Chapter 8

Building the Regression Model

8.1 Which Predictors to Include?

Data sets with many candidate predictor variables are common, and the larger the number of candidate predictor variables, the larger the number of possible regression models. For example, in the case of n candidate predictors, there are 2^n possible regression models to choose from (not including interaction terms, squared terms, etc.). How can we identify the most appropriate subset of these variables to include in the regression model? We will study two different predictor selection methods: *stepwise regression* and the *method of best subsets*. In either case, the goal is to wind up with a regression model that summarizes the response trend, provides good response predictions, and gives good estimates of model parameters.

When you build a regression model, one of four things can occur. The model could

1. be "correctly specified", meaning all the relevant predictor variables are used, no more and no less, including transformations and interaction terms. There are no missing, redundant, or extraneous variables. The regression coefficients for a correctly specified model are unbiased estimators for the actual parameters (the β_i), and the predicted responses are unbiased estimators for the conditional mean responses given the predictor variable values. In addition, MSE is an unbiased estimator for the variance σ^2 of the error terms.

2. be “under-specified”, meaning one or more important predictors have been omitted. These models are said to have “high-bias”. Underspecified models are poor because the regression coefficients and response predictions are biased estimators. MSE tends to overestimate σ^2 in this case.

3. “over-specified”, meaning we’ve included all the “right variables”, but too many of them. In this case some of the predictor variables are redundant the sense that they “indicate the same thing”- perhaps one or more predictor variables can be described well by linear functions of another or others. This is called *multicolinearity.* This is also referred to as having “high-variance”, and these models can over-fit the data by fitting to noise. Regression models that are overspecified give unbiased regression coefficient estimates, unbiased predictions of the response, and unbiased MSE. These models can be used, with caution, for prediction of the response, but should not be used to describe the effect of a predictor on the response. These models are also simply more complicated and difficult to understand than necessary.

4. the model could contain “extraneous variables”. In this case we’ve included some variables that are unrelated to the response or any of the other predictors, and we should identify and remove them. Models with extraneous variables do give unbiased regression coefficients, unbiased predictions of the response, and unbiased MSE. Unfortunately, MSE willl have fewer degrees of freedom due to there being more parameters in the model, and so CIs tend to be wider than they should and hypothesis tests tend to have lower power. As with the overspecified case, we’ve also made our model more complicated and hard to understand than necessary.

Example 41 *Bodyfat.* Open the Bodyfat data in R. Use the variables Height and Weight to predict Bodyfat:

```
> bf <- read.table("Bodyfat.txt", header=TRUE)
> head(bf)
> attach(bf)
> out1 <- lm(Bodyfat ~ Weight + Height)
> summary(out1)


Call:
lm(formula = Bodyfat ~ Weight + Height)

Residuals:
     Min       1Q   Median       3Q      Max
-12.7697  -3.9527  -0.5364   4.0473  13.2829

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 71.48247   16.20086   4.412 2.65e-05 ***
Weight       0.23156    0.02382   9.721 5.36e-16 ***
Height      -1.33568    0.25891  -5.159 1.32e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.754 on 97 degrees of freedom
Multiple R-squared:  0.494,     Adjusted R-squared:  0.4836
F-statistic: 47.35 on 2 and 97 DF,  p-value: 4.48e-15
```

The output indicates the intercept term and both slope coefficients are significant. In addition, $MSE = 5.754^2 = 33.11$. Now include the variable Abdomen as a predictor:

```
> out2 <- lm(Bodyfat ~ Weight + Height + Abdomen)
> summary(out2)


Call:
lm(formula = Bodyfat ~ Weight + Height + Abdomen)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5219 -2.9969  0.0378  2.8933  9.2859

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -56.1329    18.1372  -3.095 0.002580 **
```

```
Weight       -0.1756     0.0472  -3.720 0.000335 ***
Height        0.1018     0.2444   0.417 0.677750
Abdomen       1.0747     0.1158   9.279 5.27e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.199 on 96 degrees of freedom
Multiple R-squared:  0.7332,    Adjusted R-squared:  0.7249
F-statistic: 87.96 on 3 and 96 DF,  p-value: < 2.2e-16
```

This output indicates that by including Abdomen, the Height variable is no longer significant for prediction. Also, note that MSE has decreased to $4.199^2 = 17.63$, indicating more of the variation in the model is explained by the predictor variables. Accordingly, the coefficient of determination R^2 has increased from 0.494 to 0.7332. The first model is probably underspecified, and the second might be overspecified (it seems Height is extraneous). Omission of the Abdomen variable results in overestimating the intercept and the slope on the weight variable. In fact, the signs change! The second model indicates body fat percentage actually tends to decrease as weight increases for fixed height and abdomen circumference. This makes sense because fat is not as dense as other tissue like muscle. Now if you only use Weight and Abdomen as predictor variables, we get

```
> out3 <- lm(Bodyfat ~ Weight + Abdomen)
> summary(out3)


Call:
lm(formula = Bodyfat ~ Weight + Abdomen)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5953 -2.9776 -0.0177  2.8974  9.1920

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -48.77854    4.18098 -11.667  < 2e-16 ***
Weight       -0.16082    0.03101  -5.185 1.18e-06 ***
Abdomen       1.04408    0.08918  11.707  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.181 on 97 degrees of freedom
Multiple R-squared:  0.7328,    Adjusted R-squared:  0.7273
F-statistic:   133 on 2 and 97 DF,  p-value: < 2.2e-16
```

The coefficients are all statistically significant, and the MSE has fallen only slightly. Note R^2 as also fallen slightly. However, notice that R^2_{adj} has slightly increased (a good thing). This new model indicates body fat percentage decreases as weight increases for fixed abdomen size, and this should be the case, right? Height was extraneous in the first model. Perhaps together Weight and Abdomen would be sufficient to predict Bodyfat, and because we commonly prefer simpler models (models with fewer terms) to complicated ones, we might do well to omit Height.

Here are some basic model selection tips before going forward:

1. You can never be sure which variables are "wrong" and which variables are "right". The best you can do is use statistical methods and your knowledge of the situation to build a regression model.

2. Knowing the purpose of your regression model can assist in model building. Are some predictors of particular interest? If so, you might definitely want to include them. Are you mainly interested in predicting a response? If so, multicollinearity should worry you less. Or, are you interested in the effects that specific predictors have on the response? If so, multicollinearity could be a concern. What are you trying to accomplish by building a regression model?

3. Identify all possible candidate predictors. Don't worry about interactions or actual transformation forms that might be needed eventually (like $\log x$, $\log(x+1)$, $\sqrt{x}$, etc.) right away. Just make sure to identify all possible important predictors. If you don't consider them they will not appear in the final model, and their absence can contribute to underfitting (high bias).

4. Use variable selection procedures (coming up) to strike a compromise between underspecified models (high-bias) and ones with extraneous or redundant variables (multi-collinearity), or ones with high-variance.

5. Fine-tune the model to get a "correctly" specified model. If needed, change the functional form of the predictors and/or add interaction terms. Check residual behavior. If the residuals suggest problems with the model, try a different functional form of the predictors or remove some of the interaction terms. Iterate back and forth between formulating different regression models and checking the behavior of the residuals until satisfied. Check for lack of linear fit and maybe consider higher-order terms.

6. *Hierarchical Models.* When higher order terms are included in a model, it is typically important to include all lower order terms as well. Consider the quadratic model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon.$$

If we fit this model and the p-values suggest β_0 and β_2 are significant but β_1 is not, we might consider dropping the x term in favor of the model

$$y = \beta_0 + \beta_2 x^2.$$

However, suppose we implement a scale shift $x \to x+c$ for some constant c. Then the model becomes

$$y = \beta_0 + \beta_2 c^2 + 2\beta_2 cx + \beta_2 x^2 + \varepsilon$$

and the first order term reappears. Although they are important for interpretability and computational time reasons, scale changes do not really change the model. Removal of the first order term here implies we do not reject the hypothesis that the extremum of the model occurs at $x = 0$, and this is usually not meaningful in the context of a modeling problem.

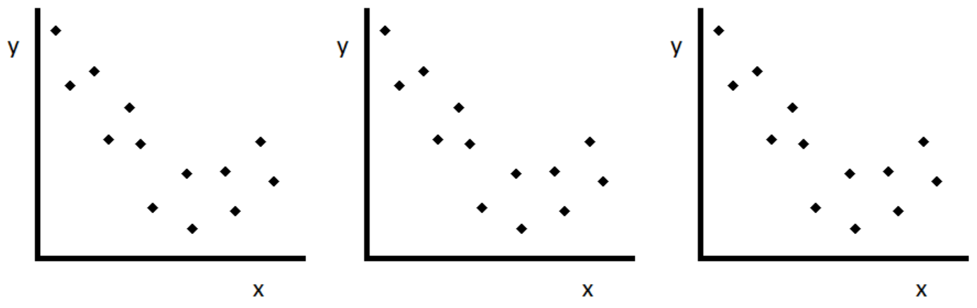
The same guideline applies to interaction terms: we typically do not remove a term with $x_1 x_2$ unless we remove all higher-order interaction terms like x_1^2, x_2^2, $x_1^2 x_2$, $x_1^2 x_2^2$, etc.

8.2 Validation and Cross-Validation Approaches

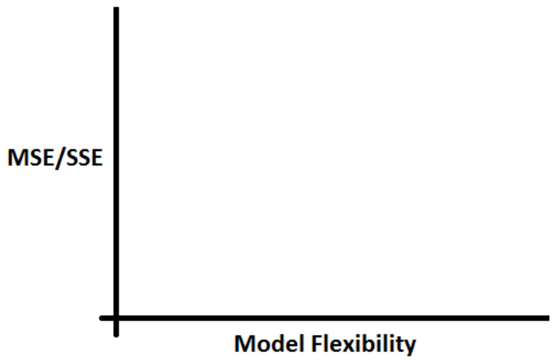
8.2.1 Validation

If enough data are available, we ideally want to randomly split the data set into two or three large sub-categories: a training set and a cross-validation set, or a training set and cross-validation set and test set. These splits are usually about %70/%30 or %60/%20/%20. Then it is a good idea to first build several different models on the training set and compare their predictive ability on the validation set. When we have finally chosen a model we think is good, we can estimate the actual MSE, SSE, or R^2 by evaluating on the test set.

We say the model (the regression line, polynomial, surface, etc.) has *high-bias* if it under-fits the data. We say the model has *high variance* it over-fits the data, meaning it could change quite a bit when building a new one on a different training sets. Different training data sets, even ones that appear similar, would likely result in fairly different models.



Here, *bias* refers to the error in using our simple model to make real predictions. *Variance* refers to the amount which our model $\hat{y}$ would change if built on a different training set.



It turns out that for fixed a fixed value of the predictor variable x_i the expected validation or test MSE for a random training set partitions as

$$E[(y_i - \hat{f}(x_i))^2] = \text{Var}(\hat{f}(x_i)) + [B(\hat{f}(x_i))]^2 + \text{Var}(\varepsilon).$$

Here, $\text{Var}(\varepsilon)$ is constant and so $E[(y_i - \hat{f}(x_i))^2] \geq \text{Var}(\varepsilon)$. So minimizing $E[(y_i - \hat{f}(x_i))^2]$ requires minimizing $\text{Var}(\hat{f}(x_i))$ and $|B(\hat{f}(x_i))|$.

The basic validation set approach:

1. Split the data set into two complementary sets of observations: a training set and a test set with about 50% of the sample in each set. This assumes there are at least 6 to 10 observations per potential predictor variable in the training set. If not, the proportions can be adjusted to about 60%/40% or 70%/30%.

2. Build models of interest on the training set.

3. Compare these models on their SSE, MSE, or R^2 values as computed on the test or validation set. Choose the model with the smallest SSE or MSE, or largest R^2.

4. If a third, unused portion of the data exists (a test set), estimate the true test error by evaluating the SSE, MSE, or R^2 for this model with respect to the test set data.

Example 42 *MPG vs. Horsepower. Adapted from ISLR.*
Begin by partitioning the data into training and validation sets.

```
library(ISLR)
attach(Auto)
set.seed(1)
train1=sample(392, 196)
```

Let's investigate under- and over-fitting with polynomial models of degree 1 through 12. We also plot the validation MSE against the polynomial degree.

```
minmax = range(horsepower)
xvals = seq(minmax[1], minmax[2], len=100)
windows()
par(mfrow=c(2,6))
val.MSE1 = rep(0, 12)
for (i in 1:12){
ddd=i
glm.fit.i = glm(mpg ~ poly(horsepower,ddd), data=Auto, subset = train1)
plot(mpg ~ horsepower, subset=train1, main=paste("Poly Deg ", ddd, sep=" "))
yvals = predict(glm.fit.i, newdata = data.frame(horsepower = xvals))
lines(xvals, yvals, col="blue")
val.MSE1[i] = mean((mpg - predict(glm.fit.i, Auto))[-train1]^2)
}

windows()
val.MSE1
plot(val.MSE1, main="Validation MSE vs. Poly Deg, Seed = 1")
```

Redo with different seeds and compare!

```
set.seed(2)
train2=sample(392, 196)


windows()
par(mfrow=c(2,6))
val.MSE2 = rep(0, 12)
for (i in 1:12){
ddd=i
glm.fit.i = glm(mpg ~ poly(horsepower,i), data=Auto, subset =train2)
plot(mpg ~ horsepower, subset=train2, main=paste("Poly Deg ", ddd, sep=" "))
yvals = predict(glm.fit.i, newdata = data.frame(horsepower = xvals))
lines(xvals, yvals, col="blue")
val.MSE2[i] = mean((mpg - predict(glm.fit.i, Auto))[-train2]^2)
}

windows()
val.MSE2
plot(val.MSE2, main="Validation MSE vs. Poly Deg, Seed = 2")


set.seed(3)
train3=sample(392, 196)

windows()
par(mfrow=c(2,6))
val.MSE3 = rep(0, 12)
```

```
for (i in 1:12){
ddd=i
glm.fit.i = glm(mpg ~ poly(horsepower,i), data=Auto, subset =train3)
plot(mpg ~ horsepower, subset=train3, main=paste("Poly Deg ", ddd, sep=" "))
yvals = predict(glm.fit.i, newdata = data.frame(horsepower = xvals))
lines(xvals, yvals, col="blue")
val.MSE3[i] = mean((mpg - predict(glm.fit.i, Auto))[-train3]^2)
}

windows()
val.MSE3
plot(val.MSE1, main="Validation MSE vs. Poly Deg, Seed = 3")
```

8.2.2 Cross-Validation

There are two shortcomings of the validation approach. One: the test error (MSE or SSE) is a variable that depends on the exact training/validation partitioning of your data. Two: the model is built from the training set data only, and does not use the validation set. Statistical models tend to perform better when trained on larger data sets. If the training set is not large enough, the validation error may tend to overestimate the test error for the model fit to the entire data set (training + validation). Cross-validation approaches are more involved but help alleviate these shortcomings.

Leave-One-Out Cross Validation (LOOCV)

In this method, do not split the data into training, validation, or test sets. Instead,

1. Set the ith data value aside and build the regression model on the remaining $n-1$ observations.

2. Compute $e_{i(i)} = y_i - \hat{y}_{i(i)}$, where $\hat{y}_{i(i)}$ is the predicted value for the ith data value from the model built with the ith data value omitted.

3. Repeat the first two steps for all data values.

4. The LOOCV estimate for the test MSE is

$$MSE_{LOOCV} = \frac{1}{n}\sum_{i=1}^{n} e_{i(i)}^2.$$

Actually, a short-cut formula exists in the case of the general linear model:

$$MSE_{LOOCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1-h_i}\right)^2$$

Note the similarity of MSE_{LOOCV} to MSE_p: smaller values of MSE_{LOOCV} typically coincide with better predictive ability. The MSE_{LOOCV} can also be used to calculate the predicted R^2 (R^2_{pred}), defined as

$$R^2_{pred} \equiv 1 - \frac{\sum_{i=1}^n e_{i(i)^2}}{SSTO} = 1 - n\frac{MSE_{LOOCV}}{SSTO}.$$

Predicted R^2 also assists in assessing predictive ability of a model without selecting another sample or splitting the data into training and validation sets. Used together, MSE_{LOOCV} and R^2_{pred} can help prevent over-fitting because both are calculated using observations not included in the model estimation.

Although R^2 and R^2_{pred} have similar form, it is possible for R^2 to be quite high relative to R^2_{pred}, which implies the model is over-fitting the sample data. However, unlike R^2, the R^2_{pred} can take on negative values, which occurs when the underlying $\sum_{i=1}^n e_{i(i)^2}$ value is inflated beyond the level of $SSTO$. In this case we can set $R^2_{pred} < -\max(R^2_{pred}, 0)$.

k-Fold Cross-Validation

1. Split the original data set into k disjoint but exhaustive sets of observations $T_1, \ldots, T_k$. Common choices for k are 5 or 10.
2. For $i = 1$ to k, build a model using T_i^C as the training set and evaluate MSE_i, SSE_i, or R_i^2 on the test set T_i.
3. Compute the k-fold cross-validation estimate as

$$MSE_{k-foldCV} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

Note LOOCV is n-fold cross-validation. One advantage of k-fold over LOOCV is computational time (when not doing least squares... the short-cut formula is pretty quick). Another advantage is it can more accurately estimate the test error rate than LOOCV because of the bias-variance trade-off.

Example 43 *MPG vs. Horsepower. LOOCV and 10-Fold CV. Adapted from ISLR.*

```
glm.fit = glm(mpg ~ horsepower, data=Auto)
## can be used with cv.glm from boot library
coef(glm.fit)


library(boot)
glm.fit = glm(mpg ~ horsepower, data=Auto)
```

```
set.seed(1)
cv.err  = cv.glm(Auto, glm.fit)
cv.err$delta
cv.error=rep (0,12)
for (i in 1:12){
glm.fit = glm(mpg ~ poly(horsepower, i), data=Auto)
cv.error[i] = cv.glm(Auto, glm.fit)$delta[1]
}
cv.error
```

Now for 10-Fold CV.

```
set.seed(1)
cv.error.10 = rep(0,12)
for (i in 1:12){
glm.fit = glm(mpg ~ poly(horsepower, i), data=Auto)
cv.error.10[i] = cv.glm(Auto, glm.fit, K=10)$delta[1]
}
cv.error.10
```

8.3 C_p, AIC_p, BIC_p, R^2, and R^2_{adj}

Often data sets are not large enough to partition into training and validation sets. In these cases statisticians often opt for building/training models on the entire data set and comparing them with statistics that attempt to estimate the model error. Several popular statistics exist for this purpose which combine information about the SSE, number of parameters in the model, and the sample size. We will consider five. These statistics are often indexed by $p =$ number of model parameters, in order to acknowledge their dependence on p as well as to compare them for models with varying numbers of parameters.

Mallow's C_p:

$$C_p \quad := \quad \frac{SSE_p}{MSE_p} + 2p - n. \tag{8.1}$$

Akaike information criterion (AIC_p):

$$AIC_p \quad := \quad n\log(SSE_p) - n\log(n) + 2p. \tag{8.2}$$

Bayesian information criterion (BIC_p) (sometimes called Schwartz's Bayesian Criterion (SBC_p)):

$$BIC_p \quad := \quad n\log(SSE_p) - n\log(n) + p\log(n). \tag{8.3}$$

Coefficient of multiple determination, R^2_p:

$$R^2_p \quad := \quad 1 - \frac{SSE_p}{SSTO} = \frac{SSR_p}{SSTO}. \tag{8.4}$$

Adjusted R^2:

$$R^2_{adj} \quad := \quad 1 - \left(\frac{n-1}{n-p}\right)\frac{SSE_p}{SSTO} = 1 - \frac{SSE_p/(n-p)}{SSTO/(n-1)}. \tag{8.5}$$

In the formulas, $n =$ sample size or number of observations, $p =$ number of regression coefficients in the model being evaluated (including the intercept), and the quantities indexed by p indicate they are evaluated on a model with that many parameters. The total sum of squares $SSTO$ is evaluated on the full model with all $p-1$ predictor variables. Better models tend to have low C_p, AIC_p, and BIC_p, and high values of $R^2_{adj,p}$.

The C_p is based on total mean squared error. If a fitted model is not the "correct" one, the fits could be biased estimators for the true means. In regression, we use random fits as estimators for mean responses, and so the sum total mean squared error for fits $\hat{Y}_i$ as estimators for mean responses μ_i, adding up over all n random observations, is

Rescaling this quantity by the actual error variance gives the quantity

which can be estimated by the quantity C_p in (8.1). When there is no bias in a model with $p-1$ predictors (meaning $E[\hat{Y}_i] = \mu_i$), then $E[C_p] \approx p$. So models with little bias tend to have C_p close to the number of parameters. Models with significant bias will have $C_p > p$. Inclusion of more predictors results in less bias, but can result in over-fitting to the noise in the data and lead to bad predictions on other data sets (such as a test set or in cross-validation). The full model always yields $C_p = p$, so do not select the full model based on C_p. If all models, except the full model, yield a "large" C_p much greater than p, it could be that some important predictor(s) are missing from the data, and perhaps we should try to identify them (by thinking about the situation, or asking a professional in the field).

Some data analysts feel the information criteria statistics (AIC_p and BIC_p) more realistically compare models than C_p because C_p can make models seem more different than they actually are. The difference between AIC_p and BIC_p is the multiplier of p (the number of parameters, including the intercept), however, the two statistics are derived differently (AIC_p by maximum-likelihood, BIC_p by Bayesian methods). The AIC_p and BIC_p are used similarly when comparing models in that models with lower values are preferred. Use of AIC_p can tend to over-fit models while BIC_p places a higher penalty on the number of parameters in the model so will tend to reward more parsimonious or smaller models.

The R_p^2 is not a great evaluation statistic because it increases monotonically as more predictors enter the model. According to R_p^2, the best model is the one that includes all the predictor variables, and we want to avoid this especially when there are many candidate predictors. The $R_{adj,p}$ acts like R_p^2 except it includes a penalty for including too many variables. However, R^2 is useful when comparing models with the same number of parameters.

Note that MSE_p and SSE_p quantify the distance between predicted and observed responses, and differ only by a factor of p. Of course we want these distances to be small. Note also that R^2 is a function of MSE_p and SSE_p, and R^2 is high if and only if SSE_p and MSE_p are low, and so uses of the three quantities to choose among models are equivalent.

8.4 Best Subsets Method

The idea is to select the subset of predictor variables that does the best job meeting some objective criterion, such as minimizing AIC, BIC, or C_p, or maximizing R^2_{adj}, etc. Keep in mind the main goal is to arrive at a reasonable and useful regression model. We always want to make sure to include all variables that actually predict the response. Otherwise the model may be under-specified and misleading (recall the Bodyfat example).

Here is the best-subsets algorithm:

1. Let k vary across the number of possible model variables. For each $k \in \{0, 1, 2, \ldots, p-1\}$, fit and rank the $\binom{p-1}{k}$ models according to one or more criteria you choose (C_k, AIC_k, BIC_k, R^2_{adj}, etc.). Because k is fixed, R^2 is often used in this step. Choose the best model or models for each criterion and k value. Note for fixed k, different criteria may suggest different models as "best". Compare these models. Further evaluate and refine them by performing residual analyses, data transformations, adding interaction terms, etc. until you believe you have models that satisfy the regression assumptions, summarize the trend in the data, and allow you to answer your research questions.

2. Select the "best" of these models across all k values, criteria, etc., and compare them on a test/cross-validation set according to some criterion (like C_k, AIC_k, BIC_k, or R^2_{adj}). Pick the best of these as your model.

Example 44 *Hitters.* Let's build a model for major league baseball salaries. This example comes from [[4]].

```
> library(ISLR)
> newHitters <- edit(Hitters) # in case you didn't know... you can
> ############################# edit data this way!
> View(Hitters)
> names(Hitters)
> dim(Hitters)
> sum(is.na(Hitters$Salary))
> Hitters = na.omit(Hitters)
> dim(Hitters)
> sum(is.na(Hitters))
> library(leaps)
> regfit.full=regsubsets(Salary~., Hitters, nbest=1,
+ ######### 1 best model for each num of pred
+ nvmax = NULL,     # NULL for no limit on num of vars
+ force.in = NULL, force.out = NULL, method = "exhaustive")
> summary(regfit.full)
```

For each value of k = number of variables, regsubsets() ranks the models according to R^2 (or MSE or SSE or RSS). R does compute values of some of the other statistics (C_p, BIC, etc.) for comparing models across k-values:

```
> reg.summary = summary(regfit.full)
> names(reg.summary)
```

R does not include AIC when doing regsubsets() because ranking according to AIC and BIC has the same result.

```
> reg.summary$rsq
> windows()
> par(mfrow=c(2,2))
> plot(reg.summary$rss, xlab = "Number of Variables", ylab="RSS or SSE",
+ type="l")
> plot(reg.summary$adjr2, xlab = "Number of Variables",
+ ylab="Adjusted RSq", type="l")
> which.max(reg.summary$adjr2)
> points(11, reg.summary$adjr2[11], col="red", cex=2, pch=20)
> plot(reg.summary$cp, xlab="Number of Variables", ylab = "Cp",
+ type = 'l')
> which.min(reg.summary$cp)
> points(10, reg.summary$cp[10], col="red", cex=2, pch=20)
> which.min(reg.summary$bic)
> plot(reg.summary$bic, xlab="Number of Variables",
+ ylab="BIC", type='l')
> points(6, reg.summary$bic[6], col="red", cex=2, pch=20)
```

```
> windows()
> par(mfrow=c(2,2))
> plot(regfit.full, scale="r2")
> plot(regfit.full, scale="adjr2")
> plot(regfit.full, scale="Cp")
> plot(regfit.full, scale="bic")
```

You can see that the model with lowest BIC (at -150) is the one with seven parameters, or six variables. We can extract that model and its coefficients:

```
> coef(regfit.full, 6)
```

```
 (Intercept)        AtBat          Hits
  91.5117981   -1.8685892     7.6043976
       Walks         CRBI     DivisionW
   3.6976468    0.6430169  -122.9515338
     PutOuts
   0.2643076
```

Example 45 *Blood pressure.* Open the dataset called bloodpressure.txt. A researcher has measured blood pressure and other variables on 20 individuals with hypertension:

$y =$	blood pressure in mmHG (BP);
$x_1 =$	age in years (Age);
$x_2 =$	weight in kg (Weight);
$x_3 =$	body surface area in square meters (BSA);
$x_4 =$	duration of hypertension in years (Dur);
$x_5 =$	basal pulse in beats per minute (Pulse);
$x_6 =$	stress index (Stress).

The researchers want to know if a predictive relationship exists between blood pressure and the variables age, weight, body surface area, hypertension duration, pulse rate and/or stress level.

```
> names(bp)
> pairs(~BP+Age+Weight+BSA+Dur+Pulse+Stress, data=bp,
+ main="Scatterplot Matrix")
> windows()
> View(bp)
> dim(bp)
> sum(is.na(bp))
> library(leaps)
> regfit.full = regsubsets(BP~., bp) #best by SSE
> reg.summary = summary(regfit.full)
> reg.summary
> reg.summary$cp
> reg.summary$aic     ### regsubsets doesn't calculate this but
> par(mfrow=c(2,2))   ### it's not needed.  BIC and AIC report same models
> plot(reg.summary$rss, xlab = "Number of Variables", ylab="SSE",
+ type = "l")
> points(which.min(reg.summary$rss), reg.summary$rss[which.min(reg.summary$rss)],
+ col="blue", cex=2, pch = 20)
> plot(reg.summary$adjr2, xlab = "Number of Variables", ylab="adjRS",
+ type = "l")
> points(which.max(reg.summary$adjr2),
+ reg.summary$adjr2[which.max(reg.summary$adjr2)],
+ col="blue", cex=2, pch = 20)
> plot(reg.summary$cp, xlab = "Number of Variables", ylab="Cp",
+ type="l")
> points(which.min(reg.summary$cp),
+ reg.summary$cp[which.min(reg.summary$cp)],
+ col="blue", cex=2, pch = 20)
```

```
> plot(reg.summary$bic, xlab = "Number of Variables",
+ ylab="BIC", type = "l")
> points(which.min(reg.summary$bic),
+ reg.summary$bic[which.min(reg.summary$bic)],
+ col="blue", cex=2, pch = 20)
> windows()
> par(mfrow=c(2,2))
> plot(regfit.full, scale="r2") #
> plot(regfit.full, scale="adjr2")
> plot(regfit.full, scale="Cp")
> plot(regfit.full, scale="bic")
> windows()
```

Based on the R^2 criterion, the best model includes two predictors: Age and Weight.

Based on R^2_{adj} and MSE, the best model includes all six predictors: Age, Weight, BSA, Duration, Pulse, and Stress. Note that patient # (Pt) should probably not be of any consequence. However, you could argue any number of sub-models are also satisfactory based on these criteria — such as the model containing Age, Weight, BSA, and Duration.

Two models look promising based on C_p: the model containing Age, Weight, and BSA; the model containing Age, Weight, BSA, and Duration.

Note some models have $C_p \gg p$. These models could exhibit significant bias. Don't worry too much about C_p that are only slightly larger than p.

Thinking practically (as opposed to statistically), you might argue in favor of using the model consisting of the predictors Age and Weight because most people know their age and weight, and so the data would be easy to obtain. Also, there appears to be nothing substantially wrong with this model:

```
> attach(bp)
> out <- lm(BP ~ Age + Weight)
> summary(out)
> library(rgl)
> library(scatterplot3d)
> plot3d(Age, Weight, BP)
> qqnorm(out$residuals)
> qqline(out$residuals)
```

8.5 Stepwise Regression

In cases with many predictor variables (say, 40 or more), best subsets methods may be computationally infeasible ($2^{40} > 1 \times 10^{12}$). In addition, there is a good chance of selecting over-fitted models with large numbers of predictor variables, resulting in an inflated sense of predictive ability on future data and underestimated variance of model coefficients. Stepwise regression methods are computationally efficient automatic routines that perform shorter searches on only part of the set of variable combinations. However, there are many pitfalls to stepwise methods, and some practitioners argue they should be totally avoided because they can often yield sub-optimal models. Regardless, they provide means-to-an-end, and we shall cover some such routines here. There are several variations on the stepwise theme.

Forward stepwise regression.

1. Begin with the null model $\mathcal{M}_0$ (model with no predictor variables).
2. For $j \in \{0, 1, \ldots, p-2\}$, choose $\mathcal{M}_{j+1}$ to be the best model that includes all the predictors from $\mathcal{M}_j$ and one additional predictor variable (note there are $p-1-j$ of these).
3. Select the best model from $\{\mathcal{M}_0, \ldots, \mathcal{M}_{p-1}\}$ using cross validation prediction error, C_p, AIC_p, BIC_p, or R^2_{adj}.

Backward stepwise regression.

1. Begin with the full model $\mathcal{M}_{p-1}$ (model with all possible predictor variables).
2. For $j \in \{p-1, \ldots, 1\}$, choose $\mathcal{M}_{j-1}$ to be the best model that includes all but one predictors from $\mathcal{M}_j$.
3. Select the best model from $\{\mathcal{M}_0, \ldots, \mathcal{M}_{p-1}\}$ using cross validation prediction error, C_p, AIC_p, BIC_p, or R^2_{adj}.

Hybrid stepwise regression. The best subset, forward, and backward stepwise methods usually result in similar models. Many other methods are used, including forward and backward hybrids where at each step variables are added and/or removed according to some criteria.

Example 46 *Bodyfat.* Apply forward, backward, and hybrid stepwise methods in predicting body fat percentage.

```
> data <- read.table("Bodyfat.txt", header = TRUE)
> regfit.fwd <- regsubsets(Bodyfat~., data=data, method="forward")
> summary(regfit.fwd)
> regfit.bwd <- regsubsets(Bodyfat~., data=data, method="backward")
> summary(regfit.bwd)
> library(MASS)
> fit <- lm(Bodyfat~., data=data)
> step <- stepAIC(fit, direction = "both") #begins with
                ##  full model, and steps up and down
                ##  using AIC to decide which variables enter
                ##  or leave.
```

The forward and backward approaches may or may not always give the same results. When those methods finish, you have to choose from the list of models using C_p or BIC, etc. as with best-subsets, or based on practical experience, intuition, etc. The stepAIC() function adds and removes predictors based on AIC criterion, and terminates with exactly one "best" model. There are many variations on these hybrid procedures.

Example 47 *Hitters.*

```
> library(ISLR)
> head(Hitters)
> library(leaps)
> regfit.fwd <- regsubsets(Salary~., data=Hitters,
+ nvmax = 19, method="forward")
> summary(regfit.fwd)
> regfit.bwd <- regsubsets(Salary~., data=Hitters,
+ nvmax = 19, method="backward")
> summary(regfit.bwd)
> regfit.full <- regsubsets(Salary~., data = Hitters,
+ nvmax=19)
> coef(regfit.fwd ,7)
> coef(regfit.bwd ,7)
> coef(regfit.full, 7)
> fit <- lm(Salary ~ ., data=Hitters)
> library(MASS)
> step <- stepAIC(fit, direction="both")
```

8.6 Best Subsets and Step-Wise Selection with CV

Validation with best subsets

1. Split the data set into two complementary sets of observations: a training set and a test set with about 50% of the sample in each set. This assumes there are at least 6 to 10 observations per potential predictor variable in the training set. If not, the proportions can be adjusted to about 60%/40% or 70%/30%.

2. Run best-subsets on the training set and obtain a "best" model for each number of predictor variables (0, 1, ..., $p-1$).

3. Run each of these best-subset models on the test data (do not include training data). Compute SSE_{test} (or MSE_{test}) for each model applied to the test data.

4. Compare the SSE_{test} or MSE_{test} values above and choose the model with the smallest SSE_{test} or MSE_{test}.

5. Finally, run best-subsets on the entire data set and choose the model with the same number of predictors as the best training model. Note this final model may differ from the training model in which predictor variables are included as well as the values of the coefficients.

Example 48 *Hitters.*

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
[1] 263  20
> set.seed(1)
> train <- sample(c(TRUE, FALSE), nrow(Hitters), rep=TRUE)
> test <- (!train)
> library(leaps)
> regfit.best=regsubsets(Salary~., data=Hitters[train, ], nvmax=19)
> test.mat <- model.matrix(Salary~.,data=Hitters[test,])
> val.errors <- rep(NA, 19)
> for(k in 1:19){
+ coefi<- coef(regfit.best, id=k)
+ pred=test.mat[,names(coefi)] %*% coefi
+ val.errors[k] <- mean((Hitters$Salary[test]-pred)^2)
+ }
> val.errors
> which.min(val.errors)
> coef(regfit.best, 10)
```

Let's write a function to calculate predictions for regsubsets(). We will use this function when we do cross-validation:

```
> predict.regsubsets = function(object, newdata, id, ...){
+ form=as.formula(object$call [[2]])
+ mat=model.matrix(form, newdata)
+ coefi = coef(object, id=id)
+ xvars = names(coefi)
+ mat[,xvars] %*% coefi
```

Next run best subsets selection on the entire data set and select the best model with ten variables. Make sure to run it on the full data set in order to obtain more accurate parameter estimates. Note the best ten-variable model from the full data set might include different predictor variables than the one obtained from the training data, which is the case here:

```
> regfit.best=regsubsets(Salary~., data=Hitters, nvmax=19)
> coef(regfit.best, 10)
```

k-fold validation with best subsets

k-Fold Cross-Validation

1. Split the original data set into k disjoint but exhaustive sets of observations $T_1, \ldots, T_k$. Common choices for k are 5 or 10.
2. For $i = 1$ to k, build a model using T_i^C as the training set and evaluate MSE, SSE, or R^2 on the test set T_i.
3. Construct a $k \times (p-1)$ matrix where element (i, j) is the resulting MSE for the best subsets steps performed on test set T_i using j predictor variables.
4. Obtain column averages of the MSE values. The number of predictor variables v you want to use is the column index that corresponds to the smallest average MSE.
5. Perform best-subsets on the full data set and choose the best model with v predictor variables.

Example 49 *Hitters.*

```
> library(leaps)
> k=10
> set.seed(1)
> folds <- sample(1:k, nrow(Hitters), replace = TRUE)
> cv.errors <- matrix(NA, k, 19, dimnames=list(NULL, paste(1:19)))
> for(j in 1:k){
+ best.fit=regsubsets(Salary~., data=Hitters[folds!=j,], nvmax=19)
+ for(i in 1:19){
+ pred <- predict.regsubsets(best.fit, Hitters[folds==j,], id=i)
+ cv.errors[j,i]=mean( (Hitters$Salary[folds == j] - pred)^2)
+ }
+ }
> cv.errors
> mean.cv.errors = apply(cv.errors, 2, mean)
> par(mfrow=c(1,1))
> plot(mean.cv.errors, type='b')
> reg.best=regsubsets(Salary~., data=Hitters, nvmax=19)
> coef(reg.best, which.min(mean.cv.errors))
```

A few remarks:

1. None of the methods we have discussed are guaranteed to be "optimal".

2. It is always best for subject matter experts to weigh in on what variables to include. It may be necessary to force the model to include certain predictors.

3. The candidate predictor variables should include all the variables that actually predict the response. Otherwise, the model will be under-specified and potentially misleading. Refer to the homework problem on *Simpson's Paradox*.

4. The stepwise procedure is typically used on data sets with many, many variables. The data sets in the following examples are intentionally small in order to highlight concepts and implement the procedure.

5. Stepwise methods are heuristic and never guarantee optimal resulting models. In fact they might be way off! While it is a popular procedure, many practitioners argue stepwise regression procedures should be avoided. For example, suppose there are three candidate predictor variables, x_1, x_2, and x_3, and the "best" model is the one that includes x_2 and x_3 only. Now, it could be the case that x_1 and x_3 are highly correlated, and in running the stepwise procedure x_1 is the first variable to enter, then x_2 enters, and then the process terminates, leading to a sub-optimal model.

6. Some texts and practitioners use a t-test p-value approach for determining which variables should enter or leave at each step. However, many researchers say this is very bad practice because it biases p-values and parameter estimates by capitalizing on noise in the data set by searching large numbers of implausible models. When this is done, there is a high probability of including unimportant predictors or excluding important ones.

7. Backward selection requires the number of cases to be larger than the number of predictor variables so the full model can be fit: $n > p - 1$. Forward stepwise does not.

8. Stepwise methods can fails to predict future obervations well in many cases. Stepwise procedures that do not include penalizaion result in overfitting and patterns that do not replicate. Researchers generally agree it's best for subject matter experts to drive model building unless you're using a good "black box" method that includes penalization for large numbers of predictor variables (like a shrinkage method).

9. Backward selection methods with lower α_R values (like %5) may result in minor damage when it comes to prediction.

8.7 Homework

1. In each case below, indicate whether a more flexible model (like a high-degree polynomial) would tend to perform better or worse on a test set than a more inflexible one (like a low-degree polynomial). Explain your reasoning.

   (a) The number of observations n is large and the number of predictor variables is small.

   (b) There is a very large number of predictor variables, but a small number of observations n.

   (c) The apparent dependence of the response on the predictor variables is highly non-linear.

   (d) The variance of the error terms is very high.

2. Answer the following.

   (a) True or false, and explain. If a regression model has high bias, it is unlikely that collecting more data to train/build the model will increase its performance on a validation or test set (with respect to, say, SSE, MSE, or R^2).

   (b) What do you think will happen to the variance of an over-fitted regression model as the size of the training set increases?

   (c) Suppose you build polynomial regressions in one variable ($\hat{y} = \beta_0 + \sum_{k=1}^{p-1} \beta_k x^k$), and you want to choose the polynomial degree by evaluating the performance of your models (using say, MSE) on a validation or test set. Once you choose the final model, would you expect the test MSE to be higher or lower than the training MSE? Why?

   (d) Suppose when you add flexibility to your model by adding higher order terms or more predictor variables that you begin to see the test or validation error (MSE or SSE evaluated on the test or validation set) begin to increase away from the training error. What kind of a problem are your models experiencing?

3. *Simpson's Paradox.* Open the Bodyfat data in R and refer to the R examples at the beginning of this set of notes. Plot Bodyfat vs Weight. The plot seems to indicate body fat percentage tends to increase as weight increases. Now make a 3D plot of Bodyfat vs. Weight and Abdomen like this:

```
> library(rgl)
> library(scatterplot3d)
> plot3d(Weight, Abdomen, Bodyfat)
```

Describe how Simpson's Paradox is apparent in this context. Recall how the sign of the Weight coefficent changed when we included the Abdomen variable in the regression model.

4. Open the prostate data from the faraway package. Model lpsa as the response and all other variables as predictors. Implement the methods of best subsets as well as forward and backward stepwise selection to determine "best" models, comparing them based on the performance measures C_p, BIC, etc.

5. Open the divusa data from the faraway package. Model divorce as the response and all other variables as predictors. Implement the methods of best subsets as well as forward and backward stepwise selection to determine "best" models, comparing them based on the performance measures C_p, BIC, etc.

6. Load and read the documentation for the Boston data set from the MASS package. We want to build a model to predict the per capita crime rate. Implement the methods of (1) best subsets, (2) forward and (3) backward stepwise selection, (4) LOOCV, and (5) k-fold cross validation to build models (use $k = 10$). Compare and discus the models.