

Regression

Statistical Learning, BGC - 2011

Niels Richard Hansen
September 18, 2011

Disclaimer: All the R handouts, written in Sweave, for this course are exactly R handouts. That is, it is R code with comments and results, and by no means a final report or a template for a final report.

1 Standard analysis of prostate cancer data

The objective is to find a model of `lpsa` (log prostate-specific antigen) given 8 different predictors. The original motivation seems to be to understand how a (simple) measurement of the level of prostate-specific antigen is related to various quantifications of the size of the prostate and the state of prostate cancer. The `age` variable is included as a possible confounder.

The data can be loaded directly into R from the web page for the book. Subsequently we select a subset, called the training data set, which will be used for estimation, and we center the observations.

```
> prostate <- read.table("http://www-stat-  
class.stanford.edu/%7Etibs/ElemStatLearn/datasets/prostate.data")  
  
> prostateTrain <- prostate[prostate$train, 1:9]  
> prostateCentered <- scale(prostateTrain, scale=FALSE)
```

We use `lm.fit` directly to compute the least squares estimate when the data are in the given form of an X -matrix and the Y -vector response.

```
> prostateLmFit <- lm.fit(x = prostateCentered[, -9],  
+                         y = prostateCentered[, 9]  
+                         )
```

In the alternative, we can use the `lm` function with a formula interface by converting `prostateCentered` back to a data frame. We subsequently compare the coefficients obtained.

```
> prostateLm <- lm(lpsa ~ . - 1,  
+                 data = as.data.frame(prostateCentered)  
+                 )  
> identical(prostateLmFit$coef, coefficients(prostateLm))
```

[1] TRUE

	1	2
lcavol	0.58	0.58
lweight	0.61	0.61
age	-0.02	-0.02
lbph	0.14	0.14
svi	0.74	0.74
lcp	-0.21	-0.21
gleason	-0.03	-0.03
pgg45	0.01	0.01

Table 1: Estimates from `lm.fit` and `lm`, respectively

Note that we can use the *generic* accessor method `coefficients` – in addition to a range of other methods – on the `lm` object returned by `lm`, whereas the object returned by `lm.fit` is just a plain list with no particular methods associated.

```
> summary(prostateLm)
```

	Estimate	Std. Error	t value	Pr(> t)
lcavol	0.5765	0.1065	5.41	0.0000
lweight	0.6140	0.2213	2.77	0.0074
age	-0.0190	0.0135	-1.41	0.1644
lbph	0.1448	0.0699	2.07	0.0425
svi	0.7372	0.2960	2.49	0.0156
lcp	-0.2063	0.1096	-1.88	0.0646
gleason	-0.0295	0.1994	-0.15	0.8829
pgg45	0.0095	0.0054	1.75	0.0848

Another thing that one could look at is the collection of marginal models where we only include one of the regressors at the time.

```
> prostateAdd <- add1(lm(lpsa ~ 1,
+                         data = as.data.frame(prostateCentered)
+                         ),
+                      scope = formula(prostateLm),
+                      test = "F"
+                      )
```

We include the intercept again in the estimation to get a table comparable to Table 3.2 in the book. This is a little messy as we need the y -variable to be un-centered but the remaining variables to be centered. Note, however, that all estimates are the same except that there is an intercept now. This is because the centering makes all the x -variables orthogonal to the column of ones.

```
> prostateLm2 <- lm(lpsa ~ .,
```

	Df	RSS	F value	Pr(F)
<none>		96.28		
lcavol	1	44.53	75.55	0.00
lweight	1	73.61	20.02	0.00
age	1	91.29	3.55	0.06
lbph	1	89.62	4.83	0.03
svi	1	66.42	29.22	0.00
lcp	1	73.24	20.45	0.00
gleason	1	84.99	8.63	0.00
pgg45	1	76.95	16.33	0.00

```
+ data = cbind(prostateCentered[, 1:8],
+             prostateTrain[, 9, drop = FALSE])
+ )
> summary(prostateLm2)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.4523	0.0870	28.18	0.0000
lcavol	0.5765	0.1074	5.37	0.0000
lweight	0.6140	0.2232	2.75	0.0079
age	-0.0190	0.0136	-1.40	0.1681
lbph	0.1448	0.0705	2.06	0.0443
svi	0.7372	0.2986	2.47	0.0165
lcp	-0.2063	0.1105	-1.87	0.0670
gleason	-0.0295	0.2011	-0.15	0.8839
pgg45	0.0095	0.0054	1.74	0.0875

Note that the mean of the y -variable equals the intercept due to the centering of the x -variables.

```
> mean(prostateTrain[, 9])
```

```
[1] 2.452345
```

You will observe that the above table is **not** identical to Table 3.2. The reason is that in the analysis in the book they center (and scale) the x -variables *before* selecting the training dataset. This could be done as follows:

```
> prostateScaled <- cbind(scale(prostate[, 1:8]),
+                         prostate[, 9, drop=FALSE]
+                         )[prostate$train, ]
> prostateLm3 <- lm(lpsa ~ .,
+                  data = prostateScaled
+                  )
> summary(prostateLm3)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.4649	0.0893	27.60	0.0000
lcavol	0.6795	0.1266	5.37	0.0000
lweight	0.2631	0.0956	2.75	0.0079
age	-0.1415	0.1013	-1.40	0.1681
lbph	0.2101	0.1022	2.06	0.0443
svi	0.3052	0.1236	2.47	0.0165
lcp	-0.2885	0.1545	-1.87	0.0670
gleason	-0.0213	0.1452	-0.15	0.8839
pgg45	0.2670	0.1536	1.74	0.0875

It is not a mistake in the book, but it seems a little unfortunate that the scaling is not done based on the training set alone. One issue is that the column vector of ones in their centering is not completely orthogonal to the centered x -columns in the training dataset – only in the full dataset. A consequence of this is that the intercept is not exactly equal to the average of the y -values.

For other models where the fitted model is not invariant to centering or scaling we would generally regard scaling as part of the model fitting process and it should always be done on the training data alone.

2 Best subset selection

We use the `leaps` package to do best subset selection.

```
> if(!require(leaps))
+   stop(paste("You don't have package leaps installed"))
```

The function `regsubsets` computes the fits for all subsets of models. Some post processing of the results is done before plotting.

```
> prostateSubset <- regsubsets(prostateScaled[, -9],
+                             prostateScaled[, 9],
+                             nbest = 70,
+                             really.big = TRUE)
> prostateRss <- prostateSubset$ress
> prostateRss <- reshape(as.data.frame(prostateRss),
+                         varying = list(1:70),
+                         direction = "long")[, c(3,2)]
> names(prostateRss) <- c("k", "RSS")
> prostateRss <- prostateRss[prostateRss$RSS < 1e35, ]
```

```
> if(require(ggplot2)) {
+   p <- qplot(prostateRss$k-1,
+              prostateRss$RSS,
+              shape = I(19),
```

```

+         colour = I("gray50"),
+         ylim = c(0, 100),
+         xlab = "Subset size k",
+         ylab = "Residual sum-of-squares") +
+         geom_line(aes(x = 0:8, y = prostateSubset$ress[, 1]),
+                   colour = "red") +
+         geom_point(aes(x = 0:8, y = prostateSubset$ress[,1]),
+                   shape = 19, colour = "red")
+   print(p)
+ } else {
+   plot(prostateRss$k-1,
+        prostateRss$RSS,
+        pch = 19,
+        col = "gray50",
+        ylim = c(0, 100),
+        xlab = "Subset size k",
+        ylab = "Residual sum-of-squares")
+   lines(0:8, prostateSubset$ress[, 1], col = "red", type = "b", pch = 19)
+ }

```

