DATA 621: BUSINESS ANALYTICS AND DATA MINING
HOMEWORK#3: LOGISTIC REGRESSION

Group 2 - Gabriel Campos, Melissa Bowman, Alexander Khaykin, & Jennifer Abinette

Last edited November 07, 2023

Contents

1 Overview

In this homework assignment, you will explore, analyze and model a data set containing information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or not (0).

Your objective is to build a *binary logistic regression model* on the training data set to predict whether the neighborhood will be at risk for high crime levels. You will provide classifications and probabilities for the evaluation data set using your binary logistic regression model. You can only use the variables given to you (or, variables that you derive from the variables provided). Below is a short description of the variables of interest in the data set:

- zn: proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable)

- indus: proportion of non-retail business acres per suburb (predictor variable)
- chas: a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable)
- nox: nitrogen oxides concentration (parts per 10 million) (predictor variable)
- rm: average number of rooms per dwelling (predictor variable)
- age: proportion of owner-occupied units built prior to 1940 (predictor variable)
- dis: weighted mean of distances to five Boston employment centers (predictor variable)
- rad: index of accessibility to radial highways (predictor variable)
- tax: full-value property-tax rate per $10,000 (predictor variable)
- ptratio: pupil-teacher ratio by town (predictor variable)
- lstat: lower status of the population (percent) (predictor variable)
- medv: median value of owner-occupied homes in $1000s (predictor variable)
- **target: whether the crime rate is above the median crime rate (1) or not (0) (response variable)**

1.1 Deliverables:

- A write-up submitted in PDF format. Your write-up should have four sections. Each one is described below. You may assume you are addressing me as a fellow data scientist, so do not need to shy away from technical details.
- Assigned prediction (probabilities, classifications) for the evaluation data set. Use 0.5 threshold. Include your R statistical programming code in an Appendix.

1.2 Write Up:

1. DATA EXPLORATION (25 Points) Describe the size and the variables in the crime training data set. Consider that too much detail will cause a manager to lose interest while too little detail will make the manager consider that you aren't doing your job. Some suggestions are given below. Please do NOT treat this as a check list of things to do to complete the assignment. You should have your own thoughts on what to tell the boss. These are just ideas. a. Mean / Standard Deviation / Median b. Bar Chart or Box Plot of the data c. Is the data correlated to the target variable (or to other variables?) d. Are any of the variables missing and need to be imputed/"fixed"?

2. DATA PREPARATION (25 Points) Describe how you have transformed the data by changing the original variables or creating new variables. If you did transform the data or create new variables, discuss why you did this. Here are some possible transformations. a. Fix missing values (maybe with a Mean or Median value) b. Create flags to suggest if a variable was missing c. Transform data by putting it into buckets d. Mathematical transforms such as log or square root (or, use Box-Cox) e. Combine variables (such as ratios or adding or multiplying) to create new variables

3. BUILD MODELS (25 Points) Using the training data, build at least three different binary logistic regression models, using different variables (or the same variables with different transformations). You may select the variables manually, use an approach such as Forward or Stepwise, use a different approach, or use a combination of techniques. Describe the techniques you used. If you manually selected a variable for inclusion into the model or exclusion into the model, indicate why this was done. Be sure to explain how you can make inferences from the model, as well as discuss other relevant model output. Discuss the coefficients in the models, do they make sense? Are you keeping the model even though it is counter intuitive? Why? The boss needs to know.

4. SELECT MODELS (25 Points) Decide on the criteria for selecting the best binary logistic regression model. Will you select models with slightly worse performance if it makes more sense or is more parsimonious? Discuss why you selected your model. * For the binary logistic regression model, will you use a metric such as log likelihood, AIC, ROC curve, etc.? Using the training data set, evaluate the binary logistic regression model based on (a) accuracy, (b) classification error rate, (c) precision, (d) sensitivity, (e) specificity, (f) F1 score, (g) AUC, and (h) confusion matrix. Make predictions using the evaluation data set

2 Data Exploration

2.1 Load the data

```
df_crime_eval <-
  read.csv(paste0(git_url,"crime-evaluation-data_modified.csv"))
```

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7
0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	10.26	18.2
0	8.14	0	0.538	6.495	94.4	4.4547	4	307	21.0	12.80	18.4
0	8.14	0	0.538	5.950	82.0	3.9900	4	307	21.0	27.71	13.2
0	5.96	0	0.499	5.850	41.5	3.9342	5	279	19.2	8.77	21.0
25	5.13	0	0.453	5.741	66.2	7.2254	8	284	19.7	13.15	18.7
25	5.13	0	0.453	5.966	93.4	6.8185	8	284	19.7	14.44	16.0
0	4.49	0	0.449	6.630	56.1	4.4377	3	247	18.5	6.53	26.6
0	4.49	0	0.449	6.121	56.8	3.7476	3	247	18.5	8.44	22.2
0	2.89	0	0.445	6.163	69.6	3.4952	2	276	18.0	11.34	21.4

```
df_crime_train <-
  read.csv(paste0(git_url,"crime-training-data_modified.csv"))
```

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	19.58	0	0.605	7.929	96.2	2.0459	5	403	14.7	3.70	50.0	1
0	19.58	1	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4	1
0	18.10	0	0.740	6.485	100.0	1.9784	24	666	20.2	18.85	15.4	1
30	4.93	0	0.428	6.393	7.8	7.0355	6	300	16.6	5.19	23.7	0
0	2.46	0	0.488	7.155	92.2	2.7006	3	193	17.8	4.82	37.9	0
0	8.56	0	0.520	6.781	71.3	2.8561	5	384	20.9	7.67	26.5	0
0	18.10	0	0.693	5.453	100.0	1.4896	24	666	20.2	30.59	5.0	1
0	18.10	0	0.693	4.519	100.0	1.6582	24	666	20.2	36.98	7.0	1
0	5.19	0	0.515	6.316	38.1	6.4584	5	224	20.2	5.68	22.2	0
80	3.64	0	0.392	5.876	19.1	9.2203	1	315	16.4	9.25	20.9	0

2.2 Remove NA's

```
df_crime_eval[is.na(df_crime_eval)]
```

```
## numeric(0)
```

```
df_crime_train[is.na(df_crime_train)]
```

```
## numeric(0)
```

2.3 Summaries

```r
summary(df_crime_eval)
```

```
##        zn             indus             chas           nox
##  Min.   : 0.000   Min.   : 1.760   Min.   :0.00   Min.   :0.3850
##  1st Qu.: 0.000   1st Qu.: 5.692   1st Qu.:0.00   1st Qu.:0.4713
##  Median : 0.000   Median : 8.915   Median :0.00   Median :0.5380
##  Mean   : 8.875   Mean   :11.507   Mean   :0.05   Mean   :0.5592
##  3rd Qu.: 0.000   3rd Qu.:18.100   3rd Qu.:0.00   3rd Qu.:0.6258
##  Max.   :90.000   Max.   :25.650   Max.   :1.00   Max.   :0.7400
##        rm             age              dis             rad
##  Min.   :3.561   Min.   :  6.80   Min.   :1.202   Min.   : 1.000
##  1st Qu.:5.874   1st Qu.: 56.62   1st Qu.:2.041   1st Qu.: 4.000
##  Median :6.143   Median : 83.25   Median :3.373   Median : 5.000
##  Mean   :6.214   Mean   : 70.99   Mean   :3.787   Mean   : 9.775
##  3rd Qu.:6.532   3rd Qu.: 93.10   3rd Qu.:4.527   3rd Qu.:24.000
##  Max.   :8.247   Max.   :100.00   Max.   :9.089   Max.   :24.000
##       tax           ptratio         lstat             medv
##  Min.   :188.0   Min.   :14.70   Min.   : 2.960   Min.   : 8.40
##  1st Qu.:276.8   1st Qu.:18.40   1st Qu.: 6.435   1st Qu.:16.98
##  Median :307.0   Median :19.60   Median :11.685   Median :20.55
##  Mean   :393.5   Mean   :19.12   Mean   :12.905   Mean   :21.88
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:17.363   3rd Qu.:25.00
##  Max.   :666.0   Max.   :21.20   Max.   :34.020   Max.   :50.00
```

```r
summary(df_crime_train)
```

```
##        zn             indus             chas              nox
##  Min.   :  0.00   Min.   : 0.460   Min.   :0.00000   Min.   :0.3890
##  1st Qu.:  0.00   1st Qu.: 5.145   1st Qu.:0.00000   1st Qu.:0.4480
##  Median :  0.00   Median : 9.690   Median :0.00000   Median :0.5380
##  Mean   : 11.58   Mean   :11.105   Mean   :0.07082   Mean   :0.5543
##  3rd Qu.: 16.25   3rd Qu.:18.100   3rd Qu.:0.00000   3rd Qu.:0.6240
##  Max.   :100.00   Max.   :27.740   Max.   :1.00000   Max.   :0.8710
##        rm             age              dis               rad
##  Min.   :3.863   Min.   :  2.90   Min.   : 1.130   Min.   : 1.00
##  1st Qu.:5.887   1st Qu.: 43.88   1st Qu.: 2.101   1st Qu.: 4.00
##  Median :6.210   Median : 77.15   Median : 3.191   Median : 5.00
##  Mean   :6.291   Mean   : 68.37   Mean   : 3.796   Mean   : 9.53
##  3rd Qu.:6.630   3rd Qu.: 94.10   3rd Qu.: 5.215   3rd Qu.:24.00
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.00
##       tax           ptratio         lstat             medv
##  Min.   :187.0   Min.   :12.6   Min.   : 1.730   Min.   : 5.00
##  1st Qu.:281.0   1st Qu.:16.9   1st Qu.: 7.043   1st Qu.:17.02
##  Median :334.5   Median :18.9   Median :11.350   Median :21.20
##  Mean   :409.5   Mean   :18.4   Mean   :12.631   Mean   :22.59
##  3rd Qu.:666.0   3rd Qu.:20.2   3rd Qu.:16.930   3rd Qu.:25.00
##  Max.   :711.0   Max.   :22.0   Max.   :37.970   Max.   :50.00
##      target
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4914
```

```
##   3rd Qu.:1.0000
##   Max.   :1.0000
```

2.4 Inspecting for Categorical Variables

```r
str(df_crime_train)
```

```
## 'data.frame':    466 obs. of  13 variables:
##  $ zn     : num  0 0 0 30 0 0 0 0 0 80 ...
##  $ indus  : num  19.58 19.58 18.1 4.93 2.46 ...
##  $ chas   : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.605 0.871 0.74 0.428 0.488 0.52 0.693 0.693 0.515 0.392 ...
##  $ rm     : num  7.93 5.4 6.49 6.39 7.16 ...
##  $ age    : num  96.2 100 100 7.8 92.2 71.3 100 100 38.1 19.1 ...
##  $ dis    : num  2.05 1.32 1.98 7.04 2.7 ...
##  $ rad    : int  5 5 24 6 3 5 24 24 5 1 ...
##  $ tax    : int  403 403 666 300 193 384 666 666 224 315 ...
##  $ ptratio: num  14.7 14.7 20.2 16.6 17.8 20.9 20.2 20.2 20.2 16.4 ...
##  $ lstat  : num  3.7 26.82 18.85 5.19 4.82 ...
##  $ medv   : num  50 13.4 15.4 23.7 37.9 26.5 5 7 22.2 20.9 ...
##  $ target : int  1 1 1 0 0 0 1 1 0 0 ...
```

2.5 Contingecny Tables / Frequency

```r
(xtabs(~ target + medv ,  data = df_crime_train))
```

```
##       medv
## target  5 5.6 6.3  7 7.2 7.4 7.5 8.1 8.3 8.4 8.5 8.7 8.8 9.5 9.6 9.7 10.2 10.4
##      0  0   0   0  1   0   0   0   1   0   0   0   0   0   0   0   0    0    0
##      1  2   1   1  1   3   1   1   0   2   1   2   1   2   1   1   1    3    2
##       medv
## target 10.5 10.9 11 11.3 11.5 11.7 11.8 11.9 12 12.1 12.3 12.5 12.6 12.7 13
##      0    0    0  0    0    0    0    0    1  0    0    0    0    0    0  0
##      1    1    1  2    1    1    1    2    2  1    1    1    1    1    1  3  1
##       medv
## target 13.1 13.3 13.4 13.5 13.6 13.8 13.9 14 14.1 14.2 14.3 14.4 14.5 14.6 14.8
##      0    0    0    1    0    0    1    0  0    0    0    0    1    0    0    0
##      1    1    4    2    4    2    1    5  2    1    2    1    1    1    2    2    1
##       medv
## target 14.9 15 15.1 15.2 15.3 15.4 15.6 16.1 16.2 16.3 16.4 16.5 16.6 16.7 16.8
##      0    0  0    1    0    1    0    0    0    0    1    0    0    2    1    0    1
##      1    1  3    2    1    2    1    2    5    3    1    1    1    0    1    2    1
##       medv
## target 17 17.1 17.2 17.4 17.5 17.6 17.8 17.9 18 18.1 18.2 18.3 18.4 18.5 18.6
##      0  0    1    1    1    2    1    0    0  0    0    1    1    0    2    2
##      1  1    2    2    2    1    0    5    1  1    1    1    0    1    1    0
##       medv
## target 18.7 18.8 18.9 19 19.1 19.2 19.3 19.4 19.5 19.6 19.7 19.8 19.9 20 20.1
##      0    2    2    4  1    0    1    4    3    3    1    1    3    0  3    3
```

```
##      1    0    0    0  1    3    1    1    3    1    4    1    0    4  2    1
##       medv
## target 20.2 20.3 20.4 20.5 20.6 20.7 20.8 20.9 21 21.1 21.2 21.4 21.5 21.6 21.7
##      0    0    3    2    3    4    2    1    2  0    1    4    2    1    1    4
##      1    2    1    2    0    2    0    2    0  2    1    1    1    1    1    2
##       medv
## target 21.8 21.9 22 22.2 22.3 22.4 22.5 22.6 22.7 22.8 22.9 23 23.1 23.2 23.3
##      0    1    1  7    4    1    2    3    4    0    2    4  1    4    2    3
##      1    1    2  0    0    1    0    0    1    2    2    0  2    3    2    1
##       medv
## target 23.4 23.5 23.6 23.7 23.8 23.9 24 24.1 24.2 24.3 24.4 24.5 24.6 24.7 24.8
##      0    2    1    2    2    1    4  1    3    1    0    4    1    2    3    4
##      1    0    0    0    2    3    0  1    0    0    3    0    1    0    0    0
##       medv
## target 25 25.1 25.2 25.3 26.2 26.4 26.5 26.6 26.7 27 27.1 27.5 27.9 28 28.1
##      0  4    0    1    1    1    2    1    2    0  0    2    1    1  1    1
##      1  2    1    0    0    0    0    0    0    1  1    0    2    1  0    0
##       medv
## target 28.2 28.4 28.5 28.6 28.7 29 29.1 29.4 29.6 29.8 29.9 30.1 30.3 30.5 30.7
##      0    1    1    1    1    3  1    2    1    1    1    1    1    1    1    0
##      1    0    0    0    0    0  1    0    0    0    1    0    2    0    0    1
##       medv
## target 30.8 31 31.1 31.2 31.5 31.6 31.7 32 32.4 32.5 32.7 32.9 33 33.1 33.2
##      0    1  0    1    1    0    1    0  2    1    1    1    1  1    2    2
##      1    0  1    0    0    1    1    1  0    0    0    0    0  0    0    0
##       medv
## target 33.3 33.4 33.8 34.6 34.9 35.1 35.2 35.4 36 36.1 36.2 36.4 36.5 37 37.2
##      0    1    2    0    1    3    1    1    2  0    1    2    1    0  1    1
##      1    0    0    1    0    0    0    0    0  1    0    0    0    1  0    0
##       medv
## target 37.3 37.6 37.9 38.7 39.8 41.3 41.7 42.3 42.8 43.1 43.5 43.8 44 44.8 45.4
##      0    1    0    1    1    1    0    0    1    0    0    0    1  1    0    1
##      1    0    1    0    0    0    1    1    0    1    1    1    0  0    1    0
##       medv
## target 46 46.7 48.5 48.8 50
##      0  1    0    1    0  4
##      1  0    1    0    1 11
```

2.6 Correlation Matrix

```r
# Create a correlation matrix for all variables
(cor_matrix <- cor(df_crime_train))
```

```
##                 zn       indus        chas         nox          rm         age
## zn       1.00000000 -0.53826643 -0.04016203 -0.51704518  0.31981410 -0.57258054
## indus   -0.53826643  1.00000000  0.06118317  0.75963008 -0.39271181  0.63958182
## chas    -0.04016203  0.06118317  1.00000000  0.09745577  0.09050979  0.07888366
## nox     -0.51704518  0.75963008  0.09745577  1.00000000 -0.29548972  0.73512782
## rm       0.31981410 -0.39271181  0.09050979 -0.29548972  1.00000000 -0.23281251
## age     -0.57258054  0.63958182  0.07888366  0.73512782 -0.23281251  1.00000000
## dis      0.66012434 -0.70361886 -0.09657711 -0.76888404  0.19901584 -0.75089759
## rad     -0.31548119  0.60062839 -0.01590037  0.59582984 -0.20844570  0.46031430
```

```
## tax     -0.31928408  0.73222922 -0.04676476  0.65387804 -0.29693430  0.51212452
## ptratio -0.39103573  0.39468980 -0.12866058  0.17626871 -0.36034706  0.25544785
## lstat   -0.43299252  0.60711023 -0.05142322  0.59624264 -0.63202445  0.60562001
## medv     0.37671713 -0.49617432  0.16156528 -0.43012267  0.70533679 -0.37815605
## target  -0.43168176  0.60485074  0.08004187  0.72610622 -0.15255334  0.63010625
##                 dis         rad         tax    ptratio       lstat       medv
## zn       0.66012434 -0.31548119 -0.31928408 -0.3910357 -0.43299252  0.3767171
## indus   -0.70361886  0.60062839  0.73222922  0.3946898  0.60711023 -0.4961743
## chas    -0.09657711 -0.01590037 -0.04676476 -0.1286606 -0.05142322  0.1615653
## nox     -0.76888404  0.59582984  0.65387804  0.1762687  0.59624264 -0.4301227
## rm       0.19901584 -0.20844570 -0.29693430 -0.3603471 -0.63202445  0.7053368
## age     -0.75089759  0.46031430  0.51212452  0.2554479  0.60562001 -0.3781560
## dis      1.00000000 -0.49499193 -0.53425464 -0.2333394 -0.50752800  0.2566948
## rad     -0.49499193  1.00000000  0.90646323  0.4714516  0.50310125 -0.3976683
## tax     -0.53425464  0.90646323  1.00000000  0.4744223  0.56418864 -0.4900329
## ptratio -0.23333940  0.47145160  0.47442229  1.0000000  0.37735605 -0.5159153
## lstat   -0.50752800  0.50310125  0.56418864  0.3773560  1.00000000 -0.7358008
## medv     0.25669476 -0.39766826 -0.49003287 -0.5159153 -0.73580078  1.0000000
## target  -0.61867312  0.62810492  0.61111331  0.2508489  0.46912702 -0.2705507
##              target
## zn      -0.43168176
## indus    0.60485074
## chas     0.08004187
## nox      0.72610622
## rm      -0.15255334
## age      0.63010625
## dis     -0.61867312
## rad      0.62810492
## tax      0.61111331
## ptratio  0.25084892
## lstat    0.46912702
## medv    -0.27055071
## target   1.00000000
```

The logistic regression model dependent variable target has

Changing categorical data into factors to ensure that the model can appropriately interpret and analyze categorical variables.Change after looking at col linearity.

```r
#Don't use this
#df_crime_train$chas <- as.factor(df_crime_train$chas)
#df_crime_train$rad <- as.factor(df_crime_train$rad)
#df_crime_train$target <- as.factor(df_crime_train$target)
```

2.7 Training Data Structure

```r
str(df_crime_train)
```

```
## 'data.frame':    466 obs. of  13 variables:
##  $ zn     : num  0 0 0 30 0 0 0 0 0 80 ...
##  $ indus  : num  19.58 19.58 18.1 4.93 2.46 ...
##  $ chas   : int  0 1 0 0 0 0 0 0 0 0 ...
```

```
##  $ nox    : num  0.605 0.871 0.74 0.428 0.488 0.52 0.693 0.693 0.515 0.392 ...
##  $ rm     : num  7.93 5.4 6.49 6.39 7.16 ...
##  $ age    : num  96.2 100 100 7.8 92.2 71.3 100 100 38.1 19.1 ...
##  $ dis    : num  2.05 1.32 1.98 7.04 2.7 ...
##  $ rad    : int  5 5 24 6 3 5 24 24 5 1 ...
##  $ tax    : int  403 403 666 300 193 384 666 666 224 315 ...
##  $ ptratio: num  14.7 14.7 20.2 16.6 17.8 20.9 20.2 20.2 20.2 16.4 ...
##  $ lstat  : num  3.7 26.82 18.85 5.19 4.82 ...
##  $ medv   : num  50 13.4 15.4 23.7 37.9 26.5 5 7 22.2 20.9 ...
##  $ target : int  1 1 1 0 0 0 1 1 0 0 ...
```

2.8 Model 1

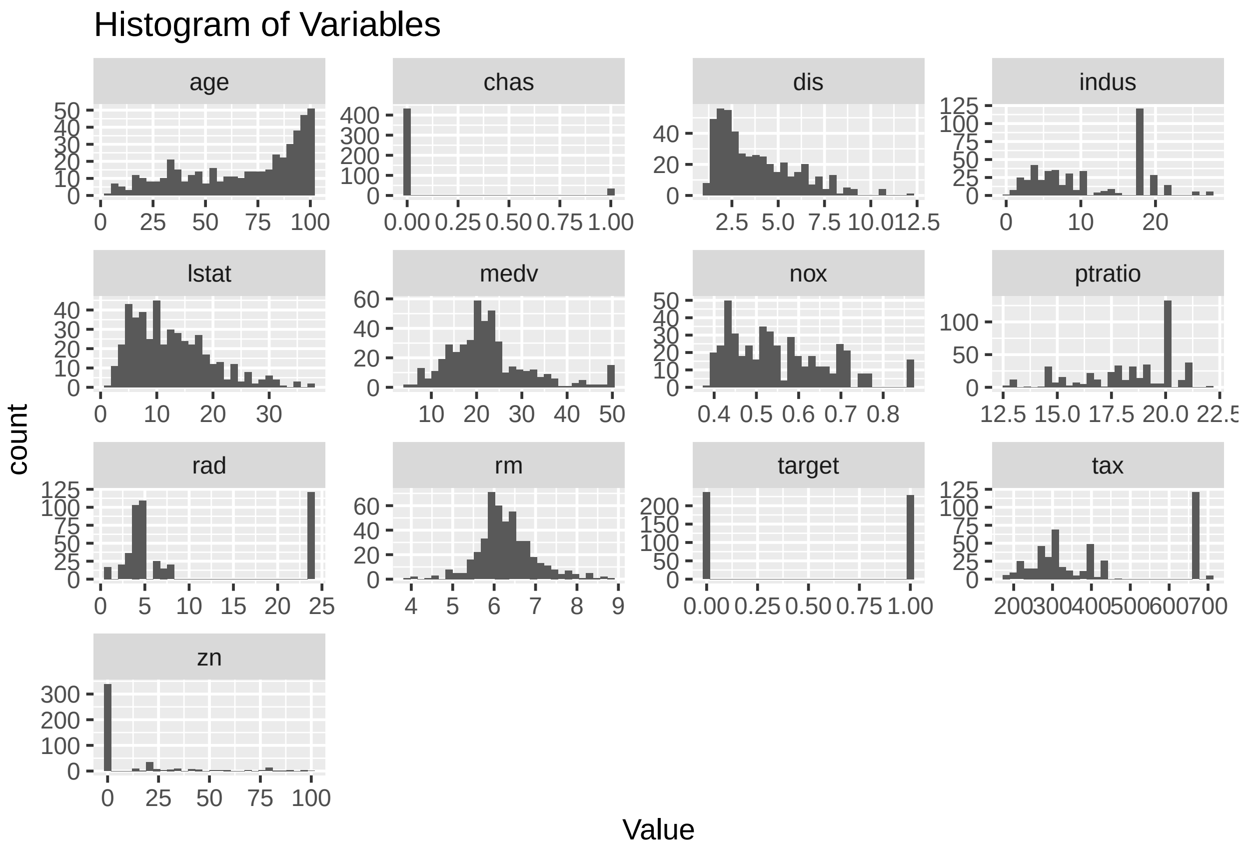
```
model_1 <- glm(formula = target ~ ., family = binomial, data = df_crime_train)

summary(model_1)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = df_crime_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -40.822934   6.632913  -6.155 7.53e-10 ***
## zn           -0.065946   0.034656  -1.903  0.05706 .
## indus        -0.064614   0.047622  -1.357  0.17485
## chas          0.910765   0.755546   1.205  0.22803
## nox          49.122297   7.931706   6.193 5.90e-10 ***
## rm           -0.587488   0.722847  -0.813  0.41637
## age           0.034189   0.013814   2.475  0.01333 *
## dis           0.738660   0.230275   3.208  0.00134 **
## rad           0.666366   0.163152   4.084 4.42e-05 ***
## tax          -0.006171   0.002955  -2.089  0.03674 *
## ptratio       0.402566   0.126627   3.179  0.00148 **
## lstat         0.045869   0.054049   0.849  0.39608
## medv          0.180824   0.068294   2.648  0.00810 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 192.05  on 453  degrees of freedom
## AIC: 218.05
##
## Number of Fisher Scoring iterations: 9
```

```
# Gather the data into a long format
data_long <- gather(df_crime_train, key = "Variable", value = "Value")

ggplot(data_long, aes(x = Value)) +
```

```r
  geom_histogram() +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Histogram of Variables")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

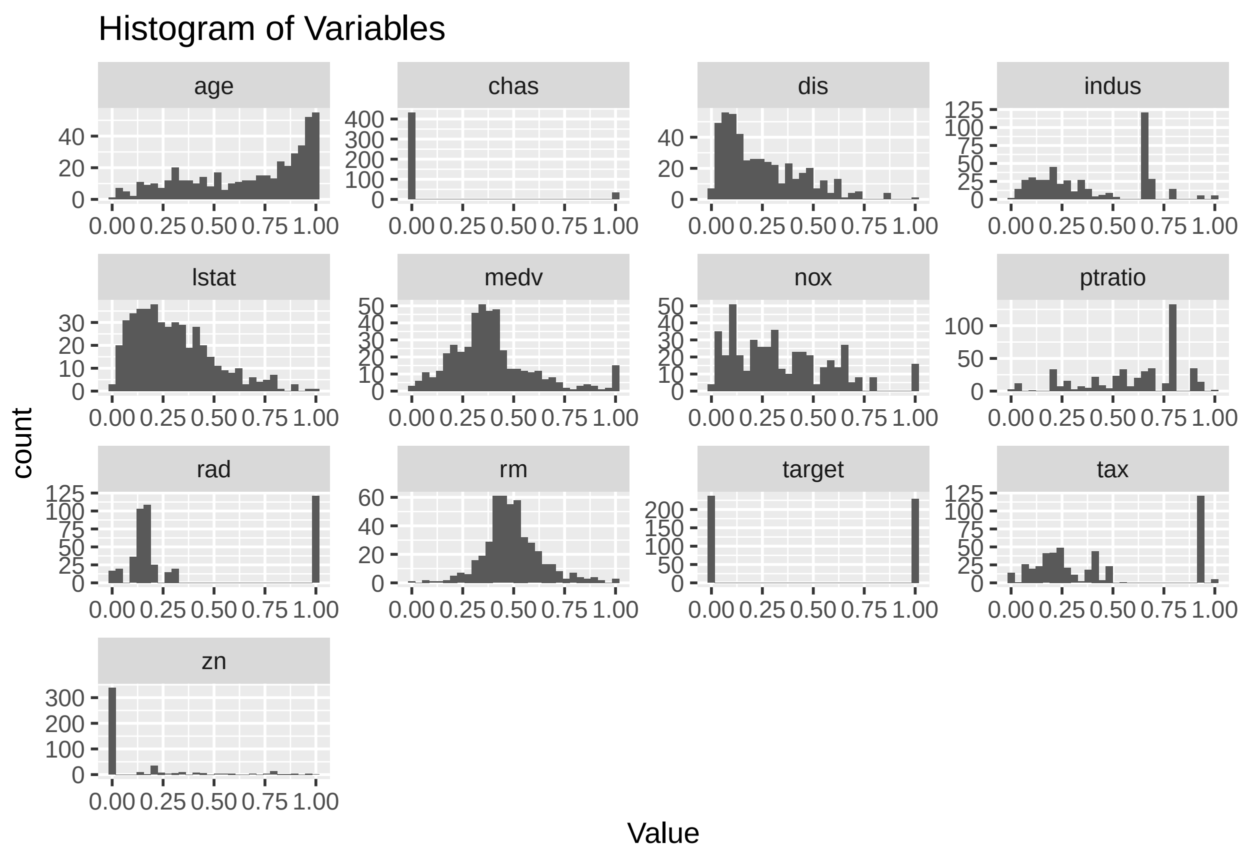


Visually it is apparent that not all the data is normalized. To assist with interpreting, attributes are re scaled to 0.00, 0.25, 0.50, 0.75 & 1.00

```r
# Apply min-max scaling to all three variables
data_scaled <- df_crime_train
data_scaled[] <- lapply(df_crime_train, rescale)
```

```r
# Gather the data into a long format
data_long_scaled <- gather(data_scaled, key = "Variable", value = "Value")

ggplot(data_long_scaled, aes(x = Value)) +
  geom_histogram() +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Histogram of Variables")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.9 Merged Model

```
# Add new columns and rename them
df_merged<-(bind_cols(df_crime_train,
                (df_merged <- data_scaled %>%
                   mutate(zn_t = zn,indus_t = indus,chas_t=chas,nox_t=nox,
                          rm_t=rm,age_t=age,dis_t=dis,rad_t=rad,tax_t=tax,
                          ptratio_t=ptratio,lstat_t=lstat,medv_t=medv,
                          target_t=target))%>%dplyr::select(ends_with("_t"))
                )%>%
         dplyr::select(zn,zn_t,indus,indus_t,chas,chas_t,nox,nox_t,rm,rm_t,
                       age,age_t,dis,dis_t,rad,rad_t,tax,tax_t,ptratio,
                       ptratio_t,lstat,lstat_t,medv,medv_t,target,target_t)
)

# Display the updated dataframe
df_merged%>%
  kable()%>%
  kable_styling()
```

Checking correlation of scaled varibles

zn	zn_t	indus	indus_t	chas	chas_t	nox	nox_t	rm	rm_t	age	age_t	d
0.0	0.000	19.58	0.7008798	0	0	0.6050	0.4481328	7.929	0.8269270	96.2	0.9608651	2.045
0.0	0.000	19.58	0.7008798	1	1	0.8710	1.0000000	5.403	0.3131991	100.0	1.0000000	1.321
0.0	0.000	18.10	0.6466276	0	0	0.7400	0.7282158	6.485	0.5332520	100.0	1.0000000	1.978
30.0	0.300	4.93	0.1638563	0	0	0.4280	0.0809129	6.393	0.5145414	7.8	0.0504634	7.035
0.0	0.000	2.46	0.0733138	0	0	0.4880	0.2053942	7.155	0.6695139	92.2	0.9196704	2.700
0.0	0.000	8.56	0.2969208	0	0	0.5200	0.2717842	6.781	0.5934513	71.3	0.7044284	2.856
0.0	0.000	18.10	0.6466276	0	0	0.6930	0.6307054	5.453	0.3233679	100.0	1.0000000	1.489
0.0	0.000	18.10	0.6466276	0	0	0.6930	0.6307054	4.519	0.1334147	100.0	1.0000000	1.658
0.0	0.000	5.19	0.1733871	0	0	0.5150	0.2614108	6.316	0.4988814	38.1	0.3625129	6.458
80.0	0.800	3.64	0.1165689	0	0	0.3920	0.0062241	5.876	0.4093960	19.1	0.1668383	9.220
22.0	0.220	5.86	0.1979472	0	0	0.4310	0.0871369	6.438	0.5236933	8.9	0.0617920	7.396
0.0	0.000	12.83	0.4534457	0	0	0.4370	0.0995851	6.286	0.4927802	45.0	0.4335736	4.502
0.0	0.000	18.10	0.6466276	0	0	0.5320	0.2966805	7.061	0.6503966	77.0	0.7631308	3.410
22.0	0.220	5.86	0.1979472	0	0	0.4310	0.0871369	8.259	0.8940411	8.4	0.0566426	8.906
0.0	0.000	2.46	0.0733138	0	0	0.4880	0.2053942	6.153	0.4657311	68.8	0.6786818	3.279
0.0	0.000	2.18	0.0630499	0	0	0.4580	0.1431535	6.430	0.5220663	58.7	0.5746653	6.062
100.0	1.000	1.32	0.0315249	0	0	0.4110	0.0456432	6.816	0.6005695	40.5	0.3872297	8.324
20.0	0.200	3.97	0.1286657	0	0	0.6470	0.5352697	5.560	0.3451291	62.8	0.6168898	1.986
0.0	0.000	18.10	0.6466276	0	0	0.6790	0.6016598	5.896	0.4134635	95.4	0.9526262	1.909
0.0	0.000	18.10	0.6466276	0	0	0.6710	0.5850622	6.545	0.5454545	99.1	0.9907312	1.519
0.0	0.000	3.24	0.1019062	0	0	0.4600	0.1473029	6.144	0.4639008	32.2	0.3017508	5.873
0.0	0.000	6.20	0.2104106	1	1	0.5070	0.2448133	6.726	0.5822656	66.5	0.6549949	3.651
0.0	0.000	2.89	0.0890762	0	0	0.4450	0.1161826	7.416	0.7225951	62.5	0.6138002	3.495
18.0	0.180	2.31	0.0678152	0	0	0.5380	0.3091286	6.575	0.5515558	65.2	0.6416066	4.090
0.0	0.000	9.90	0.3460411	0	0	0.5440	0.3215768	6.382	0.5123043	67.2	0.6622039	3.532
60.0	0.600	2.93	0.0905425	0	0	0.4010	0.0248963	6.604	0.5574537	18.8	0.1637487	6.219
0.0	0.000	5.19	0.1733871	0	0	0.5150	0.2614108	5.985	0.4315640	45.4	0.4376931	4.812
0.0	0.000	18.10	0.6466276	0	0	0.7000	0.6452282	5.390	0.3105552	98.9	0.9886715	1.728
25.0	0.250	4.86	0.1612903	0	0	0.4260	0.0767635	6.167	0.4685784	46.7	0.4510814	5.400
25.0	0.250	5.13	0.1711877	0	0	0.4530	0.1327801	6.456	0.5273541	67.8	0.6683831	7.225
0.0	0.000	6.20	0.2104106	0	0	0.5040	0.2385892	5.981	0.4307505	68.1	0.6714727	3.671
0.0	0.000	8.56	0.2969208	0	0	0.5200	0.2717842	6.474	0.5310148	97.1	0.9701339	2.432
0.0	0.000	2.89	0.0890762	0	0	0.4450	0.1161826	7.820	0.8047590	36.9	0.3501545	3.495
0.0	0.000	18.10	0.6466276	0	0	0.6790	0.6016598	6.380	0.5118975	95.6	0.9546859	1.968
0.0	0.000	5.19	0.1733871	0	0	0.5150	0.2614108	6.059	0.4466138	37.3	0.3542739	4.812
80.0	0.800	4.95	0.1645894	0	0	0.4110	0.0456432	6.630	0.5627415	23.4	0.2111226	5.116
0.0	0.000	2.46	0.0733138	0	0	0.4880	0.2053942	7.831	0.8069961	53.6	0.5221421	3.199
0.0	0.000	18.10	0.6466276	0	0	0.6140	0.4668050	6.185	0.4722392	96.7	0.9660144	2.170
0.0	0.000	4.39	0.1440616	0	0	0.4420	0.1099585	5.898	0.4138702	52.3	0.5087539	8.013
0.0	0.000	19.58	0.7008798	0	0	0.6050	0.4481328	6.402	0.5163718	95.2	0.9505664	2.262
0.0	0.000	3.24	0.1019062	0	0	0.4600	0.1473029	5.868	0.4077690	25.8	0.2358393	5.214
0.0	0.000	18.10	0.6466276	0	0	0.6710	0.5850622	6.649	0.5666057	93.3	0.9309990	1.344
0.0	0.000	4.05	0.1315982	0	0	0.5100	0.2510373	5.859	0.4059386	68.7	0.6776519	2.701
80.0	0.800	1.91	0.0531525	0	0	0.4130	0.0497925	5.663	0.3660769	21.9	0.1956746	10.585
12.5	0.125	7.87	0.2716276	0	0	0.5240	0.2800830	6.009	0.4364450	82.9	0.8238929	6.226
0.0	0.000	6.91	0.2364370	0	0	0.4480	0.1224066	6.030	0.4407159	85.5	0.8506694	5.689
0.0	0.000	18.10	0.6466276	0	0	0.5830	0.4024896	6.312	0.4980679	51.9	0.5046344	3.991
0.0	0.000	9.90	0.3460411	0	0	0.5440	0.3215768	5.782	0.3902786	71.7	0.7085479	4.031
0.0	0.000	18.10	0.6466276	0	0	0.5320	0.2966805	6.229	0.4811877	90.7	0.9042225	3.099
0.0	0.000	8.14	0.2815249	0	0	0.5380	0.3091286	6.674	0.5716901	87.3	0.8692070	4.239
55.0	0.550	2.25	0.0656158	0	0	0.3890	0.0000000	6.453	0.5267439	31.9	0.2986612	7.307
12.5	0.125	7.87	0.2716276	0	0	0.5240	0.2800830	6.004	0.4354281	85.9	0.8547889	6.592
0.0	0.000	5.96	0.2016129	0	0	0.4990	0.2282158	5.933	0.4209884	68.2	0.6725026	3.360
0.0	0.000	1.89	0.0524194	0	0	0.5180	0.2676349	6.540	0.5444377	59.7	0.5849640	6.266
0.0	0.000	21.89	0.7855572	0	0	0.6240	0.4875519	5.693	0.3721782	96.0	0.9588054	1.788
0.0	0.000	10.59	0.3713343	1	1	0.4890	0.2074689	5.344	0.3011999	100.0	1.0000000	3.875
20.0	0.200	3.33	0.1052053	0	0	0.4429	0.1118257	7.820	0.8047590	64.5	0.6343975	4.694

```r
# Create a correlation matrix for all variables
(cor_matrix <- cor(data_scaled))
```

```
##                  zn       indus        chas         nox          rm         age
## zn       1.00000000 -0.53826643 -0.04016203 -0.51704518  0.31981410 -0.57258054
## indus   -0.53826643  1.00000000  0.06118317  0.75963008 -0.39271181  0.63958182
## chas    -0.04016203  0.06118317  1.00000000  0.09745577  0.09050979  0.07888366
## nox     -0.51704518  0.75963008  0.09745577  1.00000000 -0.29548972  0.73512782
## rm       0.31981410 -0.39271181  0.09050979 -0.29548972  1.00000000 -0.23281251
## age     -0.57258054  0.63958182  0.07888366  0.73512782 -0.23281251  1.00000000
## dis      0.66012434 -0.70361886 -0.09657711 -0.76888404  0.19901584 -0.75089759
## rad     -0.31548119  0.60062839 -0.01590037  0.59582984 -0.20844570  0.46031430
## tax     -0.31928408  0.73222922 -0.04676476  0.65387804 -0.29693430  0.51212452
## ptratio -0.39103573  0.39468980 -0.12866058  0.17626871 -0.36034706  0.25544785
## lstat   -0.43299252  0.60711023 -0.05142322  0.59624264 -0.63202445  0.60562001
## medv     0.37671713 -0.49617432  0.16156528 -0.43012267  0.70533679 -0.37815605
## target  -0.43168176  0.60485074  0.08004187  0.72610622 -0.15255334  0.63010625
##                 dis         rad         tax    ptratio       lstat       medv
## zn       0.66012434 -0.31548119 -0.31928408 -0.3910357 -0.43299252  0.3767171
## indus   -0.70361886  0.60062839  0.73222922  0.3946898  0.60711023 -0.4961743
## chas    -0.09657711 -0.01590037 -0.04676476 -0.1286606 -0.05142322  0.1615653
## nox     -0.76888404  0.59582984  0.65387804  0.1762687  0.59624264 -0.4301227
## rm       0.19901584 -0.20844570 -0.29693430 -0.3603471 -0.63202445  0.7053368
## age     -0.75089759  0.46031430  0.51212452  0.2554479  0.60562001 -0.3781560
## dis      1.00000000 -0.49499193 -0.53425464 -0.2333394 -0.50752800  0.2566948
## rad     -0.49499193  1.00000000  0.90646323  0.4714516  0.50310125 -0.3976683
## tax     -0.53425464  0.90646323  1.00000000  0.4744223  0.56418864 -0.4900329
## ptratio -0.23333940  0.47145160  0.47442229  1.0000000  0.37735605 -0.5159153
## lstat   -0.50752800  0.50310125  0.56418864  0.3773560  1.00000000 -0.7358008
## medv     0.25669476 -0.39766826 -0.49003287 -0.5159153 -0.73580078  1.0000000
## target  -0.61867312  0.62810492  0.61111331  0.2508489  0.46912702 -0.2705507
##              target
## zn      -0.43168176
## indus    0.60485074
## chas     0.08004187
## nox      0.72610622
## rm      -0.15255334
## age      0.63010625
## dis     -0.61867312
## rad      0.62810492
## tax      0.61111331
## ptratio  0.25084892
## lstat    0.46912702
## medv    -0.27055071
## target   1.00000000
```

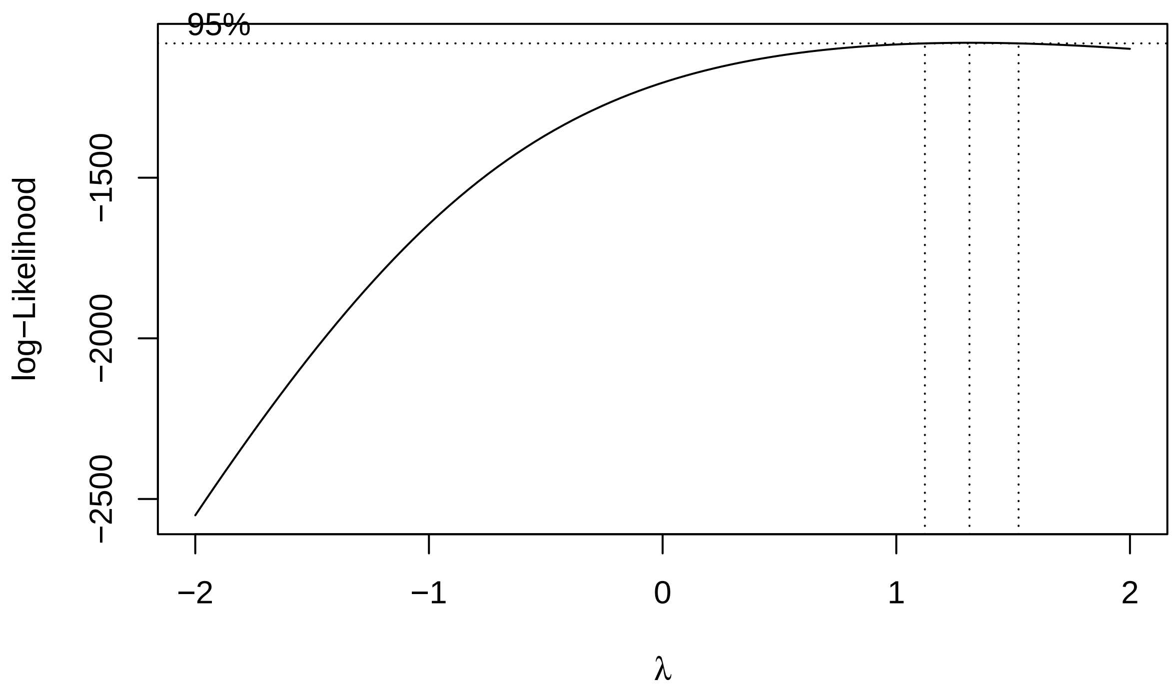
```r
model_2 <- glm(formula = target ~ ., family = binomial, data = data_scaled)

(summary=(model_2))
```

```
##
## Call:  glm(formula = target ~ ., family = binomial, data = data_scaled)
##
```

```
## Coefficients:
## (Intercept)          zn       indus        chas         nox          rm
##    -17.5119     -6.5946     -1.7627      0.9108     23.6769     -2.8887
##         age         dis         rad         tax     ptratio       lstat
##      3.3197      8.1230     15.3264     -3.2338      3.7841      1.6623
##        medv
##      8.1371
##
## Degrees of Freedom: 465 Total (i.e. Null);  453 Residual
## Null Deviance:       645.9
## Residual Deviance: 192   AIC: 218
```

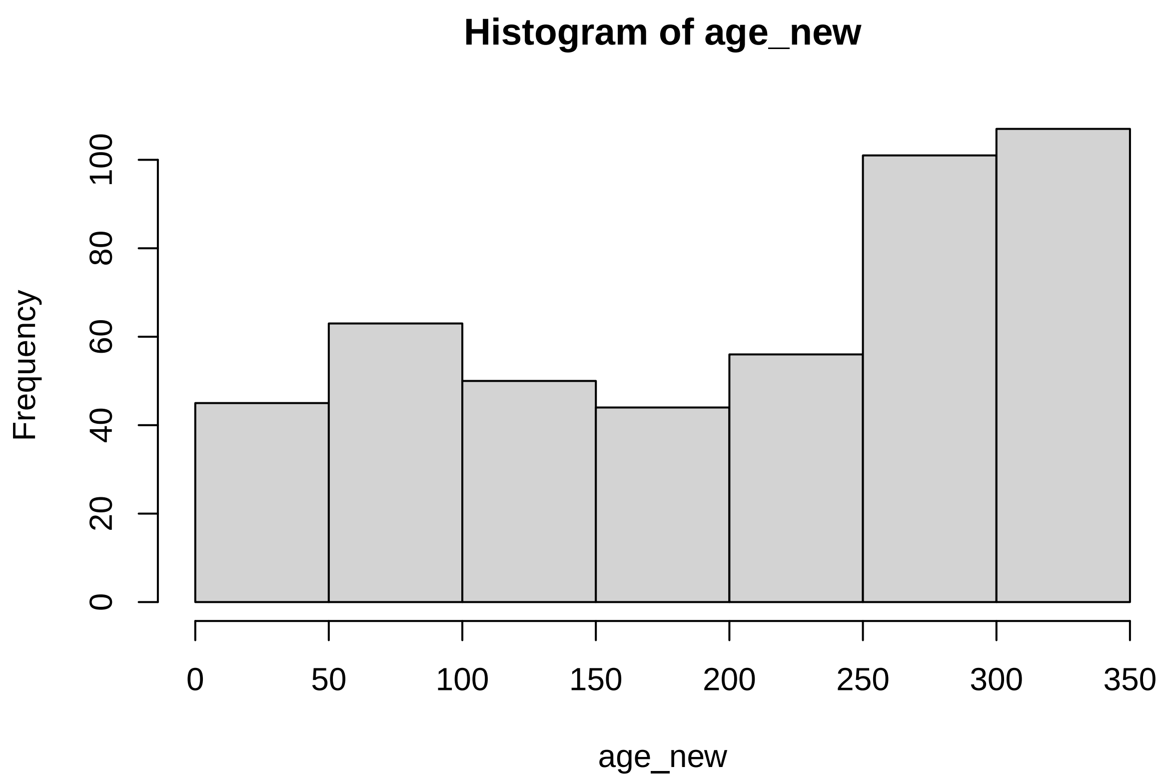
```r
df_crime_train$age <- as.numeric(df_crime_train$age)
```

```r
# Convert a DataFrame column to a list
age_list <- as.numeric(as.list(df_crime_train$age))

#find optimal lambda for Box-Cox transformation
bc <- boxcox(age_list~ 1, lambda = seq(-2,2,0.1))
```

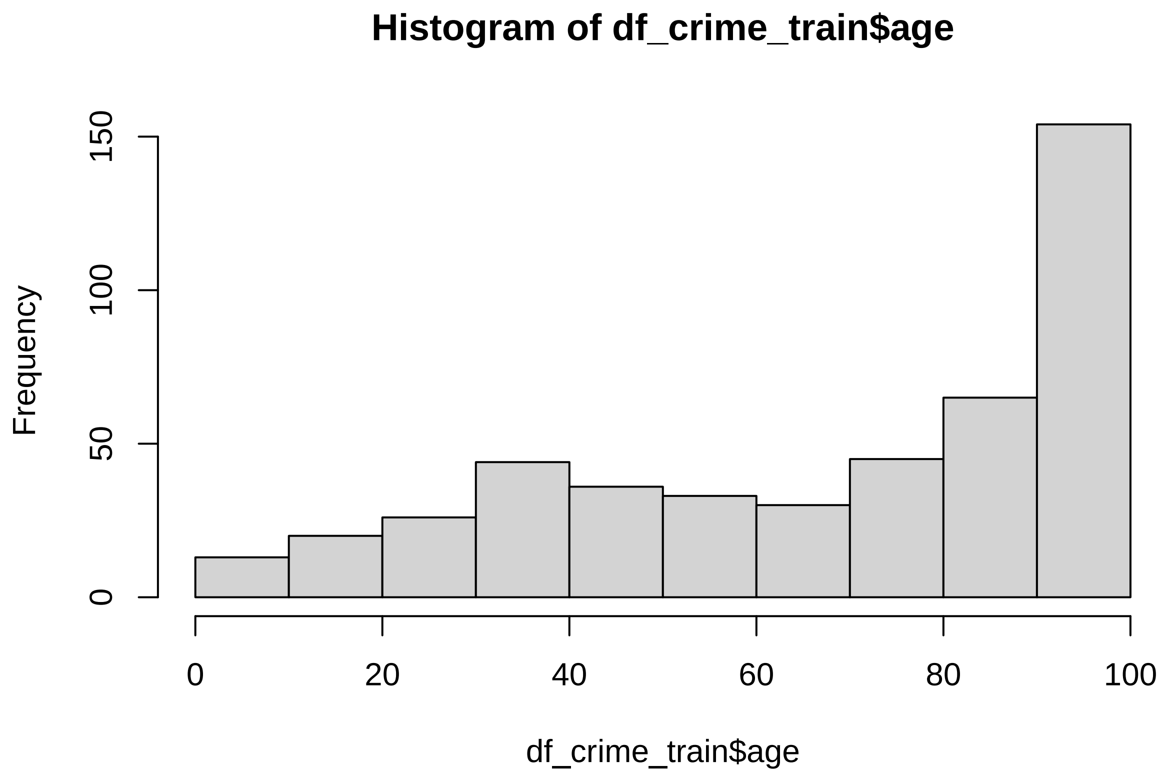



```r
lambda <- bc$x[which.max(bc$y)]

# Apply the Box-Cox transformation
age_new = (age_list^lambda-1)/lambda
```

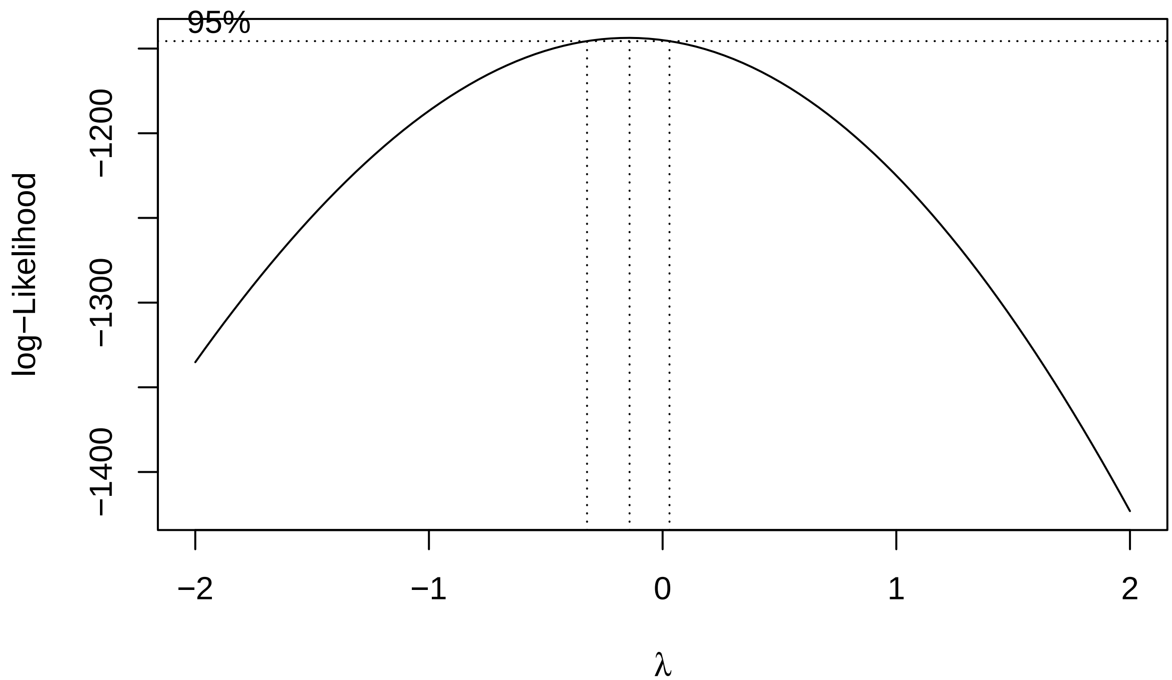
```r
hist(age_new)
```




```r
hist(df_crime_train$age)
```

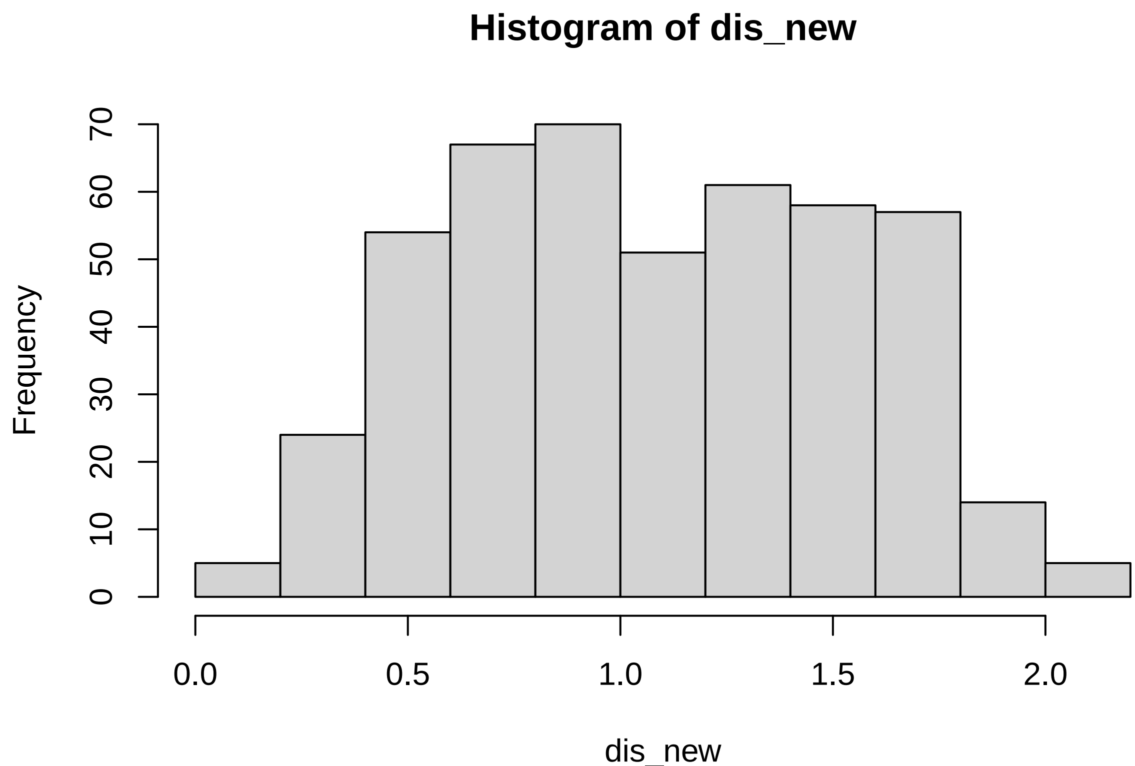



```r
# Convert a DataFrame column to a list
dis_list <- as.numeric(as.list(df_crime_train$dis))

#find optimal lambda for Box-Cox transformation
bc <- boxcox(dis_list~ 1, lambda = seq(-2,2,0.1))
```

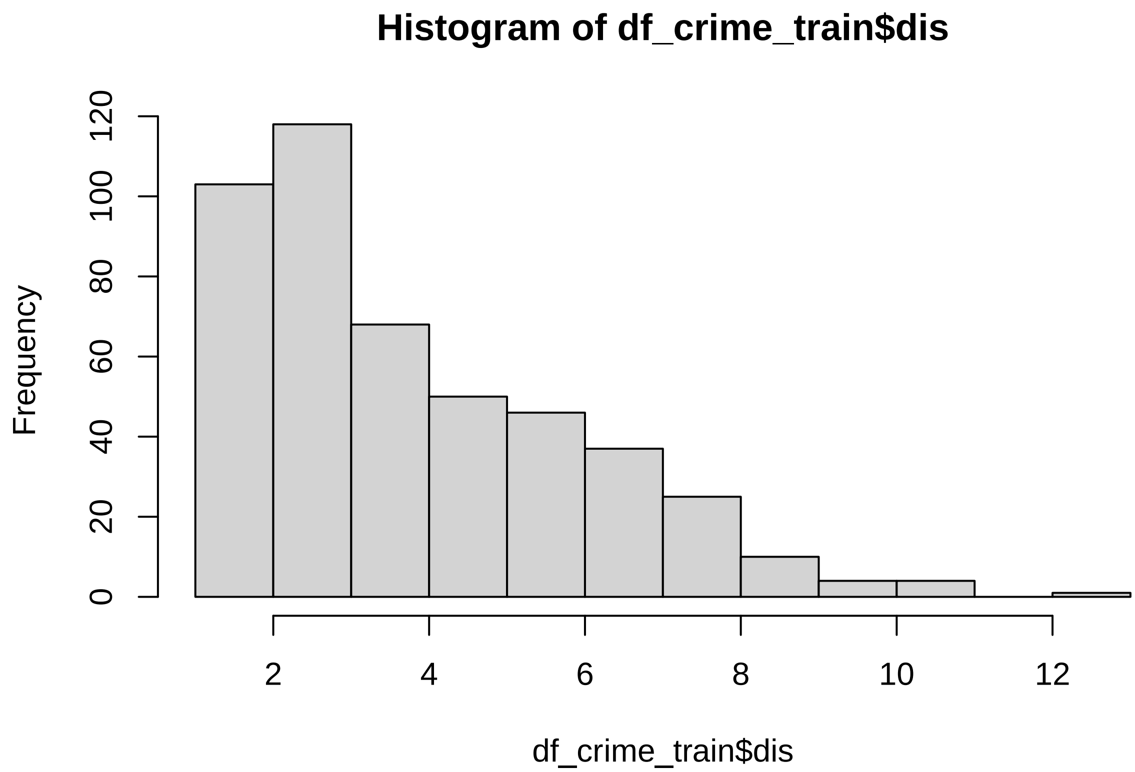



```r
lambda_dis <- bc$x[which.max(bc$y)]

# Apply the Box-Cox transformation
dis_new = (dis_list^lambda_dis-1)/lambda_dis
```

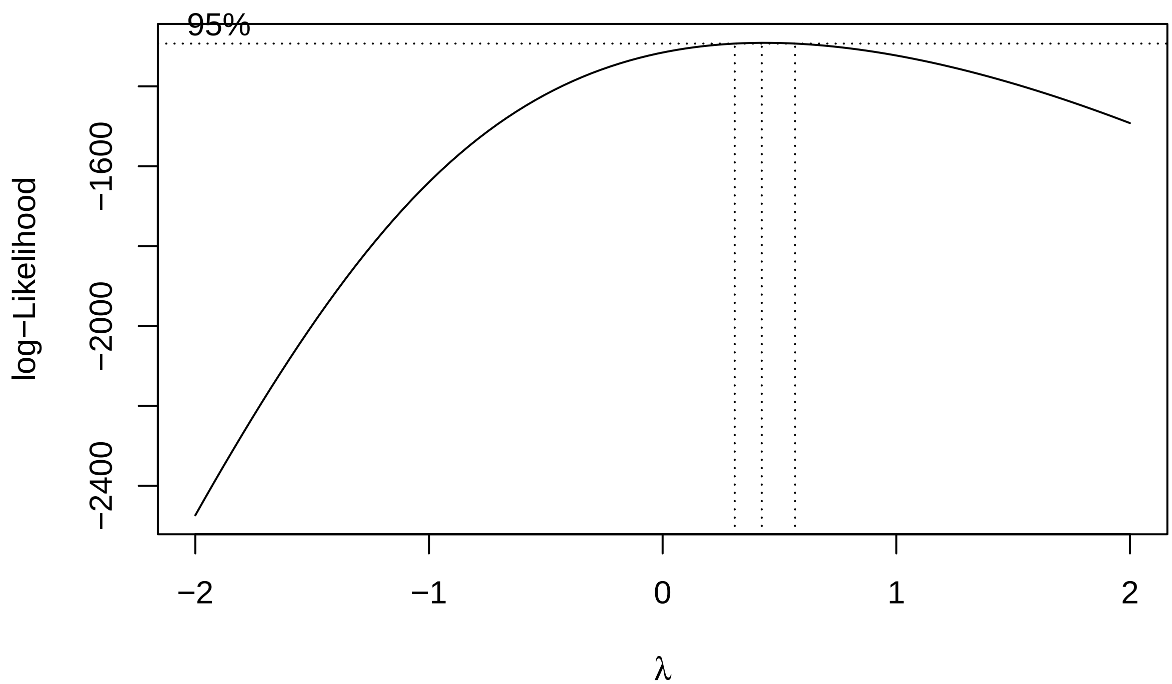
```r
hist(dis_new)
```



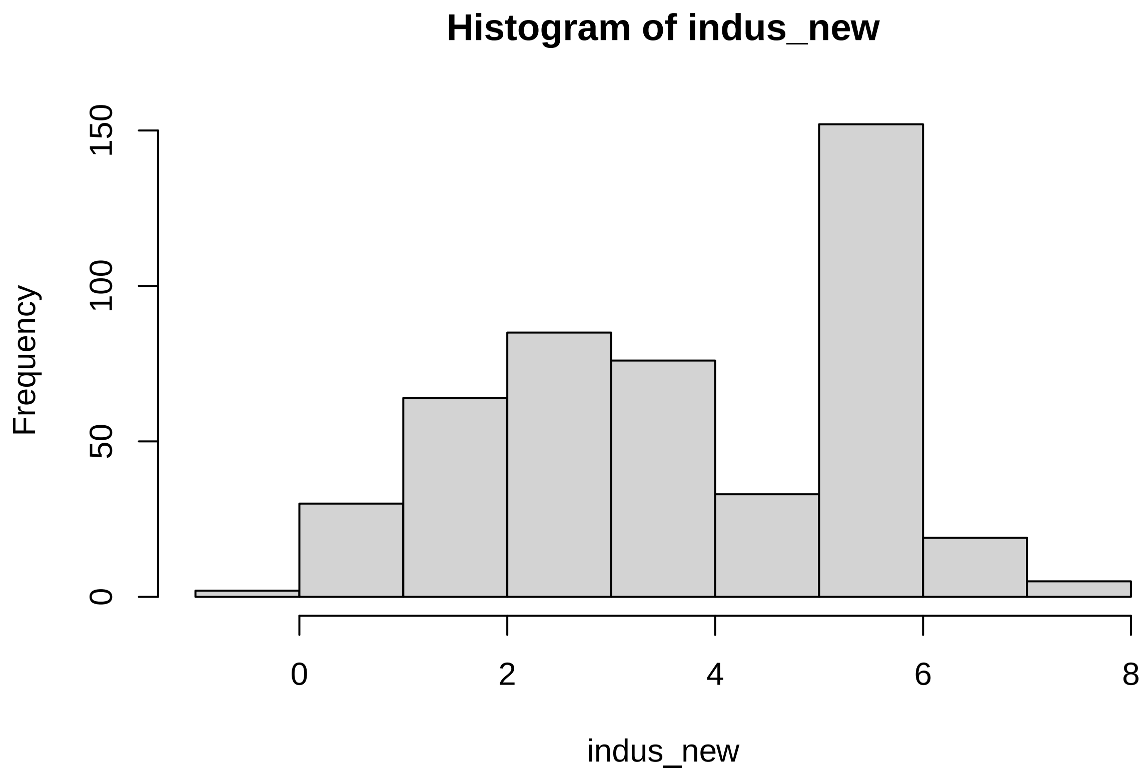

```
hist(df_crime_train$dis)
```



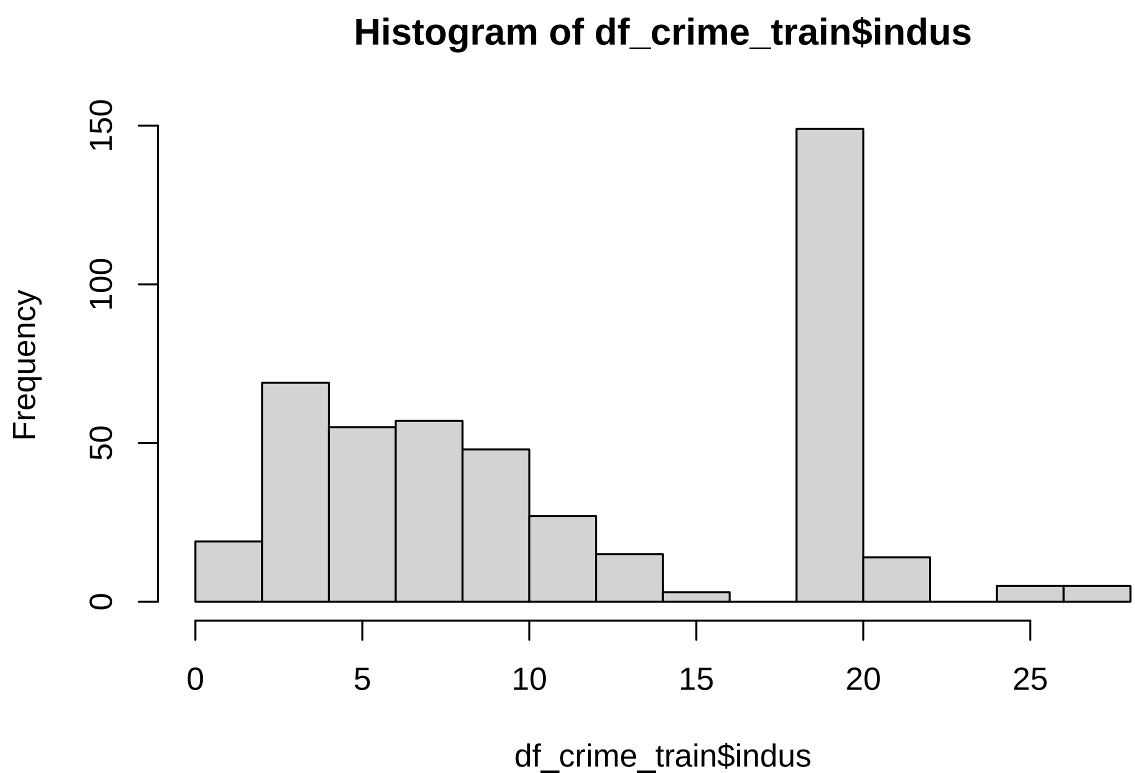

```
# Convert a DataFrame column to a list
indus_list <- as.numeric(as.list(df_crime_train$indus))

#find optimal lambda for Box-Cox transformation
bc <- boxcox(indus_list~ 1, lambda = seq(-2,2,0.1))
```



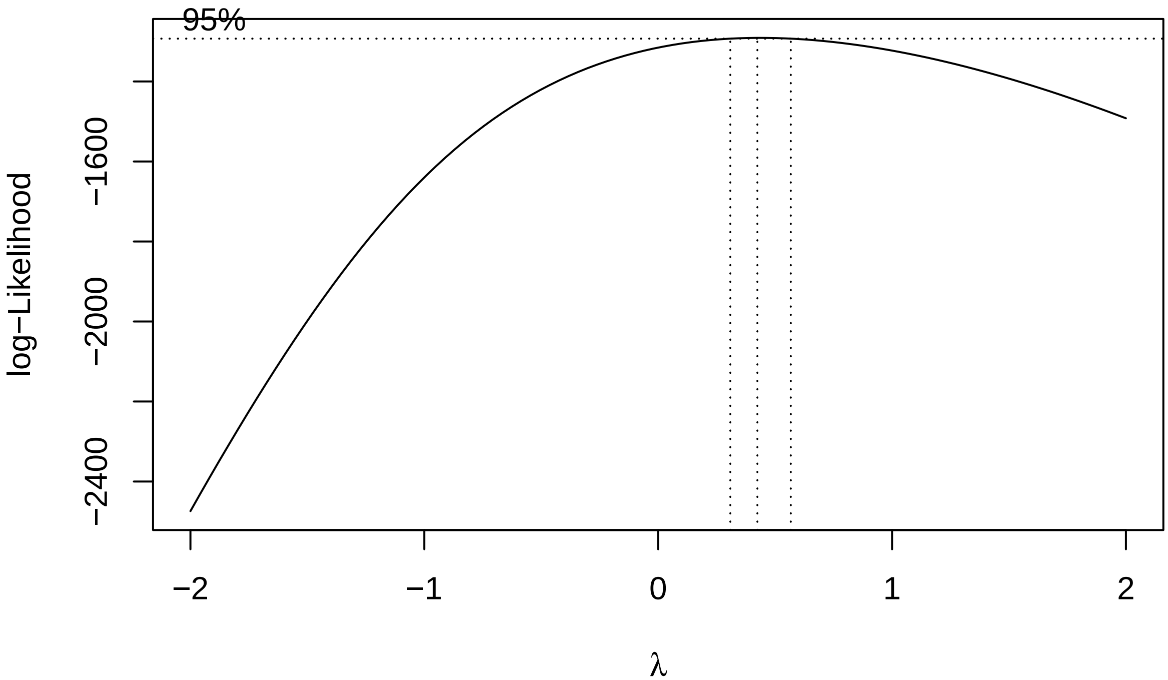

```
lambda_indus <- bc$x[which.max(bc$y)]

# Apply the Box-Cox transformation
indus_new = (indus_list^lambda_indus-1)/lambda_indus


hist(indus_new )
```

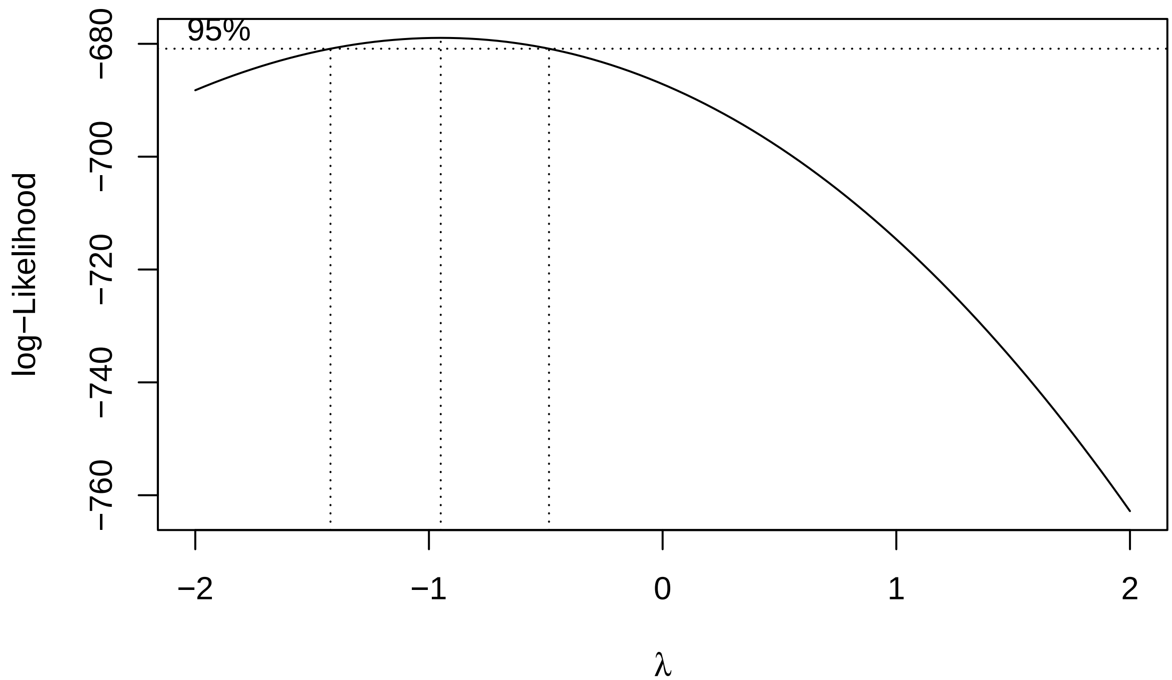


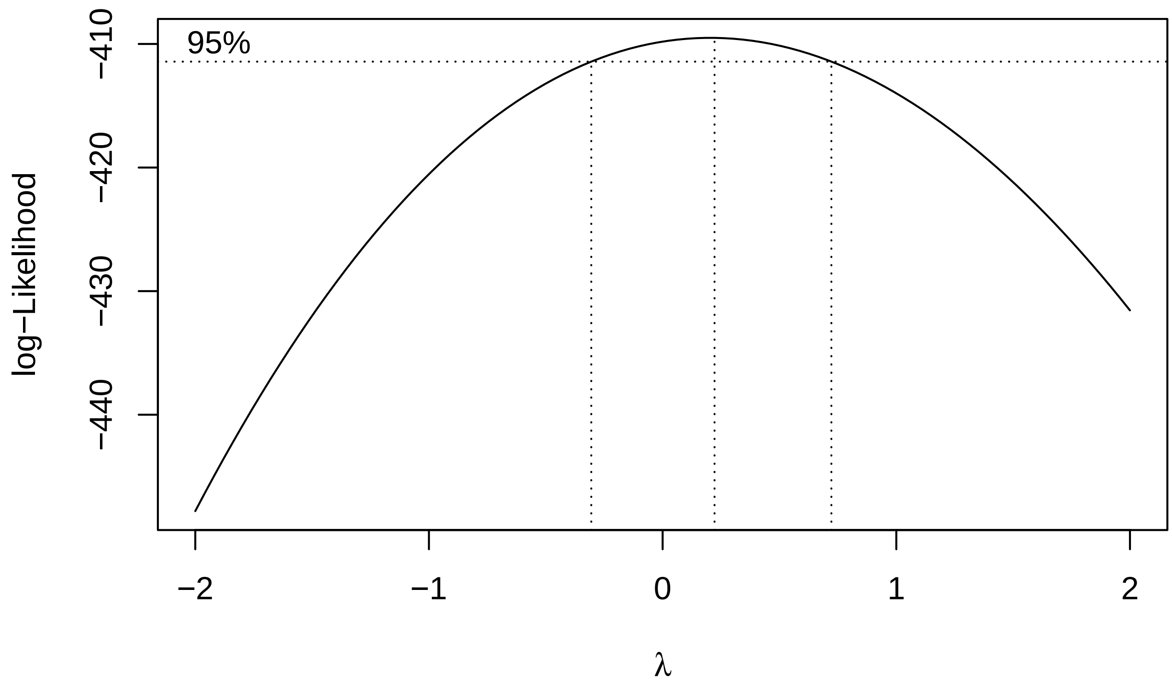

```r
hist(df_crime_train$indus)
```



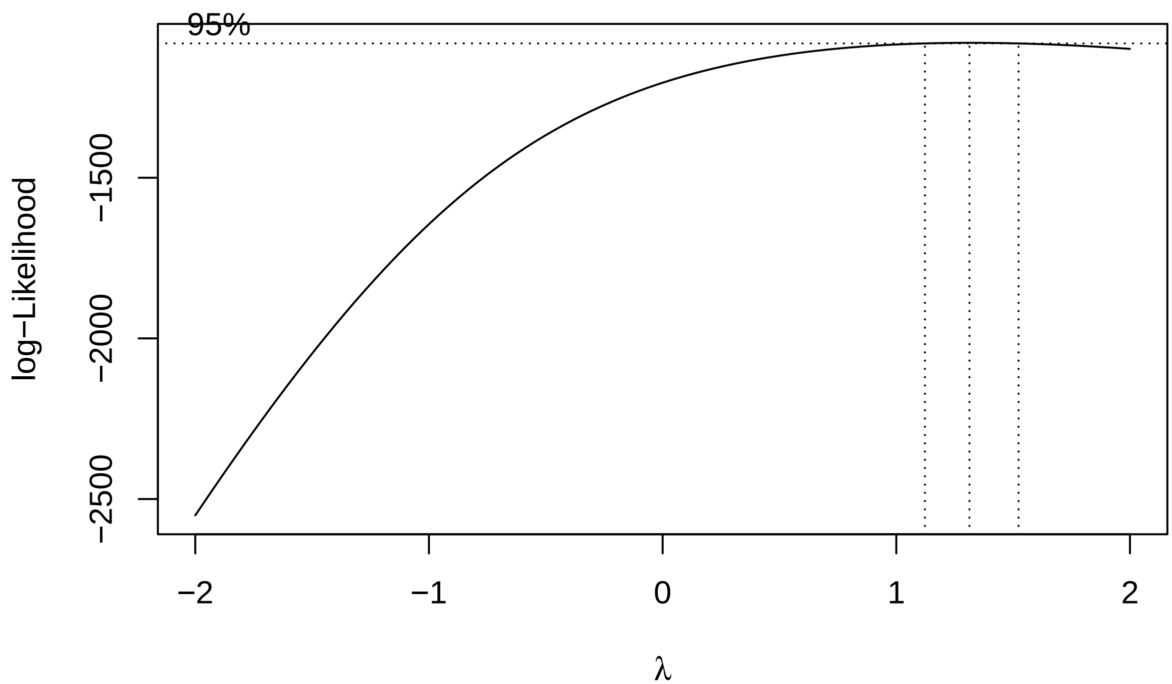

```
# Create an empty list to store the transformed columns
transformed_columns <- list()

# Define the names of columns to exclude from transformation because there variables response must be p
exclude_columns <- c("target", "zn", "chas")

# Iterate through the columns in df_crime_train
for (col_name in names(df_crime_train)) {
  # Convert the column to a list and check if it's numeric and not in the exclude list
  if (is.numeric(df_crime_train[[col_name]]) && !(col_name %in% exclude_columns)) {
    col_list <- as.numeric(as.list(df_crime_train[[col_name]]))

    # Find optimal lambda for Box-Cox transformation
    bc <- boxcox(col_list ~ 1, lambda = seq(-2, 2, 0.1))
    lambda_col <- bc$x[which.max(bc$y)]

    # Apply the Box-Cox transformation
    col_new <- (col_list^lambda_col - 1) / lambda_col


    # Store the transformed column in the list
    transformed_columns[[col_name]] <- col_new
  }
}
```

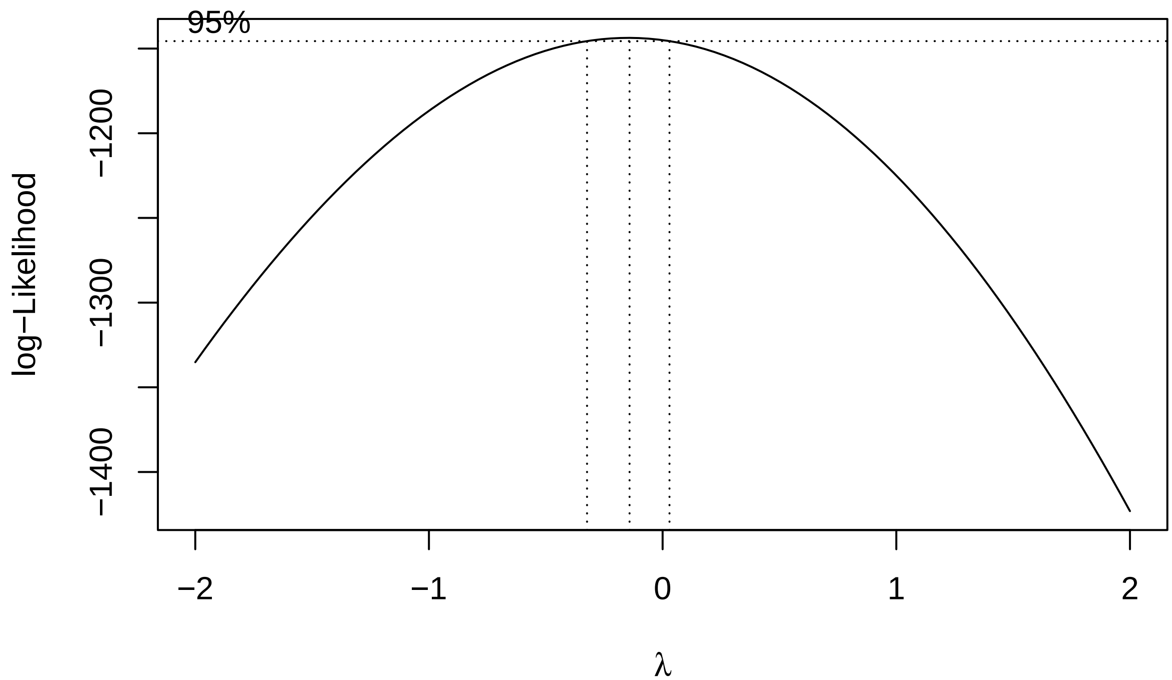

95%
log−Likelihood
−1600
−2000
−2400
−2
−1
0
1
2
λ

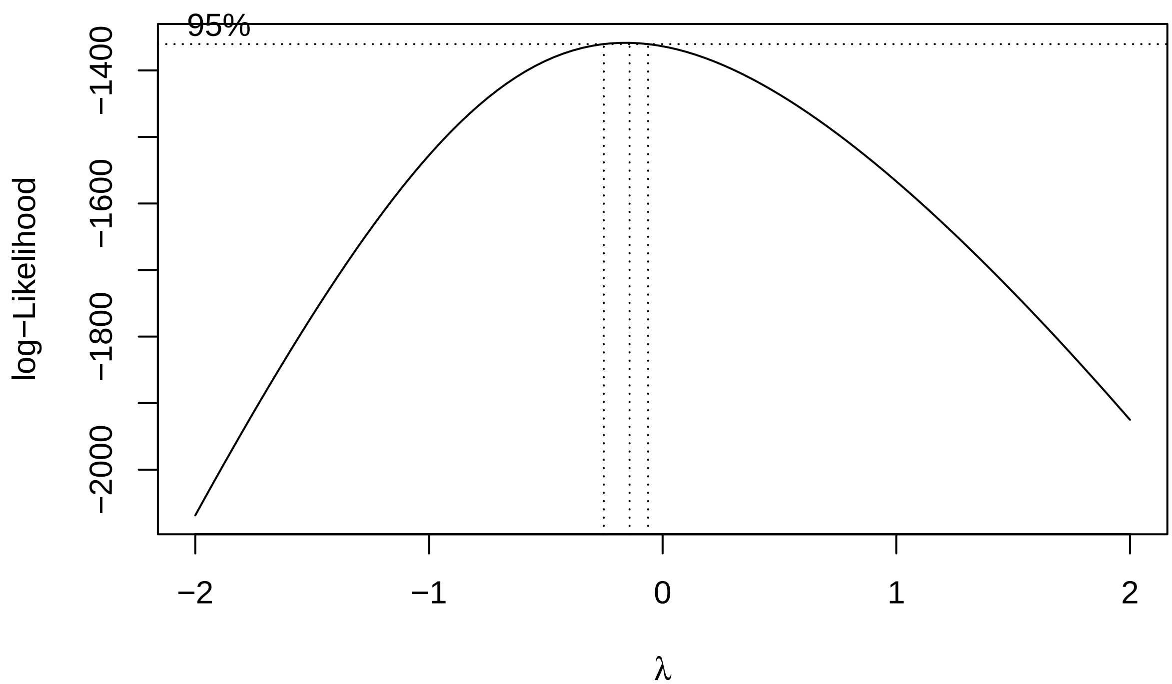

95%
log−Likelihood
−680
−700
−720
−740
−760
−2
−1
0
1
2
λ



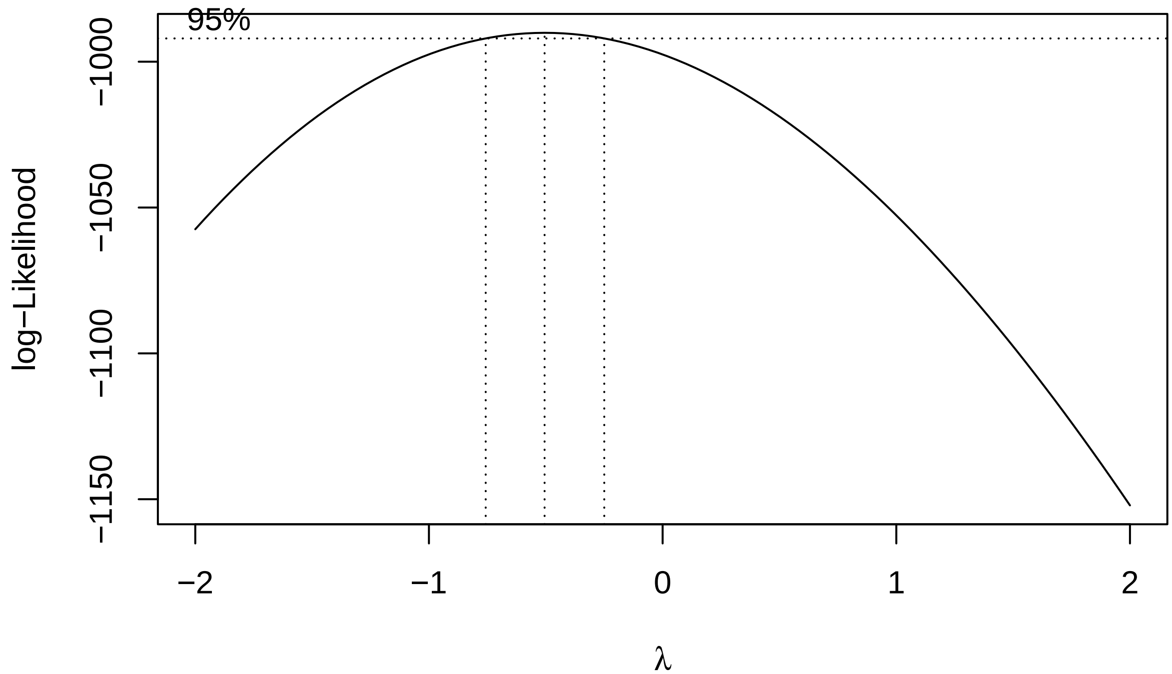
95%
−410
−420
−430
−440
log−Likelihood
−2
−1
0
1
2
λ

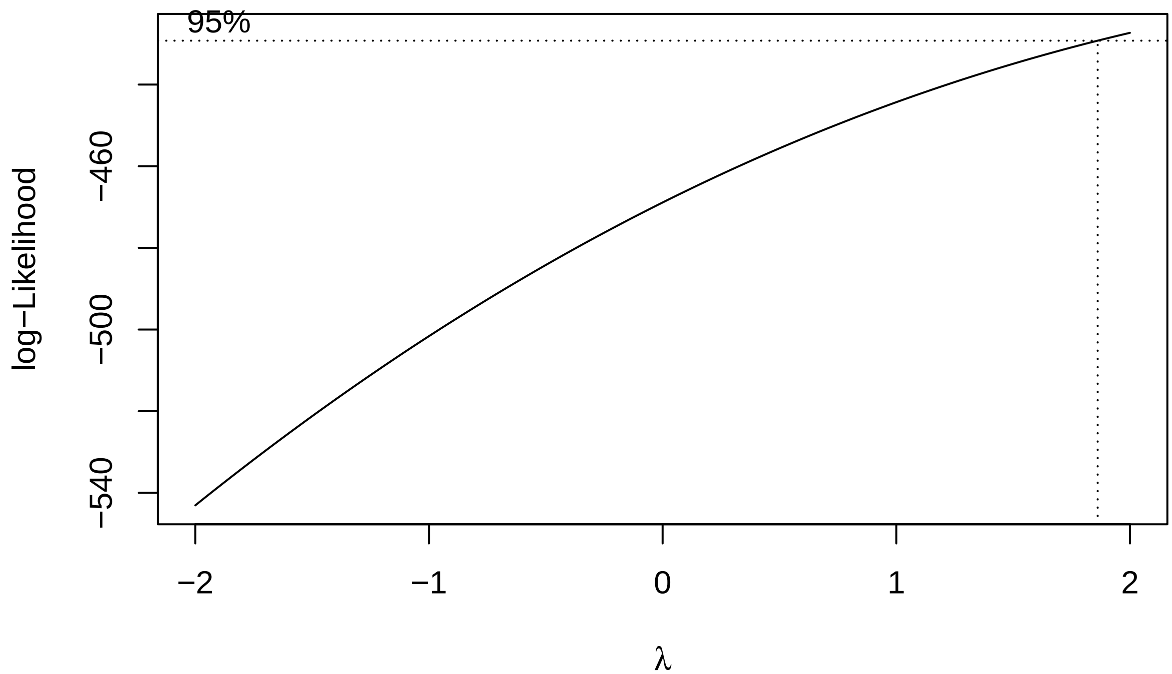

95%
log−Likelihood
−1500
−2000
−2500
−2
−1
0
1
2
λ

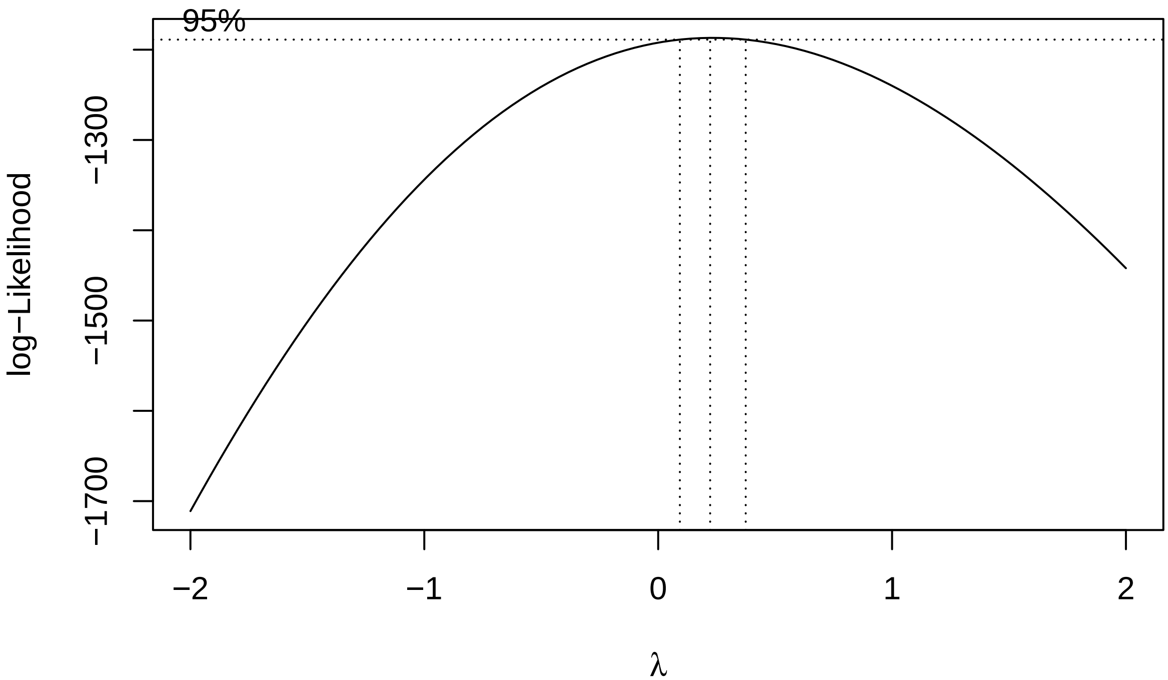

95%
log−Likelihood
−1200
−1300
−1400
−2
−1
0
1
2
λ



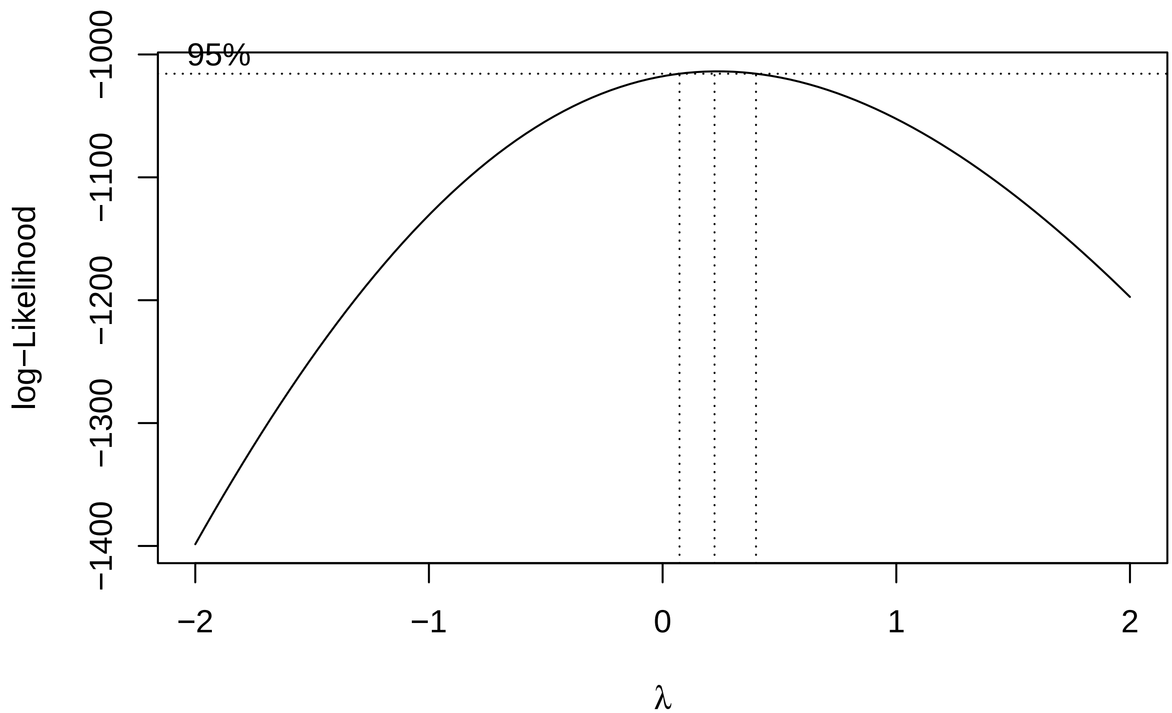
95%
−1400
−1600
−1800
−2000
log−Likelihood
−2
−1
0
1
2
λ


95%
log−Likelihood
−1000
−1050
−1100
−1150
−2
−1
0
1
2
λ


95%
log−Likelihood
−460
−500
−540
−2
−1
0
1
2
λ



95%
log−Likelihood
−1300
−1500
−1700
−2
−1
0
1
2
λ




```r
# Convert the list of transformed columns into a DataFrame
df_transformed <- as.data.frame(transformed_columns)
```

2.10 col_new2

```r
col_new2 <- ifelse(col_list==0, log(col_list), (col_list^lambda_col - 1) / lambda_col)
as.data.frame(col_new2)
```

```
##     col_new2
## 1   6.233995
## 2   3.510880
## 3   3.762396
## 4   4.593104
## 5   5.593022
## 6   4.821577
## 7   1.934861
## 8   2.434451
## 9   4.461941
## 10  4.342567
## 11  4.685243
## 12  4.389146
## 13  4.701653
## 14  5.869446
```

```
## 15  5.053582
## 16  4.988253
## 17  5.193404
## 18  4.515209
## 19  2.701983
## 20  3.151591
## 21  4.236960
## 22  5.010204
## 23  5.300386
## 24  4.618557
## 25  4.541436
## 26  5.017482
## 27  4.157250
## 28  3.243248
## 29  4.523981
## 30  4.461941
## 31  4.643764
## 32  4.236960
## 33  5.922803
## 34  2.921376
## 35  4.314202
## 36  4.928832
## 37  6.233995
## 38  3.665026
## 39  3.967874
## 40  4.470896
## 41  4.187442
## 42  3.576362
## 43  4.497576
## 44  4.074886
## 45  4.147104
## 46  3.901322
## 47  4.370617
## 48  4.236960
## 49  4.217271
## 50  4.351951
## 51  4.443936
## 52  4.147104
## 53  4.147104
## 54  3.890049
## 55  3.855908
## 56  4.256495
## 57  6.006236
## 58  6.036934
## 59  4.407540
## 60  2.520614
## 61  4.601616
## 62  3.844418
## 63  3.287687
## 64  5.220538
## 65  3.956909
## 66  4.295113
## 67  2.434451
## 68  3.912543
```

```
## 69  4.227135
## 70  4.709819
## 71  6.233995
## 72  4.266205
## 73  3.989657
## 74  4.988253
## 75  3.316828
## 76  4.567399
## 77  3.762396
## 78  4.558773
## 79  5.313475
## 80  4.541436
## 81  4.136916
## 82  4.532723
## 83  3.470674
## 84  4.973520
## 85  6.233995
## 86  4.593104
## 87  5.787563
## 88  3.497557
## 89  4.370617
## 90  4.227135
## 91  4.936331
## 92  5.010204
## 93  2.541587
## 94  4.461941
## 95  5.539262
## 96  4.389146
## 97  4.523981
## 98  4.043273
## 99  2.938665
## 100 4.361301
## 101 3.415935
## 102 4.652113
## 103 4.197425
## 104 4.147104
## 105 3.867342
## 106 4.246746
## 107 4.443936
## 108 5.166002
## 109 3.956909
## 110 4.497576
## 111 6.233995
## 112 4.558773
## 113 4.207367
## 114 4.550119
## 115 4.187442
## 116 4.626987
## 117 4.157250
## 118 4.285514
## 119 3.786122
## 120 5.089210
## 121 4.000475
## 122 4.726077
```

```
## 123 5.200212
## 124 4.928832
## 125 6.176205
## 126 4.515209
## 127 3.563414
## 128 4.126686
## 129 4.207367
## 130 4.868070
## 131 3.072162
## 132 3.923711
## 133 4.685243
## 134 4.266205
## 135 6.233995
## 136 4.966123
## 137 3.890049
## 138 4.541436
## 139 3.589238
## 140 4.558773
## 141 4.370617
## 142 4.256495
## 143 6.233995
## 144 4.064394
## 145 4.000475
## 146 5.975221
## 147 3.640037
## 148 5.193404
## 149 2.477998
## 150 4.342567
## 151 3.510880
## 152 4.813748
## 153 4.479820
## 154 2.663051
## 155 4.601616
## 156 5.247409
## 157 2.721176
## 158 5.842404
## 159 4.116413
## 160 6.233995
## 161 3.497557
## 162 5.575213
## 163 5.441123
## 164 4.197425
## 165 4.541436
## 166 4.798022
## 167 4.470896
## 168 2.740191
## 169 3.072162
## 170 4.416688
## 171 5.138325
## 172 3.470674
## 173 4.246746
## 174 4.701653
## 175 5.152198
## 176 3.039558
```

```
## 177 4.434885
## 178 4.285514
## 179 2.796214
## 180 5.131362
## 181 4.988253
## 182 4.461941
## 183 4.550119
## 184 5.441123
## 185 4.813748
## 186 4.898623
## 187 5.933361
## 188 3.844418
## 189 4.434885
## 190 4.593104
## 191 4.610100
## 192 4.668730
## 193 4.717961
## 194 2.777707
## 195 3.750443
## 196 4.677000
## 197 4.479820
## 198 3.402041
## 199 3.967874
## 200 5.508954
## 201 4.575995
## 202 4.032644
## 203 2.796214
## 204 4.032644
## 205 4.701653
## 206 3.652565
## 207 4.488713
## 208 5.428615
## 209 4.304676
## 210 4.398360
## 211 3.470674
## 212 4.425803
## 213 3.989657
## 214 4.136916
## 215 3.039558
## 216 4.323693
## 217 4.167355
## 218 5.906897
## 219 3.912543
## 220 4.443936
## 221 5.478322
## 222 4.197425
## 223 5.267393
## 224 6.233995
## 225 5.186579
## 226 4.285514
## 227 3.738429
## 228 5.809622
## 229 3.652565
## 230 4.207367
```

```
## 231 4.246746
## 232 4.032644
## 233 4.106097
## 234 4.167355
## 235 4.532723
## 236 5.280637
## 237 4.314202
## 238 5.293819
## 239 4.550119
## 240 3.497557
## 241 3.956909
## 242 4.032644
## 243 3.786122
## 244 3.627442
## 245 5.017482
## 246 4.443936
## 247 3.614777
## 248 3.665026
## 249 4.197425
## 250 4.416688
## 251 5.422340
## 252 5.254086
## 253 4.497576
## 254 5.409748
## 255 5.339472
## 256 2.098976
## 257 4.285514
## 258 5.639982
## 259 3.989657
## 260 3.287687
## 261 5.293819
## 262 4.652113
## 263 4.660435
## 264 4.085334
## 265 3.563414
## 266 4.584564
## 267 4.398360
## 268 3.786122
## 269 4.701653
## 270 3.302305
## 271 3.457112
## 272 3.302305
## 273 4.095737
## 274 5.039199
## 275 4.389146
## 276 4.275878
## 277 3.415935
## 278 3.901322
## 279 3.151591
## 280 4.643764
## 281 4.626987
## 282 4.416688
## 283 4.217271
## 284 3.702015
```

```
## 285 4.515209
## 286 3.714216
## 287 4.407540
## 288 4.106097
## 289 3.213117
## 290 3.388062
## 291 4.506408
## 292 4.074886
## 293 4.837165
## 294 4.488713
## 295 4.266205
## 296 4.032644
## 297 4.295113
## 298 6.233995
## 299 5.300386
## 300 4.829382
## 301 5.220538
## 302 4.701653
## 303 4.652113
## 304 4.541436
## 305 4.515209
## 306 5.484475
## 307 4.053856
## 308 4.256495
## 309 4.443936
## 310 4.652113
## 311 4.497576
## 312 3.563414
## 313 4.668730
## 314 4.610100
## 315 4.532723
## 316 4.177419
## 317 4.829382
## 318 5.490614
## 319 4.601616
## 320 2.477998
## 321 4.898623
## 322 5.885554
## 323 4.898623
## 324 4.266205
## 325 4.351951
## 326 4.610100
## 327 5.409748
## 328 5.103333
## 329 3.524127
## 330 4.106097
## 331 3.510880
## 332 3.714216
## 333 3.738429
## 334 4.197425
## 335 2.701983
## 336 4.550119
## 337 4.434885
## 338 4.361301
```

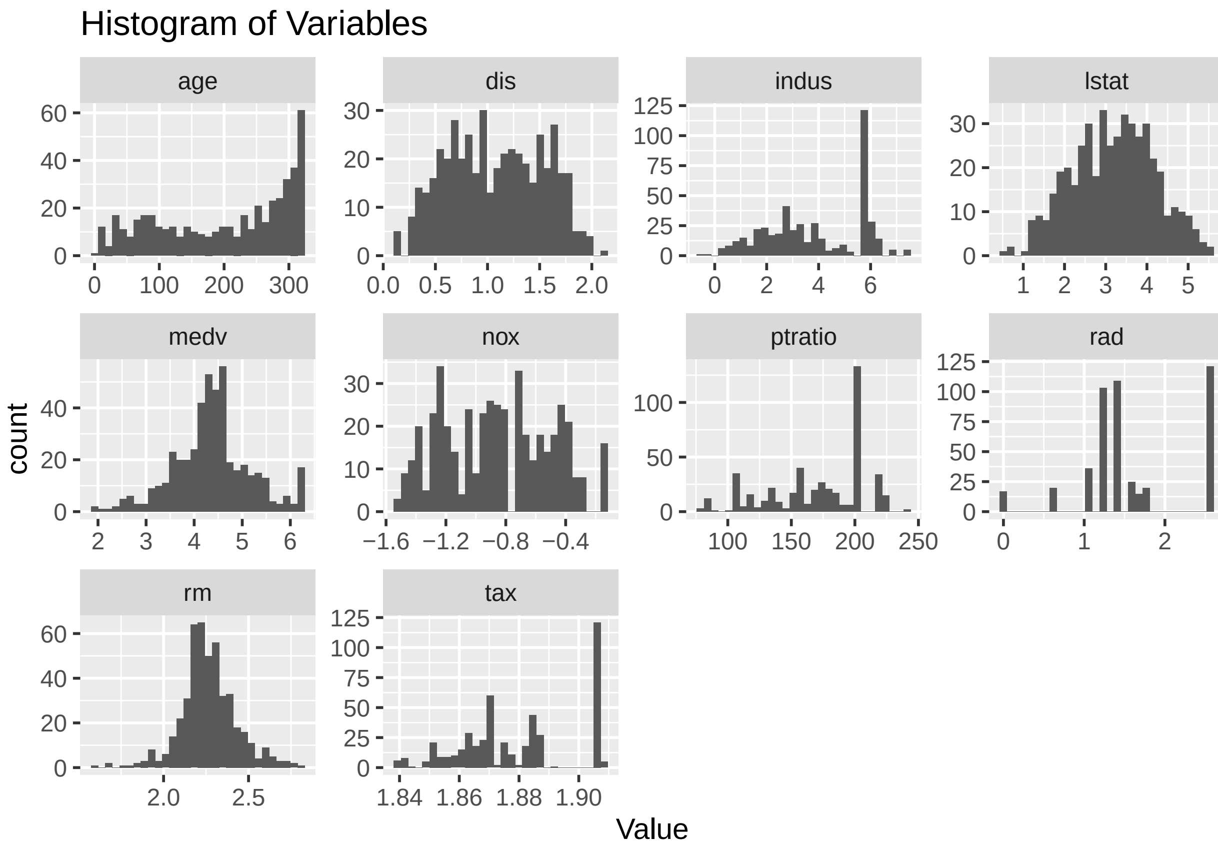
```
## 339 4.333147
## 340 4.601616
## 341 4.558773
## 342 5.703334
## 343 6.233995
## 344 4.541436
## 345 4.701653
## 346 3.576362
## 347 1.934861
## 348 5.490614
## 349 4.304676
## 350 4.323693
## 351 6.161585
## 352 4.523981
## 353 4.304676
## 354 4.187442
## 355 4.187442
## 356 4.951268
## 357 4.187442
## 358 5.089210
## 359 3.563414
## 360 4.314202
## 361 3.640037
## 362 4.541436
## 363 4.295113
## 364 4.443936
## 365 4.333147
## 366 4.488713
## 367 3.923711
## 368 2.477998
## 369 4.177419
## 370 4.295113
## 371 3.510880
## 372 4.567399
## 373 4.333147
## 374 4.011245
## 375 3.359839
## 376 3.702015
## 377 3.726354
## 378 4.685243
## 379 3.088282
## 380 4.443936
## 381 4.943810
## 382 6.072357
## 383 4.256495
## 384 3.415935
## 385 3.039558
## 386 4.506408
## 387 4.246746
## 388 3.537298
## 389 4.000475
## 390 4.416688
## 391 4.635389
## 392 5.075015
```

```
## 393 3.602043
## 394 4.197425
## 395 6.233995
## 396 5.089210
## 397 4.416688
## 398 3.331257
## 399 4.314202
## 400 3.786122
## 401 4.167355
## 402 5.117383
## 403 4.314202
## 404 4.626987
## 405 3.602043
## 406 4.584564
## 407 5.502854
## 408 4.677000
## 409 3.272973
## 410 4.416688
## 411 3.855908
## 412 4.256495
## 413 4.660435
## 414 4.314202
## 415 4.207367
## 416 4.685243
## 417 4.523981
## 418 4.126686
## 419 4.643764
## 420 5.551296
## 421 4.116413
## 422 4.860377
## 423 6.233995
## 424 3.537298
## 425 3.470674
## 426 4.217271
## 427 4.217271
## 428 3.945894
## 429 3.714216
## 430 3.738429
## 431 5.287236
## 432 5.306939
## 433 4.370617
## 434 4.677000
## 435 5.409748
## 436 3.689751
## 437 5.067889
## 438 2.273977
## 439 6.233995
## 440 4.275878
## 441 4.980897
## 442 3.878723
## 443 4.425803
## 444 5.390755
## 445 4.497576
## 446 5.557294
```

```
## 447 3.967874
## 448 5.159109
## 449 3.702015
## 450 5.067889
## 451 3.563414
## 452 4.217271
## 453 5.313475
## 454 2.740191
## 455 3.786122
## 456 4.610100
## 457 4.593104
## 458 4.868070
## 459 6.233995
## 460 4.618557
## 461 3.524127
## 462 3.272973
## 463 3.167135
## 464 3.844418
## 465 4.370617
## 466 2.955815
```

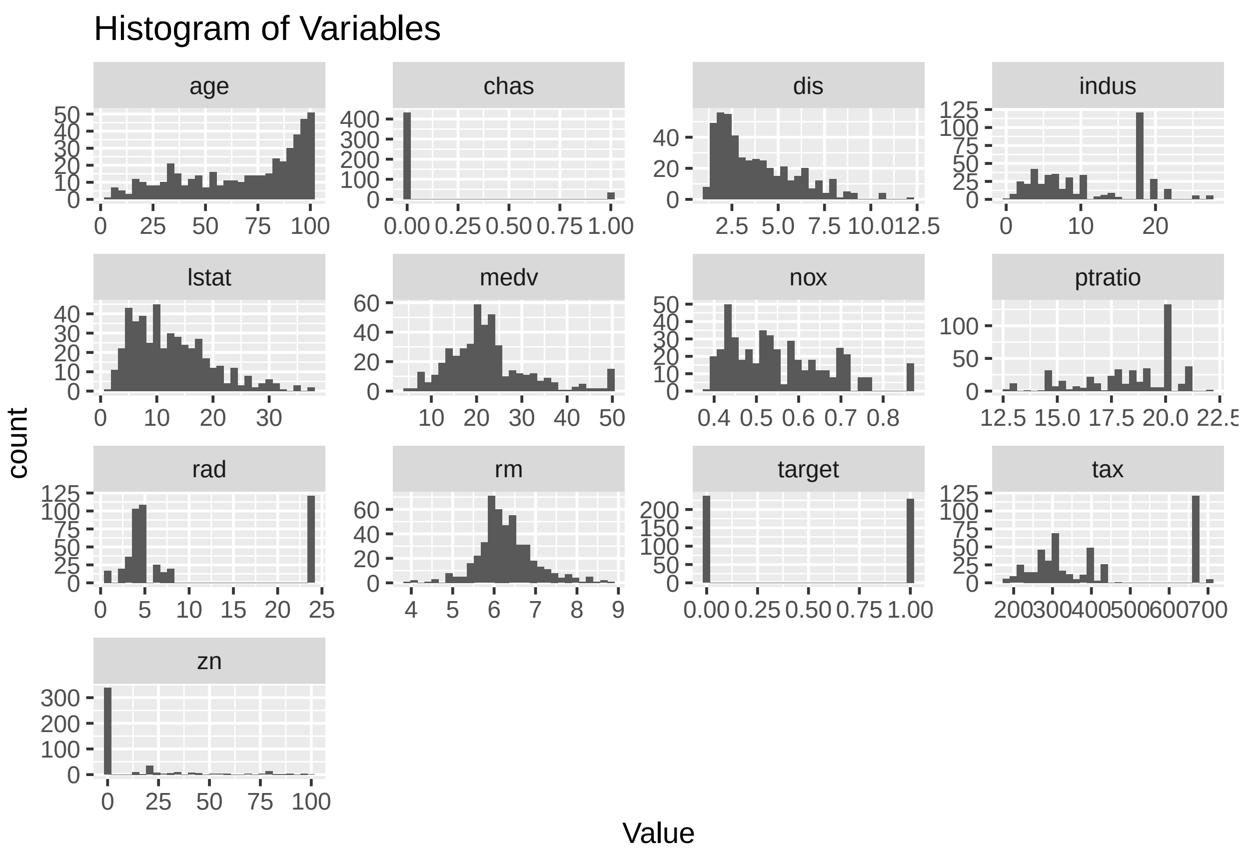
```r
# Gather the data into a long format
data_transformed_long <- gather(df_transformed, key = "Variable", value = "Value")

ggplot(data_transformed_long, aes(x = Value)) +
  geom_histogram() +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Histogram of Variables")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# Gather the data into a long format
data_long <- gather(df_crime_train, key = "Variable", value = "Value")

ggplot(data_long, aes(x = Value)) +
  geom_histogram() +
  facet_wrap(~Variable, scales = "free") +
  labs(title = "Histogram of Variables")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Histogram of Variables
age
chas
dis
indus
lstat
medv
nox
ptratio
rad
rm
target
tax
zn
count
Value