Problem Set 2 – Linear Regression and Extensions

2016-03-06

Loading and Exploring the Data Set

For this problem set we will analyze the data set *Boston* which is contained in the library *MASS*. This data set records the median house value (*medv*) for 506 neighbourhoods around Boston. The goal is to predict the variable *medv* using 13 predictors.

- Load the data set.
- Make yourself familiar with the data. Hint: *str()*, *names()*, *help()*
- Generate Descriptive statistics. Hint: *summary*, *mean*, *sd*, *var*, *min*, *max*, *median*, *range*, *quantile*, *fivenum*
- Plot the data, especially the outcome variable *medv* and the variable *lstat*. Hint: *plot*, *hist*, *boxplot*

```
options(warn=1)
library(MASS)
data(Boston)
?Boston
```

```
## starting httpd help server ...
```

```
##  done
```

```
attach(Boston)
summary(medv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   17.02   21.20   22.53   25.00   50.00
```

```
summary(lstat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.73    6.95   11.36   12.65   16.96   37.97
```

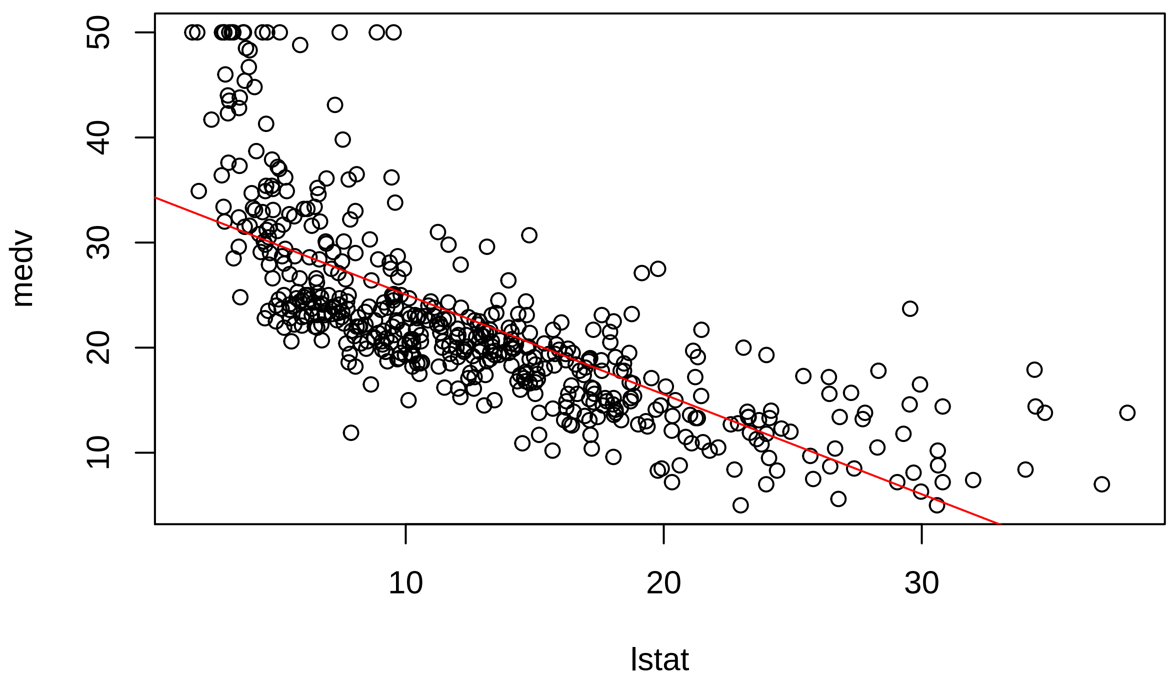
Univariate Linear Regression

- Analyse the relation between *medv* and *lstat* with a linear regression. Hint: *lm()*
- Interpret the results. Hint: *summary*
- Plot the regression line in a graph with the original data points.
- What is the predicted value of *medv* for a region with a *lstat* of 32?

```
reg1 = lm(medv ~ lstat, data=Boston)
summary(reg1)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
#plot(reg1)
plot(medv ~ lstat, data=Boston)
abline(reg1, col="red")
```



```
predict(reg1, newdata = list(lstat=32), interval="confidence")
```

```
##        fit       lwr       upr
## 1 4.152262 2.583078 5.721445
```

Multivariate Linear Regression

- Fit now a multivariate regression.
- Interpret the results, in particular with a focus on the variable *lstat*.
- Fit a more complex model, e.g. considering interaction effects and higher order polyomials.

```
reg2 = lm(medv ~ ., data=Boston)
summary(reg2)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
#plot(reg2)
reg3 = lm(medv ~ poly(lstat,3)+ crim + zn + crim:zn+ (chas + nox + rm)^2, data=Boston)
coef(reg3)
```

```
##     (Intercept) poly(lstat, 3)1 poly(lstat, 3)2 poly(lstat, 3)3
##    -62.13076794   -103.46067332     46.24380810     -8.39074055
##            crim              zn            chas             nox
##     -0.11982070     -0.05169358     15.85977935    103.99734126
##              rm         crim:zn        chas:nox         chas:rm
##     13.61213583      0.56104725    -11.78469462     -0.75608850
##          nox:rm
##    -16.85458207
```

Regression Splines

Now we consider again the relation between *lstat* and *medv*.

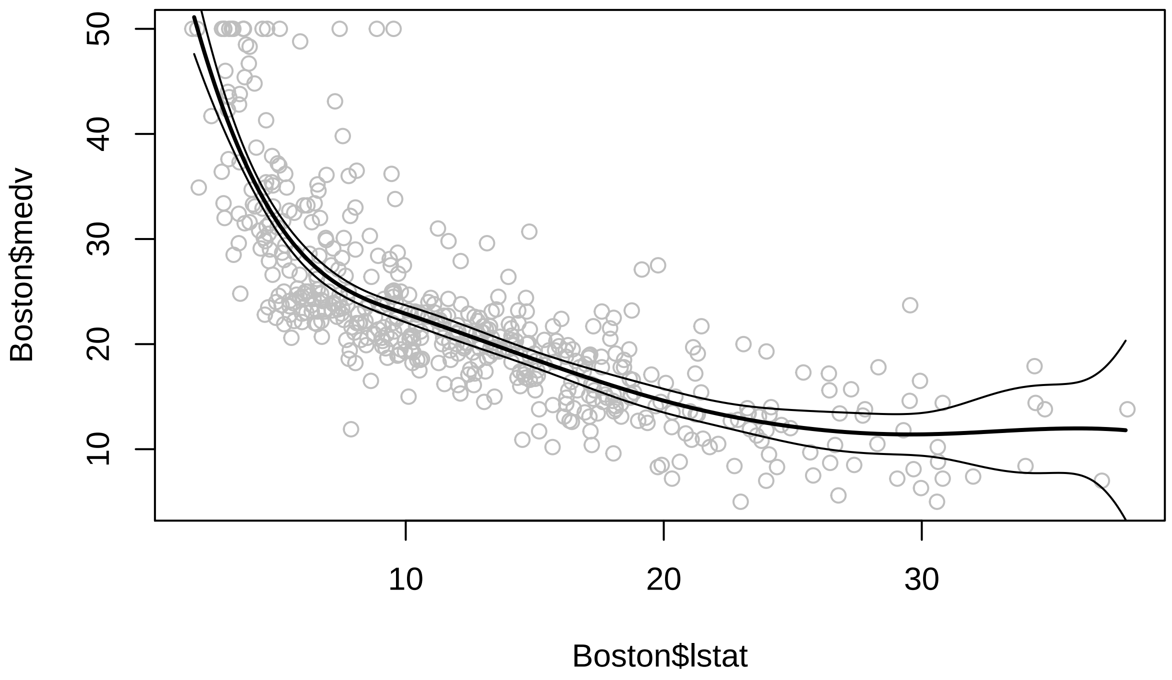
- Fit a cubic regression spline to the data! Hint: library *splines* and function *bs()*
- Plot the fitted line!

```
library(splines)
par(mfrow=c(1,1))
fit = lm(medv ~ bs(lstat, knots=c(10,20,30)), data=Boston)
lstat.grid <- seq(from=1.8, to=37.9, by=0.1)
pred = predict(fit, newdata=list(lstat=lstat.grid), se=TRUE)
plot(Boston$lstat, Boston$medv, col="gray")
lines(lstat.grid, pred$fit, lwd=2)
lines(lstat.grid, pred$fit + 2*pred$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```

```
lines(lstat.grid, pred$fit - 2*pred$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```



* Experiment with different spline specifications! Hint: options *knots* and *df*

```
fit2 = lm(medv ~ bs(lstat, df=20), data=Boston)
attr(bs(lstat, df=20), "knots")
```

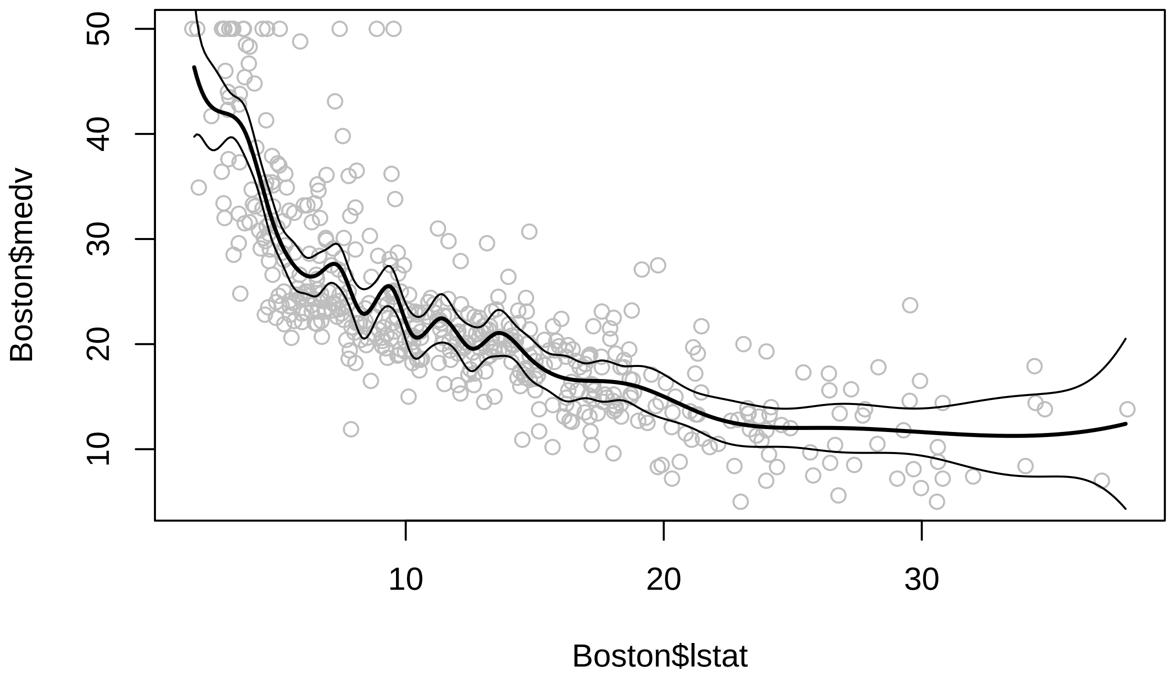
```
## 5.555556% 11.11111% 16.66667% 22.22222% 27.77778% 33.33333% 38.88889%
##  3.762778  4.822222  5.683333  6.582222  7.440000  8.316667  9.457778
## 44.44444%       50% 55.55556% 61.11111% 66.66667% 72.22222% 77.77778%
## 10.198889 11.360000 12.555556 13.486667 14.696667 16.217222 17.548889
## 83.33333% 88.88889% 94.44444%
## 18.841667 21.751111 26.448333
```

```
pred2 = predict(fit2, newdata=list(lstat=lstat.grid), se=TRUE)
plot(Boston$lstat, Boston$medv, col="gray")
lines(lstat.grid, pred2$fit, lwd=2)
lines(lstat.grid, pred2$fit + 2*pred2$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```

```
lines(lstat.grid, pred2$fit - 2*pred2$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```



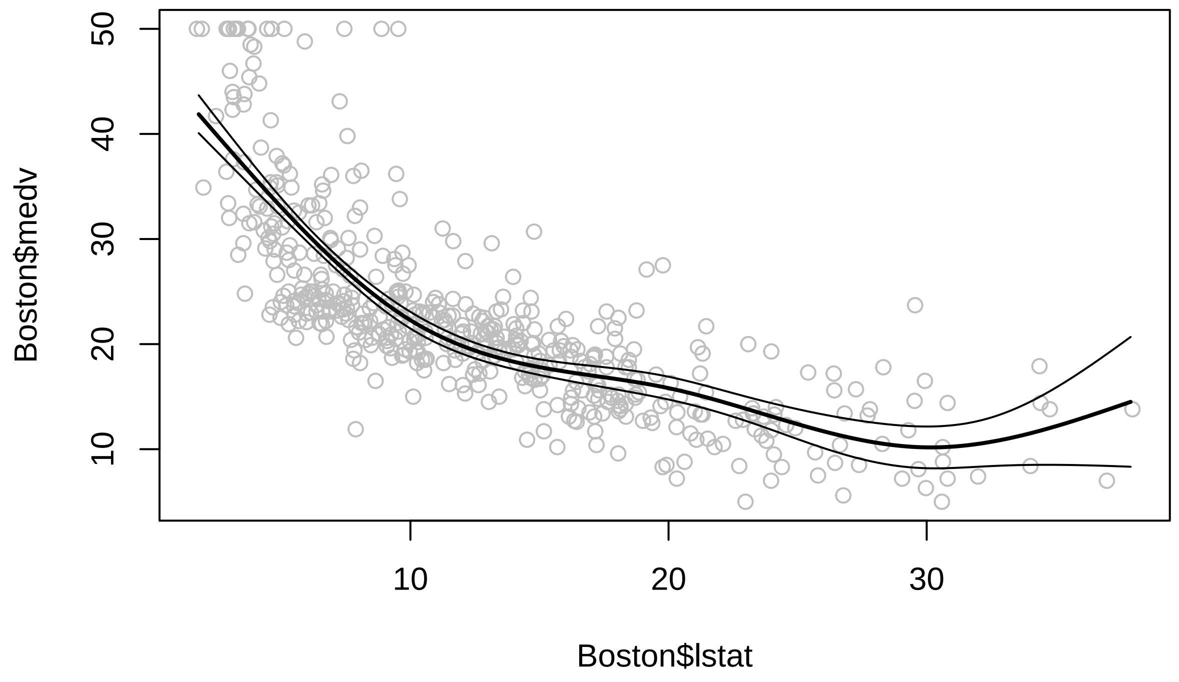
- Fit a natural spline. Hint: *ns()*

```
fit = lm(medv ~ ns(lstat, knots=c(10,20,30)), data=Boston)
lstat.grid <- seq(from=1.8, to=37.9, by=0.1)
pred = predict(fit, newdata=list(lstat=lstat.grid), se=TRUE)
plot(Boston$lstat, Boston$medv, col="gray")
lines(lstat.grid, pred$fit, lwd=2)
lines(lstat.grid, pred$fit + 2*pred$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```

```
lines(lstat.grid, pred$fit - 2*pred$se, lwd="dashed")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): NAs durch Umwandlung
## erzeugt
```



- Compare the different specifications!

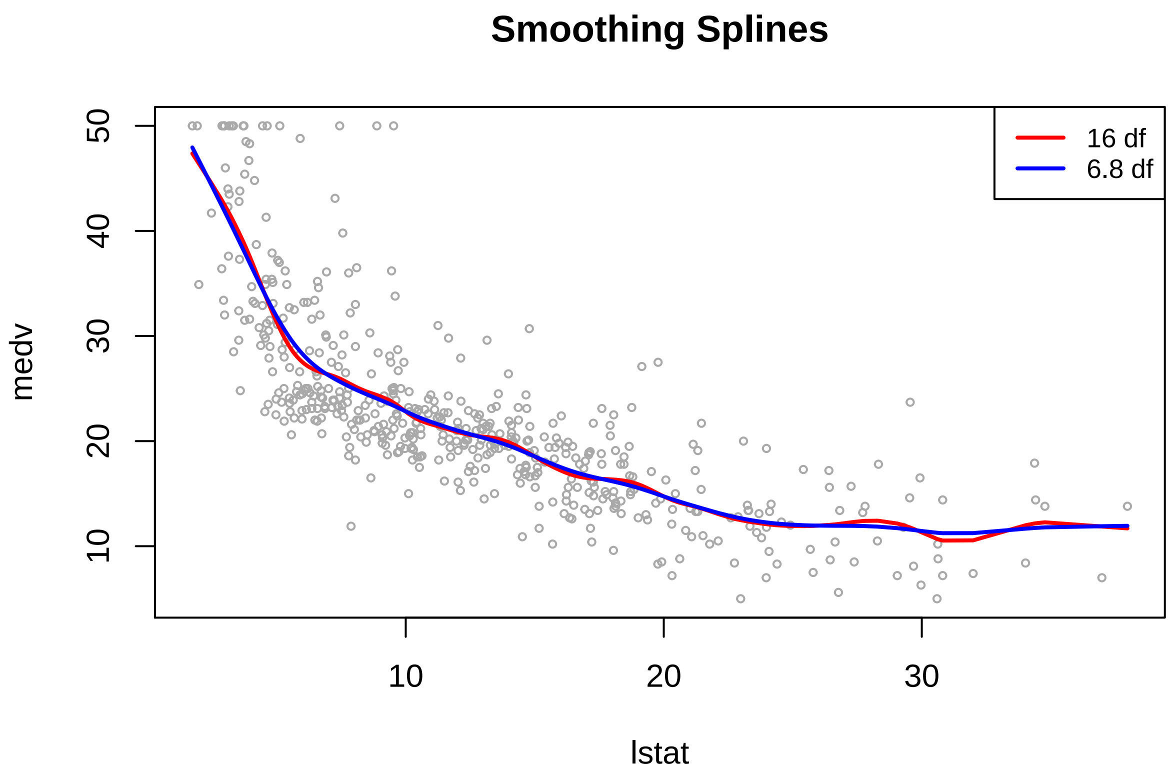
Smoothing Splines

- Fit and plot a smoothing spline to the data! Hint: *smooth.spline()*

```
plot(lstat, medv, cex=.5, col="darkgrey")
title("Smoothing Splines")
fit = smooth.spline(lstat, medv, df=16)
fit2 = smooth.spline(lstat, medv, cv=FALSE) # generalized CV
fit2$df
```

```
## [1] 10.5588
```

```
lines(fit, col="red", lwd=2)
lines(fit2, col="blue", lwd=2)
legend("topright", legend=c("16 df", "6.8 df"), col=c("red", "blue"), lty=1, lwd=2, cex=.8)
```


Smoothing Splines
medv
lstat
16 df
6.8 df
10
20
30
40
50
10
20
30