

# MSG500/MVE190

## Linear Models - Lecture 7

Rebecka Jörnsten  
Mathematical Statistics  
University of Gothenburg/Chalmers University of Technology

November 17, 2015

### 1 RECAP

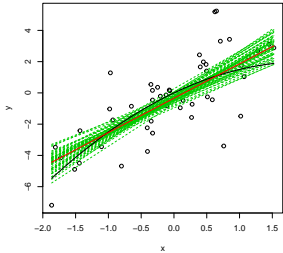
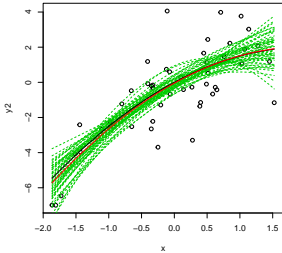
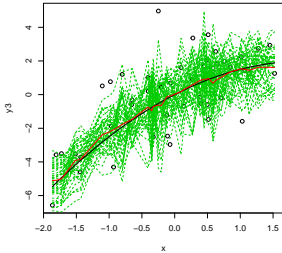
- Multivariate regression - more convenient to work with this in a matrix form
- $y = X\beta + \epsilon$  where  $X$  is a  $n \times p$  matrix where each columns are the values for an  $x$ -variable, column of 1's in the first column for the intercept.
- Normal equations  $X'X\beta = X'y$ , solve for  $\beta$  if you can  $\hat{\beta} = (X'X)^{-1}X'y$
- If  $x$ 's are correlated, then  $\hat{\beta}_j$  depend not only the correlation between variable  $x_j$  and  $y$  but also all the other correlations  $\text{corr}(x_k, y)$  - difficult to interpret multivariate regression parameters
- Correlated  $x$ 's also lead to numerical instabilities in the estimation due to the fact that the inverse of  $X'X$  is unstable
- $E[\hat{\beta}] = \beta$  and  $V(\hat{\beta}) = \sigma^2(X'X)^{-1}$ .
- Three sources of estimation variance: noise level  $\sigma^2$ , sample size and structure in  $X$  (variance of each  $x'$  and correlation between  $x$ 's).
- You can test the significance of  $\hat{\beta}_j$  using a t-test:  $t_j = \frac{(\hat{\beta})_j - 0}{\sqrt{\hat{\sigma}^2(X'X)^{-1}_{jj}}}$  compare with  $t_{n-p}$  distribution
- Caution: if  $x$ 's are correlated the t-test can give misleading results. Also, look out for multiple testing issues!
- The goodness-of-fit F-test:  $F_{obs} = \frac{(SS_T - RSS)/(p-1)}{RSS/(n-p)}$  compare with  $F_{p-1, n-p}$ . Use to test that all  $\beta_j, j = 1, \dots, p-1$  is zero
- You can also use an F-test to compare any pair of nested models.  $F_{obs} = \frac{(RSS_{simple} - RSS_{complex})/(\Delta p)}{RSS_{complex}/(n - p_{complex})}$ , where  $\Delta p$  is the difference in number of parameters between the simple and complex models. Compare  $F_{obs}$  to  $F_{\Delta p, n - p_{complex}}$ .

### 2 Model Selection

The tools we have discussed so far are t-tests on individual slope estimates and F-test to compare two nested models. Last lecture we also looked at using the F-test to select between models in a backward search - starting with a model including all variables and dropping them one by one until the drop leads to a rejection in the F-test. Now we will look into other criteria that can be used to select between models.

The ultimate validation of a model is if it can be used to *predict* future observations, i.e. does the model generalize to the population under consideration? If you look back to lecture 1, I talked about how the safe option of picking as complex a model as possible was not so safe statistically. That was because while a complex model may have little or no bias it suffers from high estimation variance. This is what will hurt us when we do prediction. We don't know how much 'off' the true model our estimate

is, but with a high estimation variance this deviation can be quite substantial. This will result in poor prediction performance for the model. On the other hand, a simple model may make consistent errors in the prediction, but these predictions will not be too sensitive to individual data sets used for estimation of model parameters. This may in fact work better for prediction. Let's recap some things we talked about in lecture 1:

Model	Simple	added complexity	Flexible
	few parameters	...	many parameters
	rigid	...	adapts to data
	Linear	polynomial/nonlinear	Local average/smoother
			
Properties	Large bias, low variance	...	Small bias, high variance

Let us summarize this bias-variance tradeoff:

$$E[\hat{y}] - E[y], \text{ average fitted value across many data sets - the true model value} = \text{Bias}$$

$$V[\hat{y}] = \text{the estimation variance}$$

With more complex models we can adapt to the data and reduce the bias, but more complex models results in increased estimation variance. In terms of prediction, both bias and variance work against us.

Bias means we are mismatching our model to the true  $y - X$  relationship, so even if we were able to repeat estimation of 1000s of data set so that randomness of estimation due to noise cancels out, our predictions would still be off.

Estimation variance has to do what happens with a particular sample, i.e. how much model fits can vary from sample to sample just due to chance. The estimated model can be quite far from its ideal (what you would get if you could average your estimated models over many finite data sets drawn from the same true model) so even if the model has no bias, due to estimation variance our predictions based on an estimated model can be off.

We want to control both bias and variance. We combine the two in the Mean Squared Error (MSE):

$$MSE = Bias^2 + Variance.$$

A good prediction model can thus be identified as one that minimizes MSE. To avoid confusion with the MSE defined as  $RSS/(n - p)$  I will refer to the  $MSE = Bias^2 + Variance$  as the *prediction MSE*.

How can we select models in practise? We can't actually compute the prediction MSE since we need to know the true model to compute the bias! We need to find a substitute for the prediction MSE that we can compute from observed data. Can we use the residual sum of squares (RSS) as a substitute? No! The  $RSS = \sum_i (y_i - \hat{y}_i)^2$  always decreasing the more complex the model is (the more parameters to allow the model to be matched to the data). In fact, if we use  $n$  parameters we can get a perfect fit to the data,  $RSS = 0$ . That would make no sense for prediction though since we are then building a model to match the random error  $\epsilon_i$ .

Let's revisit the South-African heart disease data. There are 11 explanatory variables and the outcome is the cholesterol level (ldl). Let's fit all possible subset models to this data. There are 11 models

of size 1 (size = number of variables),  $(11 \cdot 10 / 2)$  models of size 2, etc. In Figure 1 I depict the RSS as a function of model size for all subset models.

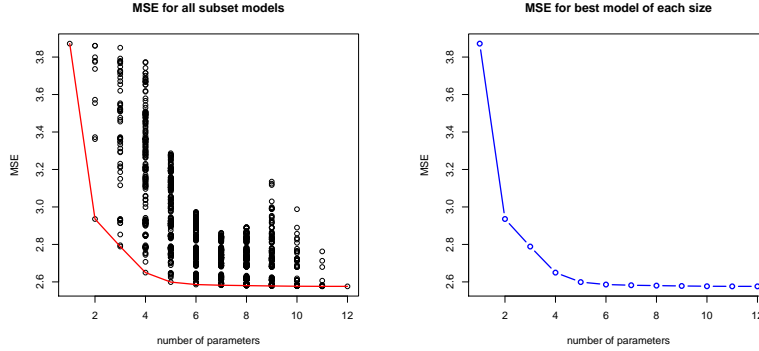


Figure 1: RSS as a function of model size for all subset models for the SA heart disease data. The blue line connects the best model of each size.

It is more common to only look at the winning models of each size. In the right panel of Figure 1 I show you the so-called *RSS curve*. Note that this is decreasing with model size. The RSS (or MSE) is minimized for the largest model, in this case the one including all the variables. Thus, RSS is not a good substitute for prediction MSE.

As a side-note, we revisit the F-test. The backward F-test works along the RSS-curve in the right panel of Figure 1 as follows: You compare the winning model of each size in a sequential fashion. You start by testing the best size 12 model against the best size 11 model. If you fail to reject you go on, comparing the best size 11 model to the best size 10 model. The first rejection means you stop dropping variables. A forward search works similarly. You compare the best size 0 model to the best size 1 model. If you reject the null you go on and compare the best size 1 model to the best size 2 model. The first time you fail to reject, you stop adding variables. What this means is that the forward/backward test stops where the RSS curves levels off. Look for an "elbow" in the RSS curve.

## 2.1 Using training and test data

Let us assume we had access to another data set. Our *training* data is used to estimate model parameters. Our *test* data is used to check which models work well for prediction.

Training data:  $(X_i, y_i)_{i=1}^n, X_i = (x_{i1}, x_{i2}, \dots, x_{i,p-1})$

Test data:  $(X_i, y_i^{new})_{i=1}^n, X_i = (x_{i1}, x_{i2}, \dots, x_{i,p-1})$

The test data consist of new outcome data drawn from the same true model and at the same  $x$ -locations as the training data.

The true model  $\beta^* = (\beta_0^*, \beta_1^*, \dots, \beta_{p-1}^*)$  is unknown to us. If the outcome is not related to some of the  $x$ -variables some of the  $\beta_j^* = 0$ . We can thus write

Training data:  $y_i = X_i \beta^* + \epsilon_i, \epsilon_i \sim N(0, \sigma^2)$

Test data:  $y_i^{new} = X_i \beta^* + \epsilon_i^{new}, \epsilon_i^{new} \sim N(0, \sigma^2)$

1. We enumerate all models  $m = 1, \dots, M$ . If we have  $p - 1$  variables there are  $M = 2^{p-1}$  possible subset models.
2. We fit each model  $m$  to the training data and obtain parameter estimates  $\hat{\beta}(m)$  with corresponding fitted values  $\hat{y}(m)_i, i = 1, \dots, n$

3.  $RSS(m) = \sum_{i=1}^n (y_i - \hat{y}(m)_i)^2$
4.  $MSE(m)_{train} = \frac{1}{n} RSS(m)$
5.  $pMSE(m) = MSE(m)_{test} = \frac{1}{n} \sum_{i=1}^n (y_i^{new} - \hat{y}(m)_i)^2$

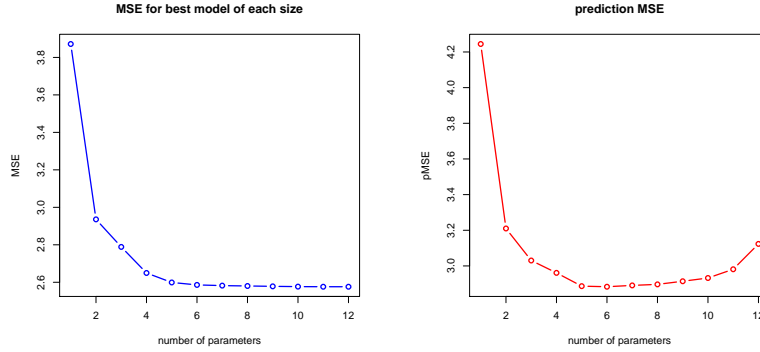


Figure 2: Left: RSS-curve. MSE on training data. Middle: pMSE-curve. MSE on test data.

In Figure 2 I mimic the training/testing setup from above. I split the data in half - use one half as training and reserve the other part for testing. (Note, this is not the exact setup from above since I am not creating a test data at the precise same  $x$  locations in the training data.)

I compare the MSE on the training data to the pMSE (MSE on the test data). Note that while MSE on the training data is minimized for the largest model, the pMSE decreases up to a point ( $p_{selected} = 6$ ) and then starts increasing. The pMSE decreases as long as the decrease in bias exceeds the increase in estimation variance. At some point, when we exceed the true model size, we are no longer gaining in terms of bias but estimation variance increases since we are including unnecessary parameters in our model. Note, while the optimum size model is here identified as the minimizer of pMSE, this model may not be the minimum MSE model of the same size (see Table 1).

	Intercept	age	sbp	adiposity	obesity	typea
pMSE win	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
MSE of same size	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE

	alcohol	alcind	tobacco	tobind	chd	famhist
pMSE win	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
MSE of same size	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE

Table 1: Selected variables using pMSE

### 3 Demo 7

We will examine training and prediction MSE using a data set on car prices and gas mileage (how many miles/km per gallons/liters). We first read the data into R.

```
> cars<-data.frame(read.table("cars.dat",header=T)) ## read in the data
> print(dim(cars)) ## dimensions

[1] 82 19

> print(names(cars)) ## variable names

[1] "mid.price"    "city.mpg"    "hw.mpg"      "airbagstd"   "cylnbr"
[6] "engsize"     "horsepwr"    "rpm.at.max"  "engrev.high" "mantrans.op"
[11] "fueltank"    "passengers"  "length"      "width"       "uturn"
[16] "rearroom"    "luggage"     "weight"      "domestic"
```

There are, as you see, 82 different cars in this data set and 19 variables. We will focus on the price of the cars as a function of the other variables, which include mileage (miles per gallon in the city and on highway (compare liters per km in Sweden)), airbag standard (0 if not included, 1 if for the driver and 2 if passenger/side), the number of cylinders, the engine size, horsepower, maximum rpm of engine, manual transmission (yes=1, no=0), size of the fuel tank, passenger room, length and width of the car, a measure about the space needed to make u-turn with the car, size of the rear room of the car, luggage room, weight of the car, and finally an indicator if the car is domestic (US built=1) or not.

Let us examine the data set. I first split the data into a random training set and a random test set for illustration purposes - this means that you will see different results when you run the demo yourselves.

```
> ## Creating a training and test data set.
> ntrain<-60
> ii<-sample(seq(1,dim(cars)[1]),ntrain)
> # ntrain cars = training set
> cars.train<-cars[ii,]
> row.names(cars.train)<-seq(1,ntrain)
> cars.test<-cars[-ii,]
> row.names(cars.test)<-seq(1,dim(cars.test)[1])
```

I first try some select pairwise scatter plots. Do this yourselves for other pairs of variables.

```
> par(mfrow=c(2,2))
> plot(cars.train$mid,cars.train$ci,main="citympg on price")
> plot(cars.train$hw,cars.train$ci,main="citympg on hwmpg")
> plot(cars.train$le,cars.train$ci,main="citympg on length")
> plot(cars.train$engsize,cars.train$ci,main="citympg on enginesize")
> p<-locator()

> par(mfrow=c(2,2))
> plot(cars.train$le,cars.train$wi,main="width on length")
> plot(cars.train$wi,cars.train$we,main="weight on width")
> plot(cars.train$le,cars.train$wh,main="wheelbase on length")
> plot(cars.train$lu,cars.train$rea,main="rearroom on luggage room")
> p<-locator()
```

In Figure 3 and 4 you can see that there are correlations between both the outcome and covariates as well as between the covariates themselves. We will focus on price as the outcome, but keep in mind that we have a collinearity problem.

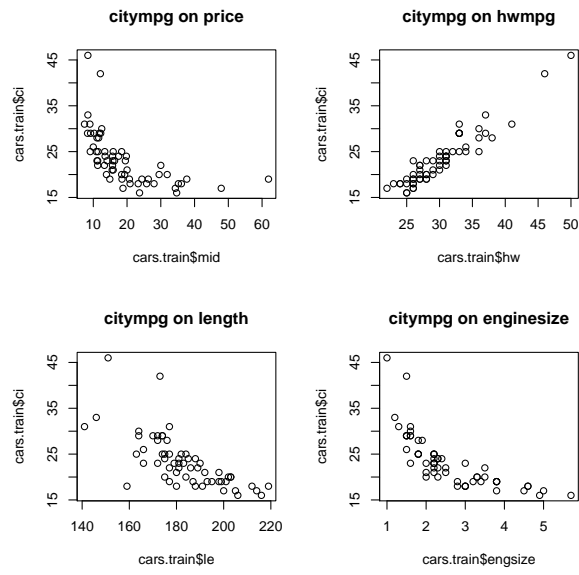


Figure 3: Scatter plots - 1

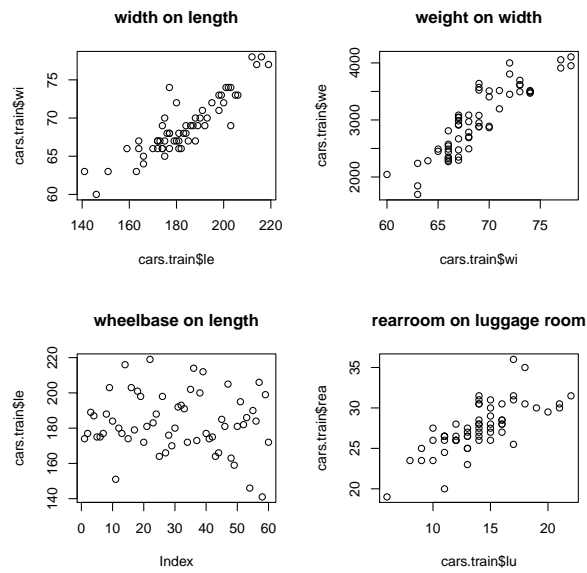


Figure 4: Scatter plots - 2

```

> par(mfrow=c(2,2))
> plot(cars.train$ci,cars.train$mid,main="price on citymg")
> plot(cars.train$we,cars.train$mid,main="price on weight")
> plot(cars.train$ho,cars.train$mid,main="price on horsepower")
> plot(cars.train$man,cars.train$mid,main="price on transmission")
> p<-locator()

```

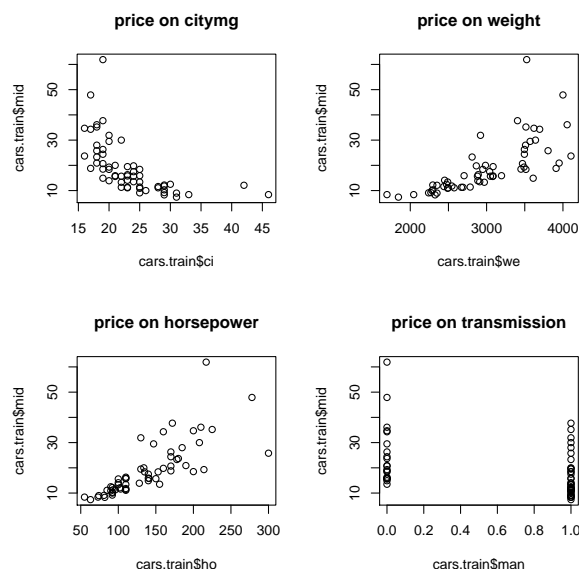


Figure 5: Price vs city milage, weight, horsepower and manual transmission.

In Figure 5 I depict some pairwise relationships between car prices and some other variables. Do you spot any problems with the modeling assumptions? I believe we have a skewed distribution for the outcome (salary, price and similar data often has this long-tailed appearance) and increasing variance with the expected value of the outcome. In addition, the relationship between price and mileage does not appear to be linear. I try some transformations (log of price, and inverse of mileage) to see if we can fix the problem.

```

> par(mfrow=c(2,2))
> plot(1/cars.train$ci,log(cars.train$mid),main="log(price) on 1/citympg")
> plot(cars.train$we,log(cars.train$mid),main="log(price) on weight")
> plot(cars.train$ho,log(cars.train$mid),main="log(price) on horsepower")
> plot(cars.train$man,log(cars.train$mid),main="log(price) on transmission")
> p<-locator()

```

From Figure 6 we see that the transformations did a pretty good job. Try some other transformations at home.

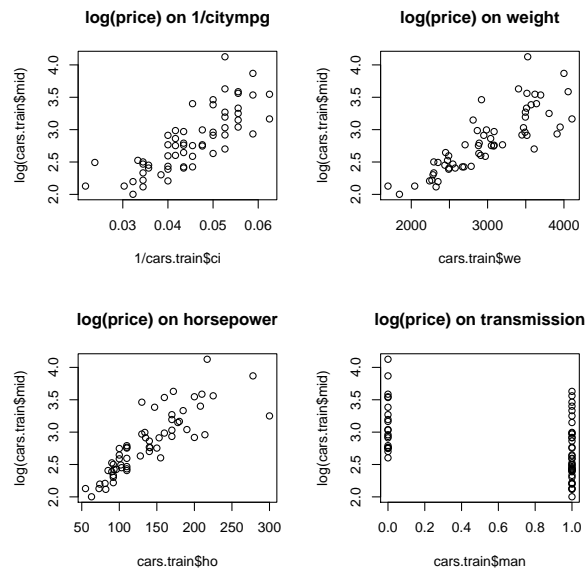


Figure 6:  $\log(\text{price})$  vs  $1/(\text{city milage})$ , weight, horsepower and manual transmission.



I decide to use this transformations and rename and transform objects in the data frame as follows:

```
> cars.train[,1]<-log(cars.train$mid)
> cars.train[,2]<-1/cars.train$ci
> cars.train[,3]<-1/cars.train$hw
> cars.test[,1]<-log(cars.test$mid)
> cars.test[,2]<-1/cars.test$ci
> cars.test[,3]<-1/cars.test$hw
> names(cars.train)[c(1,2,3)]<-c("log.mid.price", "city.gpm", "hw.gpm")
> names(cars.test)[c(1,2,3)]<-c("log.mid.price", "city.gpm", "hw.gpm")
```

We are now ready to try a linear model fit:

```
> ## Fitting the model after transforming the data
> mm<-lm(log.mid.price~city.gpm+hw.gpm+airbagstd+cylnbr+
+         engsize+horsepwr+rpm.at.max+engrev.high+mantrans.op+
+         fueltank+passengers+length+width+uturn+rearroom+
+         luggage+weight+domestic,data=cars.train)
> print(ms<-summary(mm))
```

Call:

```
lm(formula = log.mid.price ~ city.gpm + hw.gpm + airbagstd +
    cylnbr + engsize + horsepwr + rpm.at.max + engrev.high +
    mantrans.op + fueltank + passengers + length + width + uturn +
    rearroom + luggage + weight + domestic, data = cars.train)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.35395	-0.10777	0.01436	0.08027	0.39819

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	3.267e+00	1.459e+00	2.240	0.03059 *
city.gpm	3.416e+01	1.194e+01	2.860	0.00663 **
hw.gpm	-2.672e+01	1.818e+01	-1.470	0.14927
airbagstd	1.607e-01	4.999e-02	3.214	0.00255 **
cylnbr	6.568e-02	6.102e-02	1.076	0.28810
engsize	-5.701e-02	1.130e-01	-0.505	0.61650
horsepwr	2.911e-03	2.138e-03	1.362	0.18077
rpm.at.max	-8.104e-06	1.082e-04	-0.075	0.94065
engrev.high	1.569e-04	1.074e-04	1.461	0.15166
mantrans.op	-1.137e-01	9.485e-02	-1.199	0.23741
fueltank	-1.382e-02	2.655e-02	-0.521	0.60538
passengers	4.244e-02	7.978e-02	0.532	0.59760
length	1.987e-03	5.435e-03	0.366	0.71654
width	-5.005e-02	2.245e-02	-2.229	0.03132 *
uturn	4.875e-04	1.747e-02	0.028	0.97788
rearroom	1.850e-03	1.658e-02	0.112	0.91172
luggage	-9.086e-03	1.972e-02	-0.461	0.64740
weight	4.131e-04	2.970e-04	1.391	0.17181
domestic	-1.934e-01	8.393e-02	-2.304	0.02634 *

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.195 on 41 degrees of freedom

Multiple R-squared: 0.8846, Adjusted R-squared: 0.8339

F-statistic: 17.45 on 18 and 41 DF, p-value: 8.647e-14

```
> par(mfrow=c(2,2))
> plot(mm)
> p<-locator()
```

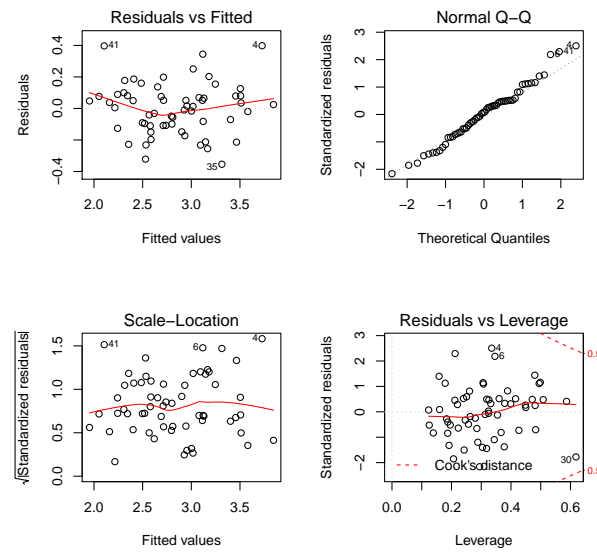


Figure 7: Diagnostic plots

From the model summary we can see that the R-squared is 0.885. Check the model summary for the Goodness-of-fit test and the p-values of individual coefficients. Are there any surprises? Do you see the collinearity problem having an impact in this model summary? In Figure 7 I display the standard diagnostic plots. We will revisit them one by one below, in addition to some other plots, to check for outliers.

## Identifying outliers

In the plots below, I identify potential outliers in the data set.

```
> ## Checking the residuals.  
> par(mfrow=c(1,1))  
> plot(mm$fit,mm$res,xlab="fitted values",ylab="residuals")  
> abline(h=0)  
> id1<-identify(mm$fit,mm$res,pos=T)
```

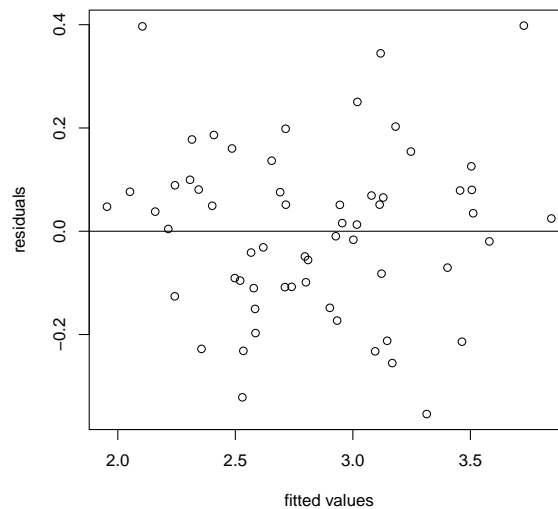


Figure 8: Residuals vs fitted values

```
> ## Checking normal error assumption  
> qq<-seq(.5/ntrain,(ntrain-.5)/ntrain,length=ntrain)  
> normq<-qnorm(p=qq)  
> rsort<-sort(rstandard(mm))  
> rlist<-sort.list(rstandard(mm))  
> plot(normq,rsort,xlab="Theoretical quantiles",ylab="Standardized residuals")  
> qr<-quantile(rstandard(mm))  
> qn<-quantile(qnorm(p=qq))  
> b<-(qr[4]-qr[2])/(qn[4]-qn[2])  
> a<-qr[4]-b*qn[4]  
> abline(a,b)  
> id2<-identify(normq,sort(rstandard(mm)),label=rlist,pos=T)  
  
> ## Checking constant error variance using absolute residuals  
> plot(mm$fit,abs(rstandard(mm)),xlab="fitted values", ylab="|standardized residuals|")  
> id3<-identify(mm$fit,abs(rstandard(mm)),pos=T)  
  
> ## Checking the Cooks distance  
> lm1<-lm.influence(mm)  
> cooks<-cooks.distance(mm)  
> plot(cooks,main="Cooks Distance",type="h")  
> abline(h=qf(.95,1,mm$df),lty=2)  
> idc<-identify(cooks,pos=T)
```

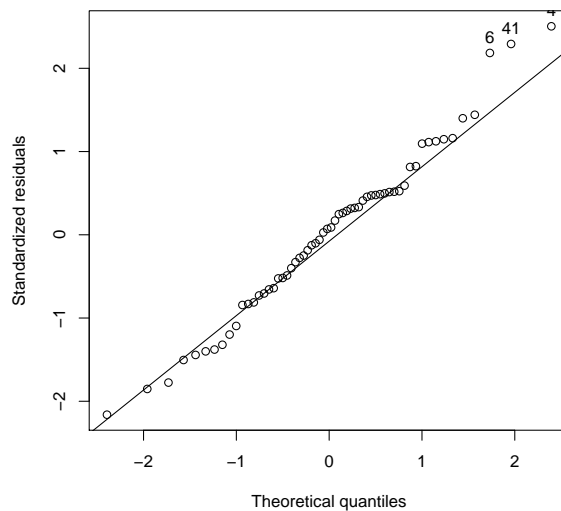


Figure 9: QQplot

```
> ## Checking leverage and impact on slopes
> plot(lm1$hat,main="Leverage")
> idlev<-identify(lm1$hat,pos=T)
```

We also examine the leverage and what the effect is of dropping an observation, on both slope estimates and the standard error estimate  $\hat{\sigma}$ .

```
> plot(lm1$coeff[,4],main="change in slope 4")
> id4<-identify(lm1$coeff[,4],pos=T)

> plot(lm1$coeff[,7],main="change in slope 7")
> id7<-identify(lm1$coeff[,7],pos=T)

> plot(lm1$sig,main="change in sigma")
> ids<-identify(lm1$sig,pos=T)
```

We collect on the outlier information and drop the observation most frequently identified (or more than one if they equally frequently identified):

```
> indvec<-sort(c(id1$ind,rlist[id2$ind],id3$ind,idlev$ind,id4$ind,id7$ind,ids$ind))
> print(table(indvec)) ## how many of each?

indvec
 4  6 30 35 41 45
 4  1  1  2  1  1

> maxid<-max(table(indvec))
> indout<-unique(indvec)[table(indvec)==max(table(indvec))]
```

Here, this is observation 4. Note, when you do this yourselves you may not need to drop an observation, or perhaps drop several. If the latter, just run the code again after dropping the first observation or pick more than one to drop in one go by manually adjusting `indout`

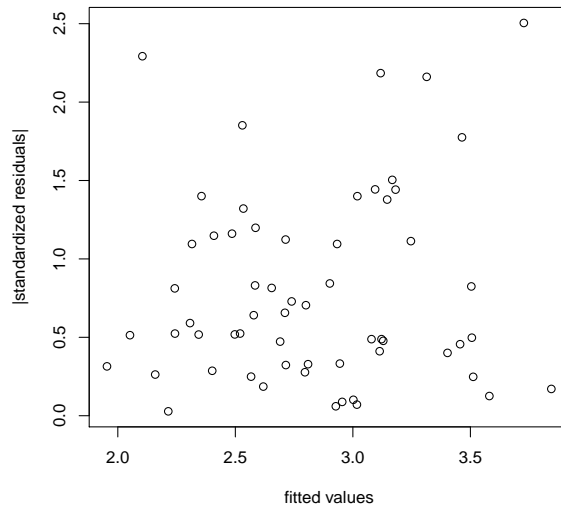


Figure 10: Absolute standardized residuals vs fitted values

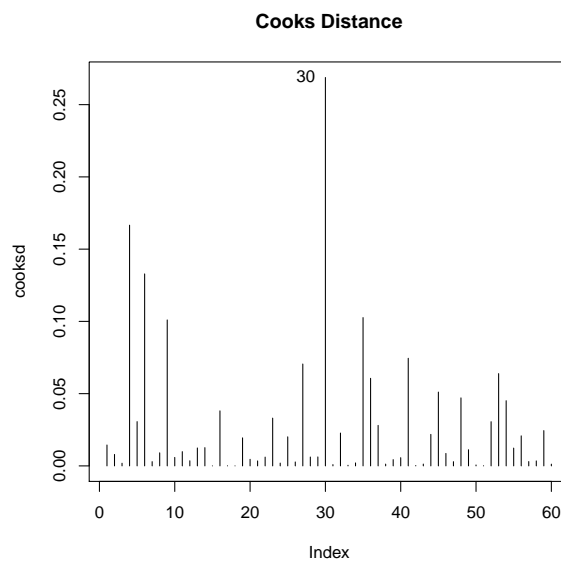


Figure 11: Cook's distance:  $D_i = e_i^2 \frac{h_{ii}}{p * MSE * (1 - h_{ii})^2}$

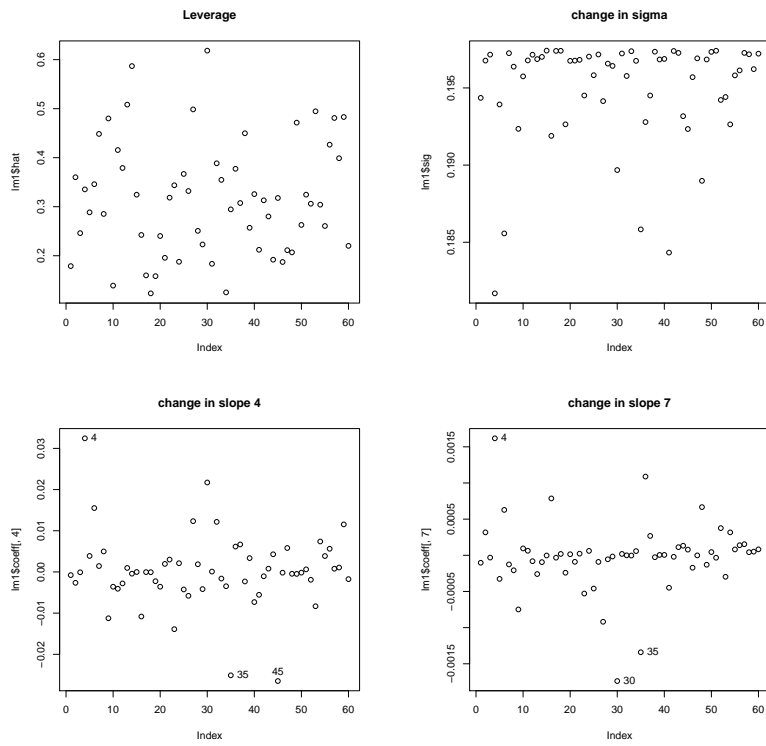


Figure 12: Top Left: Leverage. Top Right: Effect on standard error. Lower panels: impact on slope estimates.

## Updating the model

We can now update the model without the outlier:

```
> ## regression without the identified outlier
> mmb<-lm(log.mid.price~city.gpm+hw.gpm+airbagstd+cylnbr+
+         engsize+horsepwr+rpm.at.max+engrev.high+mantrans.op+
+         fueltank+passengers+length+width+uturn+rearroom+
+         luggage+weight+domestic,data=cars.train,subset=-indout)
> print(summary(mmb))
```

Call:

```
lm(formula = log.mid.price ~ city.gpm + hw.gpm + airbagstd +
    cylnbr + engsize + horsepwr + rpm.at.max + engrev.high +
    mantrans.op + fueltank + passengers + length + width + uturn +
    rearroom + luggage + weight + domestic, data = cars.train,
    subset = -indout)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.32186	-0.08921	0.01532	0.08328	0.41046

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.220e+00	1.414e+00	1.570	0.12429
city.gpm	3.622e+01	1.116e+01	3.247	0.00236 **
hw.gpm	-3.289e+01	1.709e+01	-1.924	0.06151 .
airbagstd	1.282e-01	4.812e-02	2.665	0.01106 *
cylnbr	8.259e-02	5.721e-02	1.444	0.15661
engsize	-2.772e-02	1.058e-01	-0.262	0.79473
horsepwr	1.294e-03	2.081e-03	0.622	0.53774
rpm.at.max	5.394e-05	1.034e-04	0.522	0.60481
engrev.high	1.676e-04	1.001e-04	1.674	0.10199
mantrans.op	-2.667e-02	9.413e-02	-0.283	0.77840
fueltank	-1.274e-02	2.474e-02	-0.515	0.60936
passengers	4.376e-02	7.434e-02	0.589	0.55938
length	-4.363e-04	5.144e-03	-0.085	0.93283
width	-4.385e-02	2.105e-02	-2.083	0.04365 *
uturn	1.194e-02	1.683e-02	0.709	0.48218
rearroom	6.815e-03	1.556e-02	0.438	0.66381
luggage	-1.879e-02	1.872e-02	-1.003	0.32176
weight	5.341e-04	2.804e-04	1.905	0.06400 .
domestic	-1.493e-01	7.991e-02	-1.869	0.06902 .

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1817 on 40 degrees of freedom

Multiple R-squared: 0.888, Adjusted R-squared: 0.8376

F-statistic: 17.62 on 18 and 40 DF, p-value: 1.214e-13

```
> par(mfrow=c(2,2))
> plot(mmb)
> p<-locator()
```

In Figure 13 we can see the diagnostic plot after we drop the outlier(s). Compare with Figure 7 above. You should also compare the model summary: has the fit improved? in what sense?

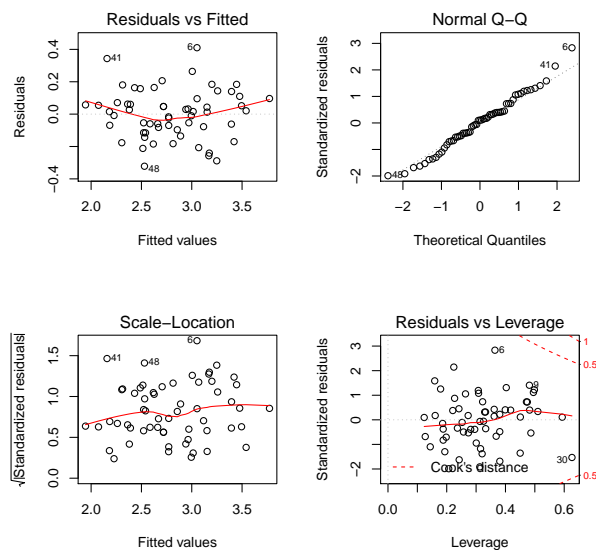


Figure 13: Updated model fit: diagnostic plots

## Stepwise model selection

We try a step wise model selection with the updated model as the starting point. Note, the option `trace=F` tells R to only show us the final model.

```
> selectstep<-step(mmb,trace=F)
```

We can use the selected model for prediction. First, we look at the updated model summary after selection:

```
> print(summary(selectstep))
```

Call:

```
lm(formula = log.mid.price ~ city.gpm + hw.gpm + airbagstd +
    cylnabr + rpm.at.max + engrev.high + width + weight + domestic,
    data = cars.train, subset = -indout)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.32642	-0.11187	0.00758	0.11179	0.43270

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.336e+00	1.088e+00	2.147	0.036746 *
city.gpm	3.256e+01	9.568e+00	3.403	0.001338 **
hw.gpm	-2.402e+01	1.304e+01	-1.843	0.071424 .
airbagstd	1.549e-01	3.992e-02	3.879	0.000313 ***
cylnabr	7.312e-02	3.447e-02	2.121	0.038972 *
rpm.at.max	1.165e-04	5.263e-05	2.213	0.031563 *
engrev.high	1.163e-04	8.106e-05	1.435	0.157651
width	-4.460e-02	1.735e-02	-2.570	0.013252 *
weight	5.341e-04	1.398e-04	3.821	0.000376 ***
domestic	-1.066e-01	5.816e-02	-1.834	0.072789 .

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1714 on 49 degrees of freedom



Multiple R-squared: 0.8779, Adjusted R-squared: 0.8555  
F-statistic: 39.14 on 9 and 49 DF, p-value: < 2.2e-16

To compute predictions for the test data we do the following:

```
> predval<-predict(selectstep,newdata=cars.test) ## Prediction
> prederror<-sum((cars.test[,1]-predval)^2) ## Prediction MSE
> ## Note that the outcome Price is in the first column of
> ## cars.test. If you use this code on other data sets, make
> ## sure you keep track of which column contains your y-variable.
```

For comparison, what happens when we try to fit a model to the test data? We use the full model fitted to the test data as a starting point for backward model selection. We compute the training error (fiterror) below for the selected model on the test data (this is just the residual sum of square for the backward selected model on the test data).

```
> # compare refit
> mmtest<-lm(log.mid.price~city.gpm+hw.gpm+airbagstd+cylnbr+
+           engsize+horsepwr+rpm.at.max+engrev.high+mantrans.op+
+           fueltank+passengers+length+width+uturn+rearroom+
+           luggage+weight+domestic,data=cars.test)
> print(selecttest<-step(mmtest,trace=F))
```

Call:

```
lm(formula = log.mid.price ~ city.gpm + hw.gpm + airbagstd +
    cylnbr + horsepwr + rpm.at.max + engrev.high + mantrans.op +
    fueltank + passengers + length + width + uturn + rearroom +
    luggage + weight + domestic, data = cars.test)
```

Coefficients:

(Intercept)	city.gpm	hw.gpm	airbagstd	cylnbr
-3.954e+01	-8.646e+01	1.710e+02	-4.434e-01	1.086e+00
horsepwr	rpm.at.max	engrev.high	mantrans.op	fueltank
-3.131e-02	4.000e-04	2.633e-03	2.224e+00	1.298e-01
passengers	length	width	uturn	rearroom
-1.717e-01	4.134e-02	-1.780e-01	4.632e-01	3.653e-01
luggage	weight	domestic		
2.124e-01	6.904e-04	1.199e+00		

```
> fiterror<-sum(summary(selecttest)$res^2)
```

Compare the selected models on the training and test data sets. Are they the same?

We can also compare the prediction error to the error sum of squares obtained from the selected model on the test data. Here, the prediction error (`prederror` above), using the selected model from the training data, is 0.908, whereas the error sum of squares, using the test data both for model fitting and prediction (`fiterror` above) is 0.035. What does this tell you? The MSE is much smaller than the prediction error since the test data was not used to guide the model estimation. That's the basis for using prediction for model selection.

## Using prediction error as a model selection tool

Let us find out which model is actually best for prediction (on *this* test data). We first enumerate and fit all models to the training data and collect the MSEs for this model fits.

```
> yy<-cars.train[,1] ## training data
> xx<-as.matrix(cars.train[,-1])
> yyt<-cars.test[,1] ## test data
> xxt<-as.matrix(cars.test[,-1])
> ###
> #install.packages("leaps") ## You only have to do this once
```

```

> library(leaps) ## But you have to do this everytime you start a new session
> rleaps<-regsubsets(xx,yy,int=T,nbest=500,nvmax=dim(cars)[2],really.big=T,method=c("ex"))
> ## all subset models
> cleaps<-summary(rleaps,matrix=T)
> ## True/False matrix. The r-th row is a True/False statement
> ## about which variables are included in model r.
> tt<-apply(cleaps$which,1,sum) ## size of each model in the matrix
> mses<-cleaps$rss/length(yy) ## corresponding MSEs

```

We then plot the MSE for all the models as well as the MSE-curve (best models of each model size).

```

> ## First add the intercept-only model to the list
> tt<-c(1,tt)
> nullrss<-sum((yy-mean(yy))^2)/length(yy)
> mses<-c(nullrss,mses)
> plot(tt,mses,xlab="number of parameters",ylab="MSE",main="MSE for all subset models")
> tmin<-min(tt) ## smallest model tried
> tmax<-max(tt) ## biggest model tried
> tsec<-seq(tmin,tmax)
> msevec<-rep(0,length(tsec))
> for (tk in 1:length(tsec)) {
+   msevec[tk]<-min(mses[tt==tsec[tk]])} ## the best model for each size
> lines(tsec,msevec,lwd=2,col=2) ## a line connecting the best models.
> p<-locator()

> plot(tsec,msevec,xlab="number of parameters",
+       ylab="MSE",main="MSE for best model of each size",type="b",col=4,lwd=2)
> p<-locator()

```

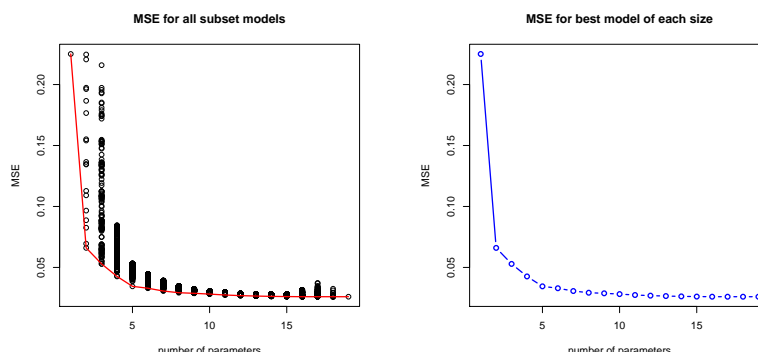


Figure 14: Left: MSE for all models fitted. Right: Best MSE models for each model size.

In Figure 14 you see that the MSE always decreases with model size.

We then apply all fitted models to the test data (without refitting, no updating of model coefficients) and collect pMSE values for each.

```

> pmses<-rep(0,dim(cleaps$which)[1]) ## creates an empty vector to store pMSE in
> for (ta in (1:dim(cleaps$which)[1])) {
+   mmr<-lm(yy~xx[,cleaps$which[ta,-1]==T]) ## Fit each of the stored models to training data
+   Xmat<-cbind(rep(1,dim(xxt)[1]),xxt[,cleaps$which[ta,-1]==T])
+   PEcp<-sum((yyt-Xmat%*%mmr$coef)^2)/length(yyt) ## Compute the prediction error
+   ## Note, I create the design matrix Xmat for the current model by
+   ## including a column of 1s first (for the intercept) and then
+   ## tagging on xxt[,cleaps$which[ta,-1]==T] which contains the
+   ## x-variables for which row ta of the cleaps$which model matrix has T for True.

```

```

+ pmses[ta]<-PEcp }
> nullpmse<-sum((yyt-mean(yy))^2)/length(yyt)
> pmses<-c(nullpmse,pmses)
> pmsevec<-rep(0,length(tsec))
> for (tk in 1:length(tsec)) {
+ pmsevec[tk]<-min(pmses[tt==tsec[tk]])} ## winning model for each size

> plot(tsec,pmsevec,xlab="number of parameters",
+      ylab="pMSE",main="prediction MSE", type="b",lwd=2,col=2)

```

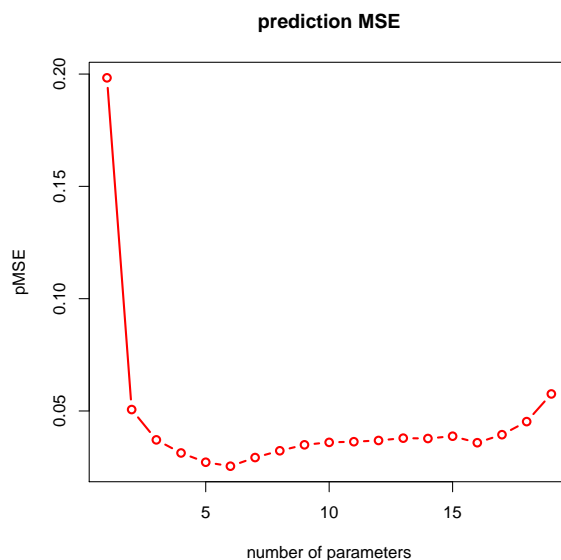


Figure 15: pMSE curve for the cars data

We plot the pMSE values as a function of model size. Compare the pMSE curve in Figure 15 to the MSE curve in Figure 14. What do you see?

Finally, we want to compare the selected model (from training using backward selection) and the selected model that works best in terms of prediction.

```

> ptmin<-which.min(pmses)
> ## which model has the smallest pmse?
> ModMat<-rbind(c("TRUE", rep("FALSE",dim(cleaps$which)[2]-1)),cleaps$which)
> # First row of ModMat is the Intercept only model
> pmod<-ModMat[ptmin,]
> winsize<-sum(pmod==T)
> ## Winning model
> print(names(pmod[pmod==T])[-1])

[1] "city.gpm" "airbagstd" "fueltank" "width" "weight"

> print(names(selectstep$model)[-1])

[1] "city.gpm" "hw.gpm" "airbagstd" "cylnbr" "rpm.at.max"
[6] "engrev.high" "width" "weight" "domestic"

> ## Compare to backward selection model

```

In the above summary you can compare the model selected on training data and the model that was actually best for prediction on *this* test data. What do you see? Repeat this exercise a couple of times and try to summarize your findings. Is the same model always best for prediction for all test data sets? What varies? The model size or the included variables or both?