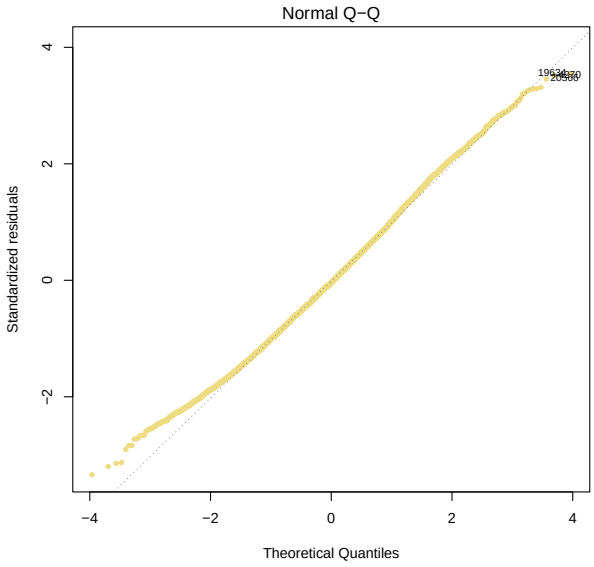
```
## # A tibble: 10 x 5
##    term                       estimate  std.error statistic   p.value
##    <chr>                         <dbl>      <dbl>     <dbl>     <dbl>
##  1 (Intercept)               -2.45     0.439          -5.57 2.57e-  8
##  2 longitude                 -0.161    0.00513       -31.4  8.15e-209
##  3 latitude                  -0.152    0.00514       -29.6  4.14e-187
##  4 housing_median_age         0.00301  0.000214       14.1  1.44e- 44
##  5 population                -0.000284 0.00000620    -45.8  0
##  6 households                 0.000925 0.0000180      51.4  0
##  7 median_income              0.179    0.00142       126.   0
##  8 ocean_proximityINLAND     -0.333    0.00864       -38.6  3.98e-309
##  9 ocean_proximityNEAR BAY   -0.0873   0.00914        -9.55 1.50e- 21
## 10 ocean_proximityNEAR OCEAN -0.0297   0.00772        -3.85 1.21e-  4
```

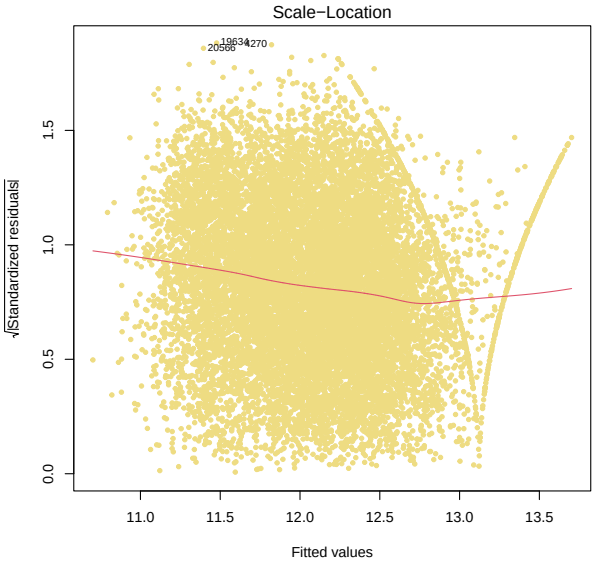
We will also run all the standard diagnostics on this model.

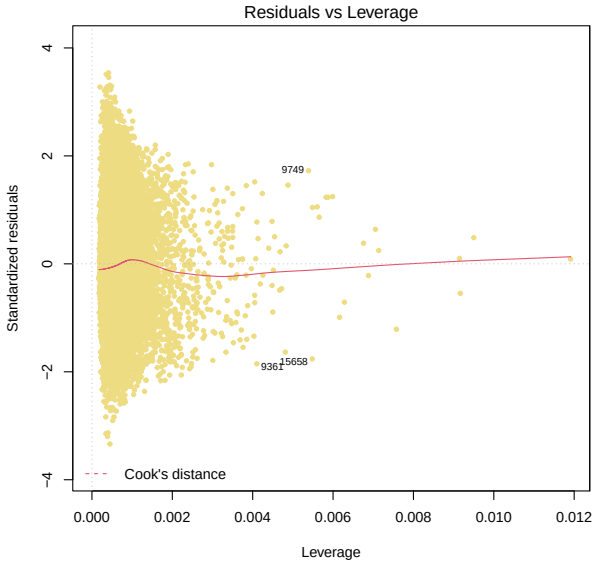
```
par(mfrow=c(2, 2))

plot(cahoudta_add_bckwd_bic_model,
     pch = 20,
     col = "lightgoldenrod2")
```


Residuals vs Fitted
Residuals
Fitted values
1.0
0.5
0.0
−0.5
−1.0
11.0
11.5
12.0
12.5
13.0
13.5


Normal Q−Q
Standardized residuals
Theoretical Quantiles
4
2
0
−2
−4


Scale−Location
√|Standardized residuals|
Fitted values
1.5
1.0
0.5
0.0
11.0
11.5
12.0
12.5
13.0
13.5


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance
9749
15658
4
2
0
−2
−4
0.000
0.002
0.004
0.006
0.008
0.010
0.012