MSG500/MVE190
Linear Models - Lecture 9

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

November 22, 2012

1 RECAP

- Ultimate validation of a model is to test its predictive capacity
- Prediction performance is affected by both Bias and Estimation variance
- We combine the two into the criterion *prediction* MSE = *Bias*^{2} + *Estimation Variance*
- We can't compute this in real life situations since we don't know the true model (needed to compute the bias)
- With training and test data we can separate model estimation from model validation (prediction)
- Training data: compute $MSE_{train} = RSS/n$ (the fit of the model on the data used to estimate model parameters)
- Test data: compute $pMSE = MSE_{test}$ (the fit of the model to new data *not* used for estimation).
- The $pMSE$ is a substitute for the real *prediction MSE* as defined above.
- We select the model that minimizes the $pMSE$
- We usually don't have separate training and test data, and use *cross-validation* to mimic this scenario, and to estimate the *prediction MSE*

2 Model Selection

Usually, parsimonious, or simple models work best for prediction. Why is that? Well, it's easier to estimate parameters of a simple model with limited amounts of data. We limit the risk of including a spurious relationships that do not generalize to future data. This preference for as simple an explanation of the data as possible is sometimes referred to as *Occam's Razor* - a simple model is 'safer' in terms of prediction performance and is also easier to interpret.

2.1 Caution

Most model selection procedures work with a global criterion based on e.g. RSS. Don't forget to check the model fit using diagnostic plots though! The selected model is not OK if you see trends or patterns in the residuals and the presence of outliers can have a huge impact on the RSS.

3 Optimism and Model selection criteria

We review the setup form previous lectures:

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

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

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

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

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

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

$pMSE(m) - MSE(m)$ is called the *optimism* for model m. That is, because we fit the model m to match the training data as best possible using the least squares criterion, the $MSE(m)$ is an underestimate of how well the model would perform on future data. We can estimate this optimism directly using training and test data we create from our observed data set. We did this in lecture 8, using a technique called *cross-validation.*

In this lecture we will try to estimate the *expected value* of the optimism, or gap between $pMSE$ and MSE directly. This is the basis for several commonly used model selection criteria in statistics, the AIC, BIC and Mallow's Cp.

4 Deriving Mallow's Cp

Let us take a closer look at the gap between the $pMSE$-curve and the MSE-curve from training data.

The *Prediction Error* of model m is defined as

$$PE(m) = E[\frac{1}{n}\sum_{i=1}^n (y_i^{new} - \hat{y}(m)_j)^2] = \frac{1}{n}\sum_{i=1}^n \underbrace{E[y_i^{new} - \hat{y}(m)_i]^2}_{***}$$

The expectation is taken over the new test data and the training data. The prediction $\hat{y}(m)_i$ is a function of the training data only. We expand the term *** above as follows:

$$*** = E\big[y_i^{new} - E(y_i^{new}) + E(y_i^{new}) - E(\hat{y}(m)_i) + E(\hat{y}(m)_i) - \hat{y}(m)_i\big]^2 =$$

$$= \underbrace{E\big[y_i^{new} - E(y_i^{new})\big]^2}_{(1)} + \underbrace{E\big[E(y_i^{new}) - E(\hat{y}(m)_i)\big]^2}_{(2)} + \underbrace{E\big[E(\hat{y}(m)_i) - \hat{y}(m)_i\big]^2}_{(3)} +$$

$$+ \underbrace{2E\big[\big(y_i^{new} - E(y_i^{new})\big)\big(E(y_i^{new}) - E(\hat{y}(m)_i)\big)\big]}_{(4)} + \underbrace{2E\big[\big(y_i^{new} - E(y_i^{new})\big)\big(E(\hat{y}(m)_i) - \hat{y}(m)_i\big)\big]}_{(5)} +$$

$$+ \underbrace{2E\big[\big(E(y_i^{new}) - E(\hat{y}(m)_i)\big)\big(E(\hat{y}(m)_i) - \hat{y}(m)_i\big)\big]}_{(6)}$$

We will work through each of the 6 terms above.

(1) * $E[y_i^{new} - E(y_i^{new})]^2 = \sigma^2$.
* The *irreducible error.*
* The noise or random scatter around the true model.

(2) * $E\big[E(y_i^{new}) - E(\hat{y}(m)_i)\big]^2 = bias^2$
* this is the local bias of model m at location x_i
* if model m is adequate the bias is 0

(3) * $E\big[E(\hat{y}(m)_i) - \hat{y}(m)_i\big]^2 = V[\hat{y}(m)_i]$
* the estimation variance of model m
* increases with the complexity of the model

(4) * This term is 0 since the constant $\big(E(y_i^{new}) - E(\hat{y}(m)_i)\big)$ can be pulled outside the outer expectation and $E\big[(y_i^{new} - E(y_i^{new})\big] = 0$

(5) * This term is 0 since y_i^{new} and y_i are uncorrelated
* $(5) = E\big[(y_i^{new} - E(y_i^{new})\big]E\big[E(\hat{y}(m)_i) - \hat{y}(m)_i\big] = 0$

(6) * This term is 0 since the constant $\big(E(y_i^{new}) - E(\hat{y}(m)_i)\big)$ can be pulled outside the outer expectation and $E\big[E(\hat{y}(m)_i) - \hat{y}(m)_i\big] = 0$

We summarize our findings:

$$PE(m) = \frac{1}{n}\sum_{i=1}^{n}(\sigma^2 + bias^2(i,m) + V[\hat{y}(m)_k]) = \sigma^2 + \bar{bias}^2(m) + \frac{1}{n}\sum_{i=1}^{n} V[\hat{y}(m)_i]$$

Example - linear model

What if $E[y_i] = \sum_{j=0}^{n} \beta_j x_{ij}$?

- $\hat{\beta}(m) = (X'X)^{-1}X'y$, where X is the $n \times p(m)$ *design* matrix
- If the model m with $p(m)$ variables is adequate there is no bias
- $V[\hat{y}] = \sigma^2 H$
- $V[\hat{y}_i] = \sigma^2 h_{ii} \to \frac{1}{n}V[\hat{y}_i] = \frac{\sigma^2}{n}\sum_i h_{ii} = \frac{\sigma^2}{n}Trace(H)$
- $Trace(H) = Trace(X(X'X)^{-1}X') = Trace((X'X)^{-1}(X'X)) = Trace(I_{p(m)}) = p(m)$

So, for a linear model with $E[y_i] = \sum_{j=0}^{n} \beta_j x_{ij}$ we have

$$PE(m) = \sigma^2\big(1 + \frac{p(m)}{n}\big)$$

4.1 The training error

What about the training error? What can we *expect* the RSS (or the training MSE) to be across multiple data sets from the same underlying, true model?

$$TE(m) = E\big[\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}(m)_i)^2\big] = \frac{1}{n}\sum_{i=1}^{n} E\big[y_i - \hat{y}(m)_i\big]^2$$

Similar to the above, we expand the expression $E\big[y_i - \hat{y}(m)_i\big]^2$ into 6 terms but now term (5) is *not* zero:

$$\textbf{(5)} \; = 2E\big[(y_i - E(y_i)(E(\hat{y}(m)_i) - \hat{y}(m)_i)\big] = 2E\big[y_i(E[\hat{y}(m)_i] - \hat{y}(m)_i)\big] =$$
$$= 2E[y_i]E[\hat{y}(m)_i] - 2E[y_i\hat{y}(m)_i] = -2Cov(y_i, \hat{y}(m)_i)$$

So, taken together we have that the *expected* training error is

$$TE(m) = PE(m) - \frac{2}{n}\sum_{i=1}^{n} Cov(y_i, \hat{y}(m)_i).$$

We have thus learnt that the training error is always less than the prediction error ($TE(m) < PE(m)$). In addition, the training error (TE) is much smaller than the prediction error (PE) if y_i and $\hat{y}_i$ are highly correlated. This is exactly what happens when we fit complex models to data: we match data to the model closely by adding parameters to our model. That means that the gap between the training error (TE) (expected MSE) and the prediction error (PE) (expected prediction MSE) increases with the complexity, or size, of the model we fit. As you can see from the above

$$gap(PE, TE) = \frac{2}{n} \sum_{i=1}^{n} Cov(y_i, \hat{y}(m)_i).$$

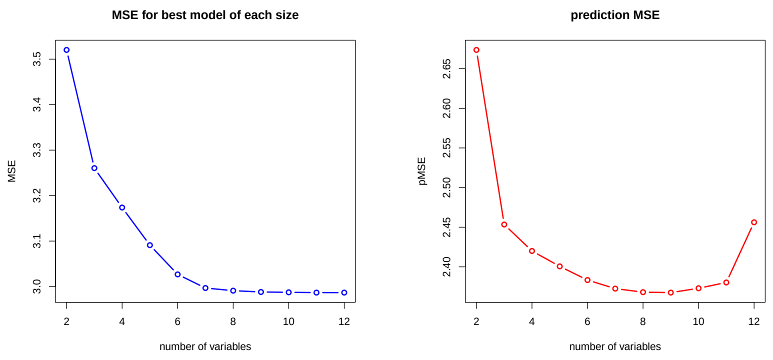

Figure 1: Left: RSS-curve. MSE on training data. Middle: pMSE-curve. MSE on test data.

In Figure 1 we depict the MSE and pMSE for a data example. For large models (where the bias is small) we see that the gap between the two curves increases with the size of the model.

Example: For a Least Squares linear models fit we can write $\hat{y} = Hy$ and $Cov(y_i, \hat{y}_i) = \sigma^2 h_{ii}$. We can thus write

$$gap(PE, TE) = \frac{2}{n} \sum_{i=1}^{n} Cov(y_i, \hat{y}(m)_i) = 2\frac{\sigma^2}{n} p(m),$$

since

$$H(m) = X(X'X)^{-1}X' \to \frac{1}{n} \sum_i h_{ii} Trace(H) \to$$

$$Trace(H) = Trace(X(X'X)^{-1}X') = Trace((X'X)(X'X)^{-1}) = Trace(I_{p(m)}) = p(m)$$

4.2 Conclusion

Why did we bother with the above derivation? Well, in most cases we don't actually have an independent test data set so we can't estimate the expected pMSE (the PE). However, we need it for model selection since we have already established that the RSS does not work. Thus, we need to get at an estimate of the PE. Since we have established that the gap between the *expected* MSE and pMSE is $\frac{2\sigma^2}{n}p(m)$ and we have a natural estimate for the expected training MSE in the MSE (RSS/(n-p)). Thus, our selection criterion is

$$p\widehat{MSE}(m) = \widehat{MSE}(m) + \frac{2\sigma^2}{n} p(m) \hat{=} \frac{RSS(m)}{n - p(m)} + \frac{2\sigma^2}{n} p(m).$$

Many model selection criteria have this format:

Goodness-of-fit measure + modelsize-penalty.

The one we just derived is called *Mallow's Cp* and take the form

$$Cp(m) = MSE + \frac{2\sigma^2}{n} p(m),$$

or

$$RSS(m) + 2\sigma^2 p(m).$$

To use this in practise we need to know σ^2. We play in the best estimate for σ^2 we have, namely $\hat{\sigma}^2$ obtained from the largest model fit to the data.

1. Enumerate all models $m = 1, \ldots, M$
2. Evaluate the $MSE(m), RSS(m)$ for all models
3. Compute $Cp(m) = RSS(m) + 2\hat{\sigma}^2 p(m)$ where $\hat{\sigma}^2 = RSS(M)/(n - p(M))$
4. Pick the model that minimizes Cp

There are other commonly used model selection criteria that take on a similar form (and are derived under somewhat similarly): the AIC and BIC where

$$AIC(m) = n \log RSS(m) + 2p(m)$$

and

$$BIC(m) = n \log RSS(m) + p(m) \log(n).$$

Note that both these criteria are similar to Cp in that they compare the goodness of fit (a function of the RSS) and the complexity of the model (the number of parameters $p(m)$). The AIC is likelihood based, which for linear regression models makes Cp and AIC behave rather similarly. The BIC is Bayesian take on model selection. What you should know is that, in general, the AIC is rather "generous", i.e. picks larger models whereas the BIC is quite conservative in comparison (picks smaller models), especially for small sample sizes. It has been shown that the AIC is overly generous asymptotically whereas the BIC is consistent. However, for finite samples it is more complicated.

5 Demo 9

We will compare cross-validation, backward selection and all subset selection using Cp, AIC and BIC using the South African heart disease data as our demo data. We start by reading the data into R. We then use the R package `leaps()` to create an enumeration of all subset models (the matrix `Models` below).

```
> SA <- data.frame(read.table("SA.dat", sep = "\t", header = T))
> yy <- SA[, 12]
> xx <- SA[, -12]
> library(leaps)
> rleaps <- regsubsets(xx, yy, int = T, nbest = 250, nvmax = 250,
+     really.big = T, method = c("ex"))
> cleaps <- summary(rleaps, matrix = T)
> Models <- cleaps$which
> Models <- rbind(c(T, rep(F, dim(xx)[2])), Models)
```

Let's review 10-fold cross-validation. First, we create the 10 folds of data:

```
> K <- 10
> ii <- sample(seq(1, length(yy)), length(yy))
> foldsize <- floor(length(yy)/K)
> sizefold <- rep(foldsize, K)
> restdata <- length(yy) - K * foldsize
> if (restdata > 0) {
+     sizefold[1:restdata] <- sizefold[1:restdata] + 1
+ }
```

We cycle trough each fold of data, fit each of the models and compute the prediction errors:

```
> Prederrors <- matrix(0, dim(Models)[1], K)
> iused <- 0
> Xmat <- as.matrix(cbind(rep(1, dim(xx)[1]), xx))
> for (k in (1:K)) {
+     itest <- ii[(iused + 1):(iused + sizefold[k])]
+     itrain <- ii[-c((iused + 1):(iused + sizefold[k]))]
+     iused <- iused + length(itest)
+     for (mm in (1:dim(Models)[1])) {
+         betahat <- solve(t(Xmat[itrain, Models[mm, ]]) %*% Xmat[itrain,
+             Models[mm, ]]) %*% t(Xmat[itrain, Models[mm, ]]) %*%
+             yy[itrain]
+         ypred <- Xmat[itest, Models[mm, ]] %*% betahat
+         Prederrors[mm, k] <- sum((yy[itest] - ypred)^2)
+     }
+ }
> PE <- apply(Prederrors, 1, sum)/length(yy)
```

There are more than 1400 models in the above comparison: here are the top 5

```
> jj <- sort.list(PE)[1:5]
> print(as.matrix(Models[jj, ]))
```

```
  (Intercept)  age   sbp adiposity obesity typea alcohol alcind tobacco tobind
6        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE    TRUE FALSE    TRUE  FALSE   FALSE   TRUE
6        TRUE TRUE FALSE      TRUE    TRUE FALSE    TRUE  FALSE   FALSE   TRUE
5        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE   FALSE  TRUE    TRUE  FALSE   FALSE   TRUE
   chd famhist
6 TRUE    TRUE
7 TRUE    TRUE
6 TRUE   FALSE
5 TRUE   FALSE
7 TRUE    TRUE
```

I also use the `xtable()` command in R to create a nicer table that works with LaTeX(if you use windows, make sure to put output like this into a proper table with a caption).

```
> z <- data.frame(as.matrix(cbind(Prederrors[jj, ], PE[jj])))
> colnames(z) <- c("Fold1", "Fold2", "Fold3", "Fold4", "Fold5",
+     "Fold6", "Fold7", "Fold8", "Fold9", "Fold10", "PE")
> row.names <- c(seq(1, 5))
> library(xtable)
> xtable(z, digits = c(0, rep(0, 10), 3), caption = "Prederrors in different folds and total",
+     label = "tab:CVldl")
```

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	69	72	86	100	85	95	89	88	106	85	2.804
2	69	71	85	100	84	96	94	89	105	83	2.807
3	69	69	89	101	81	94	98	89	108	81	2.813
4	68	71	90	101	82	93	94	87	110	83	2.815
5	71	71	87	99	85	96	88	87	113	83	2.816

Table 1: Prederrors in different folds and total

In Table 1 you can compare the fold results and total prediction error results for the different models (seen in the R output). The winning model is

```
> winmod <- Models[which.min(PE), ]
> print(winmod)
```

```
(Intercept)         age         sbp   adiposity     obesity       typea 
       TRUE        TRUE       FALSE        TRUE       FALSE       FALSE 
    alcohol      alcind     tobacco      tobind         chd     famhist 
       TRUE       FALSE       FALSE        TRUE        TRUE        TRUE 
```

Let's compare this to the backward model selection results:

```
> mm <- lm(log(ldl) ~ log(age) + sbp + adiposity + log(obesity) +
+     typea + alcohol + alcind + tobacco + tobind + as.factor(chd) +
+     as.factor(famhist), data = SA)
> ss <- step(mm, trace = F)
> print(ss)

Call:
lm(formula = log(ldl) ~ adiposity + log(obesity) + alcohol +
    tobind + as.factor(chd) + as.factor(famhist), data = SA)

Coefficients:
        (Intercept)            adiposity         log(obesity)
          -0.295887             0.019417             0.342913
            alcohol               tobind      as.factor(chd)1
          -0.003449             0.143639             0.154434
as.factor(famhist)2
           0.071745
```

Let's now try using the model selection criteria Cp, AIC and BIC instead. We use the results from `rleaps()` above (which computes Cp for us, as well as everything we need to compute AIC and BIC).

```
> tt <- apply(cleaps$which, 1, sum)
> BIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * log(length(yy))
> AIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * 2

> plot(tt, cleaps$cp, main = "Cp")

> plot(tt, AIC, main = "AIC")

> plot(tt, BIC, main = "BIC")
```

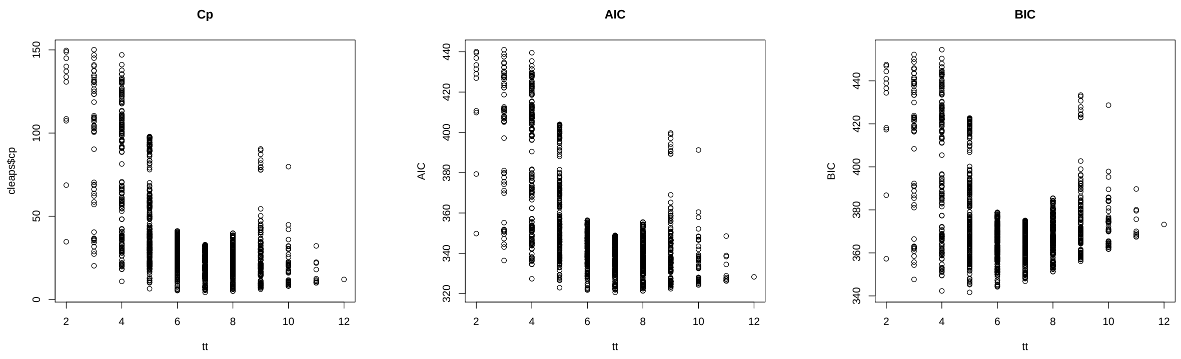

Figure 2: Cp, AIC and BIC selection criteria

In Figure 2 we depict Cp, AIC and BIC as a function of model size for all models we investigate. Notice how the lower envelope, which consists of the best models of each size, increases faster for large models using BIC compared with Cp and AIC. BIC penalizes model size more severely than Cp and AIC.

```
> rleaps <- regsubsets(xx, yy, int = T, nbest = 1, nvmax = dim(xx)[2] +
+     1, really.big = T, method = c("ex"))
> cleaps <- summary(rleaps, matrix = T)
> tt <- apply(cleaps$which, 1, sum)
> BIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * log(length(yy))
> AIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * 2
> par(mfrow = c(1, 1))
> plot(tt, cleaps$cp - min(cleaps$cp), type = "l", main = "Cp-black,AIC-blue,BIC-red")
> lines(tt, AIC - min(AIC), col = "blue")
> lines(tt, BIC - min(BIC), col = "red")
> abline(h = 0, lty = 2)
```

It is easier to see this if we limit ourselves to the winning model of each size (see Figure 3).

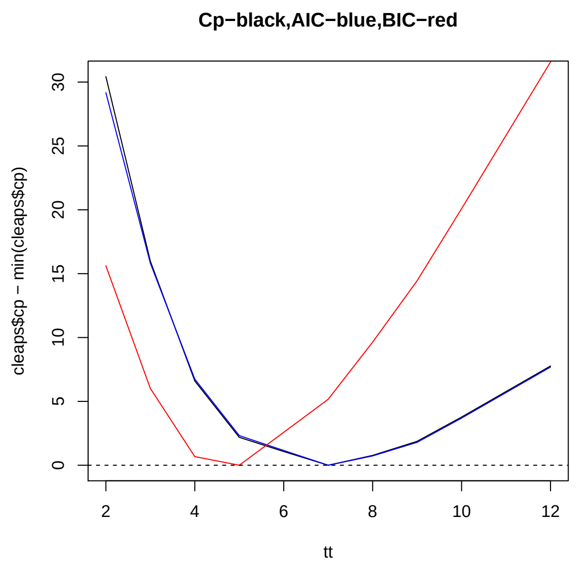

Figure 3: Cp, AIC and BIC selection criteria

We summarize the selected models

```
> cpmod <- cleaps$which[cleaps$cp == min(cleaps$cp), ]
> aicmod <- cleaps$which[AIC == min(AIC), ]
> bicmod <- cleaps$which[BIC == min(BIC), ]
> print(cpmod)

(Intercept)         age         sbp   adiposity     obesity       typea 
       TRUE        TRUE       FALSE        TRUE       FALSE       FALSE 
    alcohol      alcind     tobacco      tobind         chd     famhist 
       TRUE       FALSE       FALSE        TRUE        TRUE        TRUE 

> print(aicmod)

(Intercept)         age         sbp   adiposity     obesity       typea 
       TRUE        TRUE       FALSE        TRUE       FALSE       FALSE 
    alcohol      alcind     tobacco      tobind         chd     famhist 
       TRUE       FALSE       FALSE        TRUE        TRUE        TRUE 

> print(bicmod)

(Intercept)         age         sbp   adiposity     obesity       typea 
       TRUE       FALSE       FALSE        TRUE       FALSE       FALSE 
    alcohol      alcind     tobacco      tobind         chd     famhist 
       TRUE       FALSE       FALSE        TRUE        TRUE       FALSE 
```

5.1 Random subsets of data

Let us repeat the model selection on several random subsets of data to try to discover which variables are most frequently selected. I start by choosing a splitting fraction for the data and the number of random splits to perform.

```
> frac <- 0.5
> K <- 1000
> modsizecp <- rep(0, K)
> modsizeaic <- rep(0, K)
> modsizebic <- rep(0, K)
> PEcpK <- rep(0, K)
> PEaicK <- rep(0, K)
> PEbicK <- rep(0, K)
> modselcp <- rep(0, dim(SA)[2] - 1)
> modselaic <- rep(0, dim(SA)[2] - 1)
> modselbic <- rep(0, dim(SA)[2] - 1)

> for (kk in (1:K)) {
+     ii <- sample(seq(1, dim(SA)[1]), round(dim(SA)[1] * frac))
+     yy <- SA[ii, 12]
+     xx <- as.matrix(SA[ii, -12])
+     yyt <- SA[-ii, 12]
+     xxt <- as.matrix(SA[-ii, -12])
+     rleaps <- regsubsets(xx, yy, int = T, nbest = 1, nvmax = dim(SA)[2],
+         really.big = T, method = c("ex"))
+     cleaps <- summary(rleaps, matrix = T)
+     tt <- apply(cleaps$which, 1, sum)
+     BIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * log(length(yy))
+     AIC <- length(yy) * log(cleaps$rss/length(yy)) + tt * 2
+     cpmod <- cleaps$which[cleaps$cp == min(cleaps$cp), ]
+     aicmod <- cleaps$which[AIC == min(AIC), ]
+     bicmod <- cleaps$which[BIC == min(BIC), ]
+     mmcp <- lm(yy ~ xx[, cpmod[2:length(cpmod)] == T])
+     mmaic <- lm(yy ~ xx[, aicmod[2:length(aicmod)] == T])
+     mmbic <- lm(yy ~ xx[, bicmod[2:length(bicmod)] == T])
+     PEcpK[kk] <- sum((yyt - cbind(rep(1, dim(xxt)[1]), xxt[,
+         cpmod[2:length(cpmod)] == T]) %*% mmcp$coef)^2)/length(yyt)
+     PEaicK[kk] <- sum((yyt - cbind(rep(1, dim(xxt)[1]), xxt[,
+         aicmod[2:length(aicmod)] == T]) %*% mmaic$coef)^2)/length(yyt)
+     PEbicK[kk] <- sum((yyt - cbind(rep(1, dim(xxt)[1]), xxt[,
+         bicmod[2:length(bicmod)] == T]) %*% mmbic$coef)^2)/length(yyt)
+     modsizecp[kk] <- sum(cpmod)
+     modsizeaic[kk] <- sum(aicmod)
+     modsizebic[kk] <- sum(bicmod)
+     modselcp[cpmod[2:length(cpmod)] == T] <- modselcp[cpmod[2:length(cpmod)] ==
+         T] + 1
+     modselaic[aicmod[2:length(cpmod)] == T] <- modselaic[aicmod[2:length(cpmod)] ==
+         T] + 1
+     modselbic[bicmod[2:length(cpmod)] == T] <- modselbic[bicmod[2:length(cpmod)] ==
+         T] + 1
+ }
```

The above is a loop that cycles over splitting the data, model selection and prediction on the left-out part of the data several times. The `PE` vectors store the prediction errors associated with the selected models using each of the criteria (Cp, AIC, BIC). The vectors `modesize` stores the selected models' sizes and `modsel` is a vector which for each variable counts up the number of times it is selected.

We summarize our findings as follows:

```
> modseltabs <- cbind(names(SA)[-12], modselcp, modselaic, modselbic)
> print(modseltabs)

                  modselcp modselaic modselbic
 [1,] "age"       "294"    "315"     "41"
 [2,] "sbp"       "99"     "108"     "8"
 [3,] "adiposity" "995"    "996"     "958"
 [4,] "obesity"   "260"    "272"     "69"
 [5,] "typea"     "205"    "215"     "56"
 [6,] "alcohol"   "888"    "894"     "557"
 [7,] "alcind"    "71"     "81"      "6"
 [8,] "tobacco"   "110"    "114"     "12"
 [9,] "tobind"    "728"    "747"     "266"
[10,] "chd"       "933"    "940"     "738"
[11,] "famhist"   "343"    "364"     "86"
```

For a final report, you should put this is a proper table:

```
> z <- data.frame(as.matrix(cbind(modselcp, modselaic, modselbic)))
> colnames(z) <- c("Cp", "AIC", "BIC")
> row.names(z) <- c(names(SA)[-12])
> library(xtable)
> xtable(z, digits = c(0, rep(0, 3)), caption = "Random splits results",
+     label = "tab:seltable")
```

	Cp	AIC	BIC
age	294	315	41
sbp	99	108	8
adiposity	995	996	958
obesity	260	272	69
typea	205	215	56
alcohol	888	894	557
alcind	71	81	6
tobacco	110	114	12
tobind	728	747	266
chd	933	940	738
famhist	343	364	86

Table 2: Random splits results

In Table 2 we see that some variables stand out as almost always selected (`adiposity, alcohol, chd` using Cp and AIC, `adiposity` and `chd` using BIC). Several other variables are selected fairly frequently as well, e.g. the smoking indicator `tobind`.

We also compare the average model size and average prediction error over the random splits.

```
> print(c("mean PE for cp, aic and bic"))

[1] "mean PE for cp, aic and bic"

> print(c("PEcp=", round(mean(PEcpK), 4), " PEaic=", round(mean(PEaicK),
+     4), " PEbicK=", round(mean(PEbicK), 4)))

[1] "PEcp="    "3.015"    " PEaic="  "3.014"    " PEbicK=" "3.0676"

> print(c("mean model size for cp, aic and bic"))

[1] "mean model size for cp, aic and bic"
```

```
> print(c("sizecp=", round(mean(modsizecp), 4), " sizeaic=", round(mean(modsizeaic),
+     4), " sizebic=", round(mean(modsizebic), 4)))
```

```
[1] "sizecp="   "5.926"     " sizeaic=" "6.046"     " sizebic=" "3.797"
```

```
> pes <- c(round(mean(PEcpK), 4), round(mean(PEaicK), 4), round(mean(PEbicK),
+     4))
> siz <- c(round(mean(modsizecp), 4), round(mean(modsizeaic), 4),
+     round(mean(modsizebic), 4))
> z <- data.frame(as.matrix(rbind(pes, siz)))
> colnames(z) <- c("Cp", "AIC", "BIC")
> row.names(z) <- c("Prediction Error", "Model size")
> xtable(z, digits = c(0, rep(3, 3)), caption = "Random splits results: PE and model size",
+     label = "tab:seltable2")
```

	Cp	AIC	BIC
Prediction Error	3.015	3.014	3.068
Model size	5.926	6.046	3.797

Table 3: Random splits results: PE and model size

In Table 3 we can see that BIC picks smaller models than Cp and AIC, whereas the prediction errors are quite comparable. Results vary quite a lot between random splits which a simple mean summary does not reveal. We therefore also check the results using boxplots.

```
> par(mfrow = c(1, 2))
> boxplot(as.data.frame(cbind(modsizecp, modsizeaic, modsizebic)))
> boxplot(as.data.frame(cbind(PEcpK, PEaicK, PEbicK)))
```

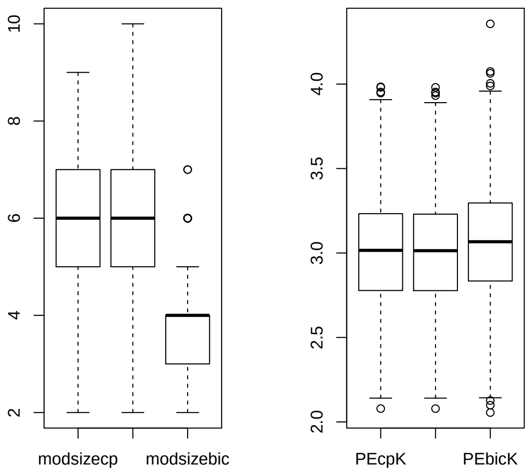

Figure 4: Cp, AIC and BIC selection criteria - model sizes and prediction errors

As you can see in Figure 4, model sizes differ between BIC and the other two criteria, whereas prediction error distributions overlap. However, this is not a fair comparison either since in fact, each random split has its own characteristics. We should compare prediction errors for each random split (cf. t-test and paired t-test).

```
> par(mfrow = c(1, 1))
> boxplot(as.data.frame(cbind(modsizeaic - modsizecp, modsizebic -
+     modsizecp, modsizebic - modsizeaic)), names = c("AIC-CP",
+     "BIC-CP", "BIC-AIC"), main = "Model sizes")
> abline(h = 0, lty = 2)

> boxplot(as.data.frame(cbind(PEaicK - PEcpK, PEbicK - PEcpK, PEbicK -
+     PEaicK)), names = c("AIC-CP", "BIC-CP", "BIC-AIC"), main = "PE")
> abline(h = 0, lty = 2)
```

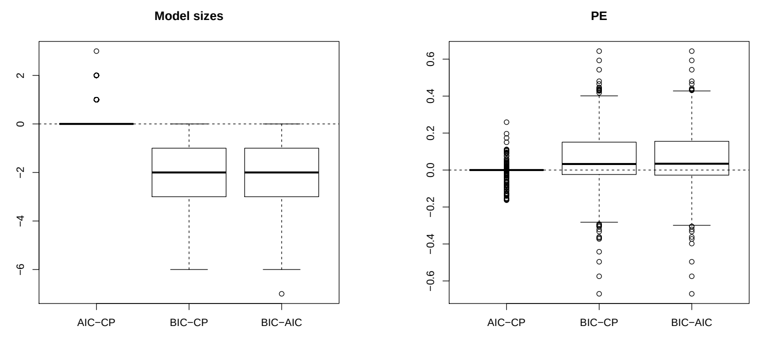

Figure 5: Cp, AIC and BIC selection criteria. Left: difference in model size. Right: difference in prediction error.

In Figure 5 we see from the pairwise comparisons that prediction errors don't differ much on average between the different selection criteria, but you might have a better prediction error using BIC over AIC for one random split and the reverse for another. Note, on other data sets and other random splits this may not be so and in fact one criterion may perform better than another. In general, for small sample sizes BIC tends to be overly conservative, whereas for large sample sizes Cp and AIC tend to produce unnecessarily large models.

```
> plot(PEcpK, PEaicK, xlab = "PE with Cp", ylab = "")
> points(PEcpK, PEbicK, pch = 2, col = 2)
> abline(0, 1)
```

Finally, in Figure 6 we summarize the random splits with a scatter plot.

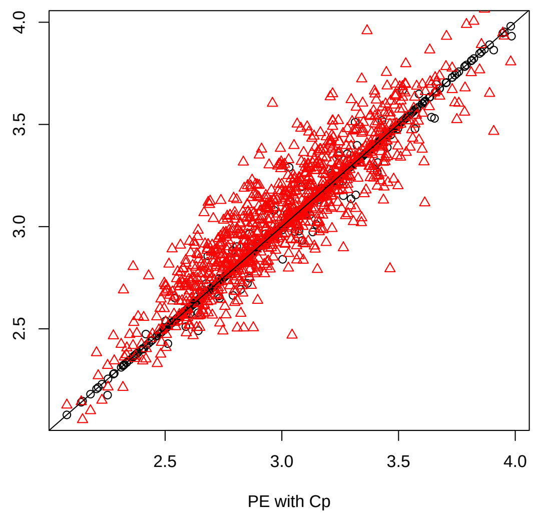

Figure 6: Cp, AIC and BIC selection criteria. Prediction errors: AIC vs Cp (o) and BIC vs Cp (triangle)