Chapter 5
Model Selection

Often it is not clear which model you should use for the data at hand—maybe because it is not known ahead of time which combination of variables should be used to predict the response, or maybe it is not obvious how the response should be modelled. In this chapter we will take a look at a few strategies for comparing different models and choosing between them.

Key Point
How do you choose between competing models? A natural approach to this problem is to choose the model that has the *best predictive performance* on new, independent data, whether directly (using training data to fit the model and separate test data to evaluate it) or indirectly (using information criteria).

A key issue to consider is the level of *model complexity* the data can support—not too simple and not too complex! If the model is too simple, there will be bias because of important features missing from the model. If the model is too complex, there will be too much variance in predictions, because the extra parameters will allow the model to chase the data too much. (Occasionally, it is better to leave out a term even when it is thought to affect the response if there are insufficient data to do a good job of estimating its effect.)

Exercise 5.1: Plant Height and Climate
Which climate variables best explain plant height?

Angela collects data on how tall plants are in lots of different places around the globe. She also has data on 19 different climate variables (precipitation and temperature summarised in many different ways). She is interested in how plant height relates to climate and which climate variables height relates to most closely.

D. I. Warton, *Eco-Stats: Data Analysis in Ecology*, Methods in Statistical Ecology,
https://doi.org/10.1007/978-3-030-88443-7_5

What does the question tell us—descriptive, hypothesis test, interval estimation, . . . ?

What do the data tell us—one variable/more, what type of variable is the response?

So what sort of analysis method are you thinking of using?

Consider Exercise 5.1. The key difference in this example, compared to those of previous chapters, is that we are primarily interested in choosing the best x variables (which climate variables height relates to most closely). This is a *variable selection* or *model selection* problem—the goal is to select the best (or a set of best) models for predicting plant height.

The paradigm we will use for model selection is to maximise predictive capability—if presented with new data, which model would do the best job of predicting the values of the new responses?

5.1 Understanding Model Selection

Model selection is a new way of thinking about things compared to what we have seen before and introduces some new issues we have to consider.

5.1.1 The Bias–Variance Trade-Off

In model selection, as well as trying to choose the right predictors, you are trying to choose the right number of them, i.e. the right level of model complexity. When making a decision about model complexity, you are making a decision about how to trade off bias against variance (Geman et al., 1992). If you make a model too simple, leaving out important terms, predictions will be systematically wrong (they will be *biased*). If you make a model too complex, adding terms that don't need to be there, it will "overfit" the data, chasing it too much and moving away from the main trend. This will increase the *variance* of predictions, essentially absorbing some of the error variance into predictions. The outcome of this is usually a J curve in predictive error, with a steep decrease in predictive error as bias is removed, a gradual increase in variance with overfitting, and a minimum somewhere in between that optimally manages this *bias–variance trade-off* (Maths Box 5.1).

The idea of the bias–variance trade-off applies any time you are choosing between models of differing complexity—most commonly, when deciding which predictors and *how many* predictors to add to a model, but in many other contexts, too.

One example useful for illustrating the idea is when predicting a response as a non-linear function of a single predictor. Many responses in ecology are thought

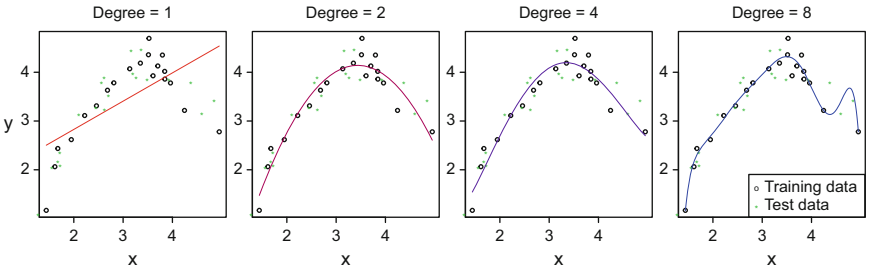

Fig. 5.1: Overfitting a model increases the variance in predictions. Data were generated using a quadratic model (degree = 2), and training data (black circles) were fitted with a model that is linear (degree = 1), quadratic (degree = 2), quartic (degree = 4) or a polynomial with degree 8. Test data (green stars) were used to assess model fit but were not used in model fitting. The straight line is biased; all other models can capture the true trend. But as the degree increased beyond 2, the extra model parameters enabled better tracking of the training data, at the cost of pulling the fitted model away from the true trend and, hence, away from test data. This is especially apparent for x-values between 3 and 4, where several large y-values in the training data "dragged" the fit above the test data for the quartic and 8-degree polynomial models

to follow such non-linear functions (e.g. growth rate as a function of temperature, as in Cooper et al., 2001). In Fig. 5.1, the true model is a quadratic trend, and we consider modelling it using a linear model (a polynomial with degree 1), quadratic (a polynomial with degree 2), quartic (degree 4), or a polynomial with degree 8. As the degree of the polynomial increases, so does the number of parameters needed to fit the model. Note that the linear model misses the curve in the trend (Fig. 5.1) and so is biased. The quartic and 8-degree polynomials are too complex for the data and seem to chase them. This reduces the error variance of the data to which the model was fitted (training data, black curve in Fig. 5.2), but the real test is how well the model would perform on new test data (green curve). Note the J shape of the green curve in Fig. 5.2 (well, a mirror image of a J).

Maths Box 5.1: Bias–Variance Trade-Off

Consider a linear model fitted to n observations, giving predictions $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \ldots, \hat{\mu}_n)$, where $\hat{\mu}_i = \hat{\beta}_0 + \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ for $i = 1, \ldots, n$. The predictive performance of the model can be measured using mean squared error (MSE) estimating the true means μ_i:

$$\text{MSE}(\hat{\boldsymbol{\mu}}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{\mu}_i - \mu_i)^2$$

In practice, we don't know the μ_i, so we don't know MSE($\widehat{\boldsymbol{\mu}}$). But theory tells us a bit about how it will behave. First, we can write it in terms of bias and variance:

$$\text{MSE}(\widehat{\boldsymbol{\mu}}) = \frac{1}{n}\sum_{i=1}^{n} \text{bias}(\hat{\mu}_i)^2 + \frac{1}{n}\sum_{i=1}^{n} \text{var}(\hat{\mu}_i)$$

Bias—one way to get bias is to include too few predictors in the model, missing some important ones. For example, in Maths Box 4.1, we studied the situation where there was one predictor x_i in the model and one missing predictor z_i, where $\mu_i = \beta_0 + x_i\beta_x + z_i\beta_z$, and we fit $\mu_i = \beta_0 + x_i\beta^*$. In this situation the bias is

$$\text{bias}(\hat{\mu}_i) = \beta_0 + x_i\beta^* - (\beta_0 + x_i\beta_x + z_i\beta_z) = (x_i\gamma - z_i)\beta_z = -\delta_i\beta_z$$

and $\frac{1}{n}\sum_i \text{bias}(\hat{\mu}_i)^2 \simeq \sigma_z^2\beta_z^2$, where σ_z^2 is the error variance when predicting $\mathbf{z}$ from $\mathbf{x}$. Bias gets larger if the missing predictor is more important (larger β_z) and more different from other predictors in the model (larger σ_z^2).

Variance—variance increases when there are more predictors in the model, because estimating extra terms introduces extra uncertainty into predictions. In Maths Box 3.2, for a linear model with p terms in it, the sum of variances of $\hat{\mu}_i$ is $p\sigma^2$, when predicting on the same set of predictors the model was fitted to:

$$\frac{1}{n}\sum_{i=1}^{n} \text{var}(\hat{\mu}_i) = \frac{1}{n}p\sigma^2 \tag{5.1}$$

If we use too many predictors, p will be too large, so var($\hat{\mu}_i$) will be too large.

Our goal in analysis is to build a model that is big enough to capture the main trends (not too much bias), but not excessively large (not too much variance). For models with an increasing number of terms, MSE typically follows a J curve—there is a steep initial decrease in MSE as bias is reduced, because there are fewer missing predictors, then a gradual increase as too many predictors are added. The increase is gradual because the $1/n$ in (5.1) keeps the variance small relative to bias, except in the analysis of a small, noisy (large σ) dataset, in which case a small model is often best.

The aim is to find the optimal point in the bias–variance trade-off. This point will be different for different datasets because it depends not just on how much data you have and the relative complexity of the models being compared but also on how well each model captures the true underlying process. In Fig. 5.2, the optimum was at degree = 2, which was the correct answer for this simulation, since a quadratic model was the true underlying process from which these data were simulated. (Sometimes, the optimum can be much smaller than the true model, if the true model is too complex for our data to fit it well.)

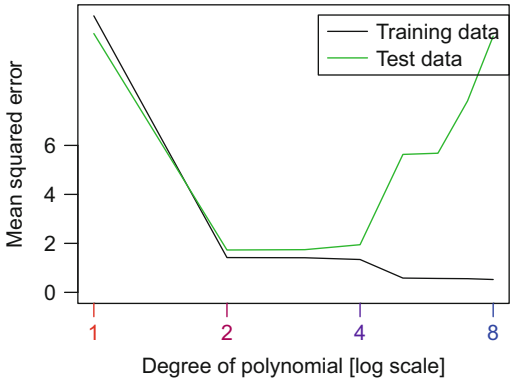

Fig. 5.2: The bias–variance trade-off for polynomial models in Fig. 5.1. Note that for the test data (green curve), the biased model (degree = 1) has a high predictive error, and as the model gets overfitted (degree > 2), the predictive error increases due to an increased variance of predictions. The predictive error for the training data does not account for overfitting, so it always decreases as more parameters are added to the model (black curve). Predictive error was measured here using MSE, defined later

5.1.2 The Problem with R^2 and P-Values for Model Selection

Many use R^2 and P-values to decide how well a model fits, but these aren't good tools to use for model selection.

R^2 makes *no attempt* to account for the costs of model complexity—it keeps going up as you add more terms, even useless ones! If you used R^2 as a basis for including potential predictors in a model, you would end up putting all of them in because that would maximise R^2, irrespective of whether or not each predictor was useful. The same is true of estimated error variance for the data the model was fitted to ($\hat{\sigma}^2$), except this (usually) decreases as you add more terms to the model, as in Fig. 5.2 (black line).

OK, well why not use hypothesis tests? Why not add terms to the model if they are significant and remove terms if they are not significant? This is commonly done and for many years was the main strategy for model selection. Much software still uses this approach as the default, which encourages its use. But there are a few problems with the technique. From a philosophical perspective, it is not what hypothesis testing was designed for, there is not really an *a priori* hypothesis being tested. So it is not really the right way to think about the problem. From a pragmatic perspective, using hypothesis testing for model selection has some undesirable properties. In particular, it is not variable selection consistent, i.e. is not guaranteed to pick the right model even when given as much data as it wants in order to do so—it *overfits*, choosing too complex a model, especially when considering a large number of potential predictors.

5.1.3 Model Selection as Inference

Recall that statistical inference is the process of making some general claim about a population based on a sample. So far we have talked about two types of statistical inference:

1. Hypothesis testing—to see if data are consistent with a particular hypothesis
2. Confidence interval estimation—constructing a plausible range of values for some parameter of key interest.

Now we have a third type of statistical inference:

3. Model selection—which model (or which set of predictor variables) best captures the true underlying process from which the data were generated.

This can be understood as statistical inference because again we are using a sample to make general claims—this time about models (or combinations of predictors), and how well they predict, instead of about parameters.

Note that model selection should *never* be used in combination with hypothesis testing or confidence interval estimation to look at related questions on the same dataset – these methods of inference are not compatible. The process of model selection will only include a term in the model if it is considered important – hence it is doing something kind of like significance testing already. If you were to perform model selection to choose key predictors, then do a hypothesis test on one of these predictors, this is known to lead to high rates of false significance, and similarly, performing model selection then constructing a confidence interval is known to lead to intervals that are biased away from zero. It is best to think of model selection and hypothesis testing/CIs as mutually exclusive: you either use one of these approaches or the other. Although having said this, there is a growing literature on post-selection inference (Kuchibhotla et al., 2022) which offers approaches to address this, the simplest of which is *data splitting* – splitting your data into two independent sets, and applying model selection to one part, and inference to the other.

Key Point
Model selection can be thought of as a method of inference, alongside hypothesis testing and confidence interval estimation. However, it should *not* be applied to the same data you plan to use for a hypothesis test or confidence interval to answer a related question, because these methods won't work correctly in this situation, unless using methods specifically designed for post-selection inference.

5.1.4 It Gets Ugly Quickly

Consider a situation where you have a set of predictor variables and you want to fit all possible models ("all subsets"). If there are p predictor variables, there are 2^p possible models—this gets unmanageable very quickly, as in Table 5.1. If you have 200 observations and 10 variables, the use of all subsets means trying to choose from 1000+ models using just 200 observations. Good luck!

Table 5.1: Variable selection gets ugly quickly—the number of candidate models increases exponentially with the number of predictor variables, such that it is no longer feasible to explore all possible models with just 20 or 30 predictors

# variables	# models to fit
2	4
3	8
5	32
10	1024
20	1,048,576
100	1.27×10^{30}
300	More than the number of electrons in the known universe!

This means two things:

- When there are a lot of possible or *candidate models* being compared, what the data say is the "best model" should be taken with a grain of salt. When there are lots of possible models, it is very hard for the data to make the right call.
- Simplify! The fewer candidate models you are comparing, the better—don't bother with anything you think is unrealistic, and if you know something is important, then include it in all candidate models. Try to refine your question. Do you really need all those variables?

5.1.5 A Cautionary Tale—Building a Spam Filter

In a third-year class several years ago, I asked students, as a group assignment, to construct an e-mail spam filter. The idea was that they would study regular e-mails and spam e-mails from their inbox, find predictors to distinguish between them, and build a model that could predict which e-mails in their inbox were spam.

Some groups put a huge amount of effort into this assignment, using complex models—usually via logistic regression or some variation thereof (as in Chap. 10), with a dozen or so carefully constructed predictors that did a near perfect job of distinguishing spam from real e-mails in the data they built the model on (the

training data). One group even wrote a program for their spam filter and put it on a website, so if you copy-pasted text into it, it would return the predicted probability that your e-mail was spam.

At the other extreme, one group put hardly any effort into the assignment—it seemed like they had forgotten about the assignment and slapped something together at the last minute, with a handwritten report and a simple linear model with just a few terms in (for which model assumptions looked like they wouldn't be satisfied, if they had thought to check them).

As part of the assessment, I brought five e-mails to class and asked students to classify them using their filter (a "test" dataset). Most groups did poorly, getting one or two out of five correct, but one group managed four out of five correct—the last-minute group with the simple linear model.

The problem was that students weren't thinking about the costs of model complexity and were assuming that if they could do a really good job of modelling their training data, their method would work well on new data, too. So they built overly complex models that overfitted their data, chasing them too much and ending up with highly variable predictions, a long way past the optimum on the bias–variance trade-off. The group with the last-minute, simple linear model had the best predictive performance because their model did not overfit the data, so they ended up a lot closer to the optimum choice for model complexity.

The way to beat this problem is to use model selection tools, as described in the following sections, to make sure the level of model complexity is appropriate for the data at hand—not too complex and not too simple. Directly or indirectly, all such methods work by thinking about how well a model can predict using new data.

5.2 Validation

The simplest way to compare predictive models is to see how well they predict using new data, *validation*. In the absence of new data, you can take a *test* or *hold-out* sample from the original data that is kept aside for model evaluation. The remaining *training* data are used to fit each candidate model. Overfitted models may look good on the training data, but they will tend to perform worse on test data, as in Figs. 5.1 and 5.2 or as in the spam filter story of the previous section.

It is critical that the test sample be *independent* of the training sample; otherwise this won't work (see Maths Box 5.2). If all observations are independent (given x), then a random allocation of observations to test/training will be fine. If you have spatial data that are not independent of each other (Chap. 7), a common approach is to break it into coarse spatial blocks and assign these to training and test datasets (Roberts et al., 2017, for example).

Maths Box 5.2: Validation Data Can Be Used to Estimate Mean Squared Error

Predictive performance can be measured using MSE:

$$\text{MSE}(\widehat{\boldsymbol{\mu}}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{\mu}_i - \mu_i)^2$$

But how can we calculate this when we don't know the true mean, μ_i? We can use new observations, since $y_i = \mu_i + \epsilon_i$. We can compare the new responses to their predicted values by estimating the variance of $y_i - \hat{\mu}_i$. Using the adding rule for standard deviations (from Maths Box 1.5) yields

$$\begin{aligned}\sigma^2_{y_i-\hat{\mu}_i} &= \sigma^2_{\mu_i+\epsilon_i-\hat{\mu}_i} \\ &= \sigma^2_{\hat{\mu}_i-\mu_i} + \sigma^2_{\epsilon_i} + 2\sigma_{\epsilon_i,\hat{\mu}_i-\mu_i}\end{aligned}$$

The first term $\sigma^2_{\hat{\mu}_i-\mu_i}$ is another way of writing MSE. The second term is a constant. The third term, the covariance of ϵ_i and $\hat{\mu}_i$, is zero if ϵ_i is independent of $\hat{\mu}_i$. This independence condition is satisfied if y_i is a new observation that is independent of those used in fitting the model.

So when using a set of new "test" observations that are independent of those used to fit the model, estimating $\sigma^2_{y_i-\hat{\mu}_i}$ will estimate MSE($\widehat{\boldsymbol{\mu}}$), up to a constant.

How should we choose the size of the test sample? Dunno! (There is no single best answer.)

One well-known argument (Shao, 1993) is that as sample size n increases, the size of the training sample should increase, but as a proportion of n it should decrease towards zero. This ensures "variable selection consistency", guaranteeing the correct model is chosen for very large n.

An example strategy Shao (1993) suggested (which hence became a bit of a thing) is to use $n^{3/4}$ observations in the training sample. This can be quite harsh, though, as in Table 5.2. This rule tends not to be used so much in ecology, but the general strategy is certainly worth keeping in mind—using a smaller proportion of data in the training set when analysing a larger dataset, rather than sticking with the same proportion irrespective of sample size.

Table 5.2: How the suggested number of training observations changes with sample size if using the $n^{3/4}$ rule mentioned in Shao (1993)

n	20	50	100	200	1000
$n^{3/4}$	9	19	32	53	178

How should we measure predictive performance? For linear regression, the obvious answer is MSE:

$$\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{\mu}_i)^2$$

where the summation is over test observations, for each of which we compare the observed y-value, y_i, to the value predicted by the model fitted to the training sample, $\hat{\mu}_i$. This criterion was used in Fig. 5.2. Maths Box 5.2 explains how this quantity estimates the MSE of predictions. It makes sense to use this criterion for models where we assume equal variance—if not assuming equal variance, then it would make sense to use a criterion that weighted observations differently according to their variance. In later chapters, we will learn about models fitted by maximum likelihood, and in such a situation, it would make sense to maximise the likelihood on test data rather than minimising MSE.

An example using validation via MSE for model selection is in Code Box 5.1.

Code Box 5.1: Using Validation for Model Selection Using Angela's Plant Height Data

Comparing MSE for test data, for models with `rain` and considering inclusion of `rain.seas` (seasonal variation in rainfall)

```
> library(ecostats)
> data(globalPlants)
> n = dim(globalPlants)[1]
> indTrain = sample(n,n^0.75) #select a training sample of size n^0.75:
> datTrain = globalPlants[indTrain,]
> datTest = globalPlants[-indTrain,]
> ft_r = lm(log(height)~rain,dat=datTrain)
> ft_rs = lm(log(height)~rain+rain.seas,dat=datTrain)
> pr_r = predict(ft_r,newdata=datTest)
> pr_rs = predict(ft_rs,newdata=datTest)
> rss_r = mean( (log(datTest$height)-pr_r)^2 )
> rss_rs = mean( (log(datTest$height)-pr_rs)^2 )
> print( c(rss_r,rss_rs) )
[1] 2.145927 2.154608
```

So it seems from this training/test split that the smaller model with just `rain` is slightly better.

Try this yourself—do you get the same answer? What if you repeat this again multiple times? Here are my next three sets of results:

```
> print( c(rss_r,rss_rs) )
[1] 2.244812 2.116212
> print( c(rss_r,rss_rs) )
[1] 2.102593 2.143109
> print( c(rss_r,rss_rs) )
[1] 2.575069 2.471916
```

The third run supported the initial results, but the second and fourth runs (and most) gave a different answer—suggesting that including `rain.seas` as well gave the smaller MSE. But when the answer switches, it suggests that it is a close run thing and the models are actually quite similar in performance.

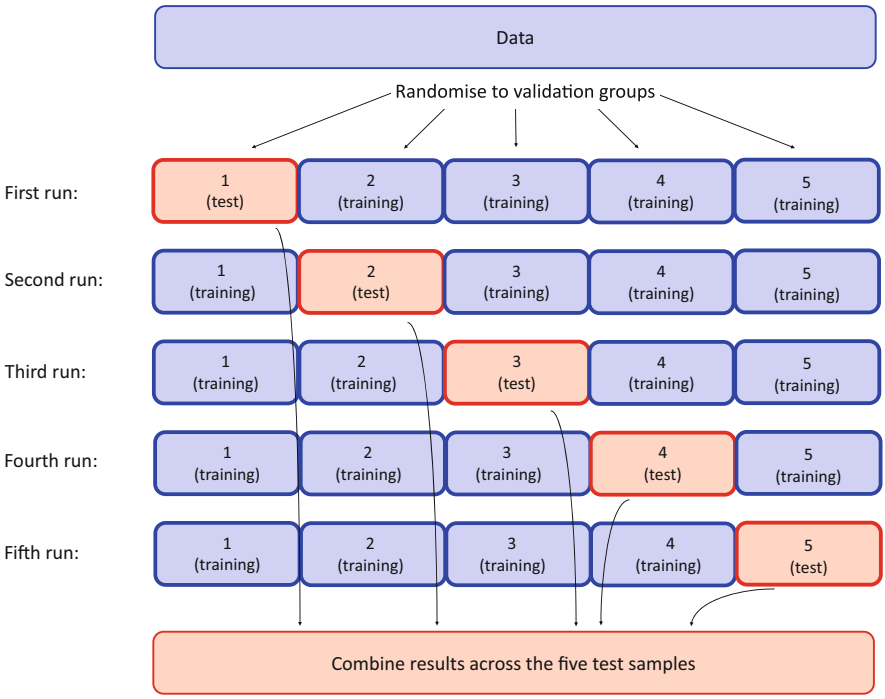

Fig. 5.3: A schematic diagram of five-fold CV. Each observation in the original dataset is allocated to one of five validation groups, and the model is fitted five times, leaving each group out (as the test dataset) once. Estimates of predictive performance are computed for each run by comparing predictions to test observations, then pooled across runs, for a measure of predictive performance that uses each observation exactly once

When using a test dataset to estimate predictive performance on new data, clearly the *test/training split matters*—it is a random split that introduces randomness to results. In Code Box 5.1, four different sets of results were obtained, leading to different conclusions about which model was better. This issue could be handled by repeating the process many (e.g. 50) times and averaging results (and reporting standard errors, too). The process of repeating for different test/training splits is known as *cross-validation* (CV).

5.3 K-fold Cross-Validation

A special case of CV is when you split data into K groups (usually $K = 5$, 5-fold CV, or $K = 10$) and fit K models—using each group as the test data once, as in Fig. 5.3.

Results tend to be less noisy than just using one training/test split, because each observation is used as a test observation once, so one source of randomness (choice of test observation) has been removed.

Code Box 5.2: 5-Fold Cross-Validation for Data of Exercise 5.1

```
> library(DAAG)
> ft_r = lm(log(height)~rain,dat=datTrain)
> ft_rs = lm(log(height)~rain+rain.seas,dat=datTrain)
> cv_r = cv.lm(data=globalPlants, ft_r, m=5, printit=FALSE) # 5 fold CV
> cv_rs = cv.lm(data=globalPlants, ft_rs, m=5, printit=FALSE) # 5 fold CV
> print( c( attr(cv_r,"ms"),attr(cv_rs,"ms") ), digits=6 )
[1] 2.22541 2.15883
```

suggesting that the models are very similar, the model without `rain.seas` predicting slightly better, but by an amount that is likely to be small compared to sample variation. For example, repeating analyses with different random splits (controlled through the `seed` argument):

```
> cv_r = cv.lm(data=globalPlants, ft_r, m=5, seed=1, printit=FALSE)
> cv_rs = cv.lm(data=globalPlants, ft_rs, m=5, seed=1, printit=FALSE)
> print( c( attr(cv_r,"ms"),attr(cv_rs,"ms") ), digits=6 )
[1] 2.21103 2.16553
> cv_r = cv.lm(data=globalPlants, ft_r, m=5, seed=2, printit=FALSE)
> cv_rs = cv.lm(data=globalPlants, ft_rs, m=5, seed=2, printit=FALSE)
> print( c( attr(cv_r,"ms"),attr(cv_rs,"ms") ), digits=6 )
[1] 2.22425 2.14762
> cv_r = cv.lm(data=globalPlants, ft_r, m=5, seed=3, printit=FALSE)
> cv_rs = cv.lm(data=globalPlants, ft_rs, m=5, seed=3, printit=FALSE)
> print( c( attr(cv_r,"ms"),attr(cv_rs,"ms") ), digits=6 )
[1] 2.2783 2.2373
```

we are now getting consistent results on different runs, unlike in Code Box 5.1, suggesting that adding rain. seas to the model improves predictive performance. Also note the answers are looking much more consistent now than before, with predictive errors within 2–3% of each other across runs.

How do you choose K? Dunno! (There is no single correct answer to this.)

The most common choices are N-fold or "leave-one-out" CV, 10-fold, or 5-fold CV. For no particular reason.

You could use the $n^{3/4}$ rule again; no one ever does, though, in K-fold CV. You could try some compromise between this and current K-fold conventions; I use something like the following:

- N-fold or "leave-one-out" cross-validation for small datasets ($n < 20$, say)
- 10-fold CV for medium-sized datasets ($20 < n < 100$, say)
- 5-fold CV for large datasets ($n > 100$, say).

For larger datasets you could go further, in the spirit of Shao (1993), and use two-fold or start to use increasing values of K but just use one of the folds for training and the rest for testing (rather than the other way around, as is done for small samples). This tends not to be done in practice, but there are good theoretical arguments for such an approach.

5.4 Information Criteria

Another way to do model selection is to use the whole dataset (no training/test split) and to penalise more complex models in some way to try to account for the additional variance they introduce. Such approaches are referred to as information criteria, largely for historical reasons (specifically, the first such criterion was derived to minimise an expected Kullback-Leibler information). The two most common criteria are AIC and BIC, which for linear models can be written

$$AIC = n \log \hat{\sigma}^2 + 2p$$
$$BIC = n \log \hat{\sigma}^2 + p \log(n)$$

(sometimes plus a constant), where p is the number of parameters in the model, n is sample size, and $\hat{\sigma}^2$ is the estimated error variance from the linear model.

The aim of the game is to

choose the model that minimises the information criterion

If we were to try to minimise $\hat{\sigma}^2$ alone, this would tend to favour complex models, as if we tried to maximise R^2. By adding $2p$ or $p \log(n)$ to the criterion, there is a larger penalty on models with more terms in them (larger p), so larger models are only chosen if they appreciably improve the fit of the model (by reducing $\hat{\sigma}^2$ appreciably). This penalty is intended to approximate the effects on the variance of predictions when adding unneeded terms to a model.

AIC stands for Akaike information criterion, named after Akaike (1972), although he originally intended the acronym to stand for "an information criterion" (Akaike, 1974). The $2p$ is motivated by thinking about predictive performance on test data and is an approximation to the amount of so-called optimism bias (Efron, 2004) that comes from estimating predictive performance on the same dataset used to fit the model in the first place.

BIC stands for Bayesian information criterion, and while the criterion looks similar, it has quite a different motivation. It was derived as an approximation to the posterior probability of a model being correct, integrating over all its parameters (Schwarz, 1978). Despite this Bayesian motivation, it tends not to be used in Bayesian statistics (Clark, 2007, Chapter 4), but is common elsewhere.

Both criteria can be understood as taking a measure of predictive error and adding a penalty for model complexity (as measured by the number of parameters in the model). For linear models, the measure of predictive error is a function of the error variance, but it will take other forms for other models. The better a model fits the sample data, the smaller the predictive error. While bigger models will always fit the sample data better than smaller models, the penalty for model complexity aims to correct for this.

Both criteria, despite quite different derivations, end up being different only in the value that is multiplied by the number of parameters in the model. AIC uses a 2, whereas BIC uses $\log n$, which will (as long as $n > 7$) take a larger value and, hence, penalise larger models more harshly. Hence when AIC and BIC differ, it is BIC that is choosing the smaller model. AIC is known to overfit, in the sense that

even if the sample size is very large, it will often favour a model that is larger than the best-fitting one. BIC, in contrast, is known to be model selection consistent in a relatively broad range of conditions, i.e. as sample size increases, it will tend towards selecting the best model all the time.

The similarities in the form of the criteria motivate a more general approach, aptly named the generalised information criterion (GIC) (Nishii, 1984), which takes the form

$$GIC = n \log \hat{\sigma}^2 + \lambda p$$

where λ is an unknown value to be estimated by some method (preferably from the data). Usually, we would want to estimate λ in such a way that if sample size (n) were to increase, λ would get larger and larger (going to infinity) but at a slower rate than n. One way to ensure this would be to use CV, but with an increasing proportion of the data in the test sample in larger datasets, as in Table 5.2. This is something of a hybrid approach between those of the two previous sections, using both information criteria and CV. The main advantage of this idea is that predictive error is better estimated, because it is estimated by fitting the model to the whole dataset all at once (using information criteria), while at the same time, the appropriate level of model complexity is chosen using CV, to ensure independent data are used in making this decision.

Code Box 5.3: Computing Information Criteria on R for Exercise 5.1

The `AIC` or `BIC` function can be used to compute information criteria for many models:

```
> ft_r = lm(log(height)~rain,dat=globalPlants)
> ft_rs = lm(log(height)~rain+rain.seas,dat=globalPlants)
> c( AIC(ft_r), AIC(ft_rs) )
[1] 479.6605 475.4343
> c( BIC(ft_r), BIC(ft_rs) )
[1] 488.2861 486.9351
```

These results favour the larger model, although not by a lot, so results should be interpreted tentatively. Note that there is an advantage to the larger model when measured using the BIC score, because this criterion penalises complexity more harshly.

5.4.1 Pros and Cons of Information Criteria

Information criteria have the advantage that there are no random splits in the data—you get the same answer every time. This makes them simpler to interpret. (An exception is when using GIC with λ estimated by CV—in that case, the choice of λ can vary depending how the data are split into validation groups.)

The disadvantages are that they are slightly less intuitive than CV, derived indirectly as measures of predictive performance on new data, and in the case of AIC and BIC, their validity relies on model assumptions (essentially, the fitted models need to be close to the correct model). CV requires only the assumption that the

test/training data are independent—so it can still be used validly when you aren't sure about the fitted model.

This is the first example we shall see of the distinction between *model-based* and *design-based* inference; we will see more about this in Chap. 9.

5.5 Ways to Do Subset Selection

It's all well and good if you only have a few candidate models to compare, but what if you have a whole bunch of predictor variables and you just want to find the subset that is best for predicting y? Common approaches:

- Forward selection—add one variable at a time, adding the best-fitting variable at each step
- Backward selection—add all variables, then delete one variable at a time, deleting the worst-fitting variable at each step
- All subsets—search all possible combinations. For p predictors there are 2^p possible combinations, which is not easy unless there are only a few variables (p small), as in Code Box 5.4.

There are also hybrid approaches that do a bit of everything, such as the `step` function in R, as in Code Box 5.5.

Code Box 5.4: All Subsets Selection for the Plant Height Data of Exercise 5.1

```
> library(leaps)
> fit_heightallsub<-regsubsets(log(height)~temp+rain+rain.wetm+
      temp.seas, data=globalPlants,nbest=2)
```

The results are most easily accessed using `summary`, but we will look at two parts of the summary output side by side (the variables included in models, stored in `outmat`, and the BICs of each model, stored in `bics`):

```
> cbind(summary(fit_heightallsub)$outmat,summary(fit_heightallsub)$bic)
         temp rain rain.wetm temp.seas
1  ( 1 ) " "  " "  "*"       " "       "-21.06175277099"
1  ( 2 ) " "  "*"  " "       " "       "-19.2868231448677"
2  ( 1 ) "*"  "*"  " "       " "       "-24.8920679441895"
2  ( 2 ) "*"  " "  "*"       " "       "-23.9315826810965"
3  ( 1 ) "*"  "*"  " "       "*"       "-20.9786934545272"
3  ( 2 ) "*"  "*"  "*"       " "       "-20.3405400349995"
4  ( 1 ) "*"  "*"  "*"       "*"       "-16.4229239023018"
```

The best single-predictor model has just `rain.wetm` and has a BIC of about −21; the next best single-predictor model has just `rain`. But including both `temp` and `rain` does the best among the models considered here, with a BIC of about −25.

Code Box 5.5: Stepwise Subset Selection for Plant Height Data of Exercise 5.1

```
> ft_clim = lm(log(height)~temp+rain+rain.wetm+temp.seas,
   data=globalPlants)
> stepClim=step(ft_clim,trace=0)
> stepClim$anova
         Step Df  Deviance Resid. Df Resid. Dev       AIC
1             NA        NA       126   260.6727 100.13694
2 - rain.wetm  1 0.6363946       127   261.3091  98.45637
3 - temp.seas  1 1.9256333       128   263.2347  97.41819
```

This table lists the three steps that were taken along the variable selection path, which ended up being backward selection (as indicated by the - signs at Steps 2 and 3). In Step 1, all four predictors are in the model, and the AIC is about 100. At Step 2, `rain.wetm` is removed from the model, which has little impact on the log-likelihood (residual deviance is changed only a little) and reduces the AIC by almost 2. At Step 3, `temp.seas` is removed from the model, reducing the AIC slightly further and leaving a final model with `temp` and `rain` as the remaining predictors.

To do forward selection, you need to start with a model with no terms in it and specify a `scope` argument with a formula including all the terms to be considered:

```
> ft_int = lm(log(height)~1,data=globalPlants)
> stepForward <- step(ft_int,scope=formula(ft_clim),
     direction="forward",trace=0)
> stepForward$anova
         Step Df Deviance Resid. Df Resid. Dev       AIC
1             NA       NA       130   355.9206 132.93585
2 + rain.wetm -1 74.59845       129   281.3221 104.12370
3      + temp -1 16.15030       128   265.1718  98.37867
```

Again the path has three steps in it, but this time they involve the addition of variables to the model (as indicated by the + sign). Step 1 starts with an intercept model, but the AIC is much improved upon the addition of `rain.wetm` at Step 2. Step 3 adds `temp`, which also decreases the AIC.

Notice that we ended up with a slightly different model! But recall that `rain` and `rain.wetm` are highly correlated, so the two final models are actually quite similar.

So which method is best? Dunno! (There is no simple answer.) You can explore this yourself by simulating data and seeing how different methods go—no method is universally best. Results of a small simulation looking at this question are given in Fig. 5.4, but the simulation settings could readily be varied so that any of the three methods was the best performer.

All-subsets selection is more comprehensive but not necessarily better—because it considers so many possible models, it is more likely that some quirky model will jump out and beat the model with the important predictors, i.e. it is arguably less robust. In Fig. 5.4, all-subsets selection seemed to perform best when looking at AIC on training data (left), but when considering how close predicted values were to their true means, forward selection performed better (right). This will not necessarily be true in all simulations. Recall also that all subsets is simply not an option if you have lots of x variables.

Backward selection is not a good idea when you have many x variables—because you don't really want to use all of them, and the model with all predictors (the "full model") is probably quite unstable, with some parameters that are poorly estimated. In this situation, the full model is not the best place to start from and you would be better off doing forward selection. In Fig. 5.4 (right), backward selection was the worst performing measure, probably because the sample size (32) was not large compared to the number of predictors (8), so the full model was not a good starting point.

Forward selection doesn't work as well in situations where the final model has lots of terms in it—because the starting point is a long way from the final answer, there are more points along the way where things can go wrong. In Fig. 5.4 (right), it was the best performing method, probably in part because in this simulation only two of the predictors were associated with the response, so this method started relatively close to the true answer.

Multi-collinearity (Page 70) can muck up stepwise methods—well it will cause trouble for any variable selection method, but especially for stepwise methods where the likelihood of a term entering the model is dramatically reduced by correlation with a term already in the model, the end result being that the process is a lot more noisy. In Fig. 5.4, all predictors had a correlation of 0.5, and the true model was correctly selected about 20% of the time. When the correlation was increased to 0.8, the correct model was only selected about 5% of the time.

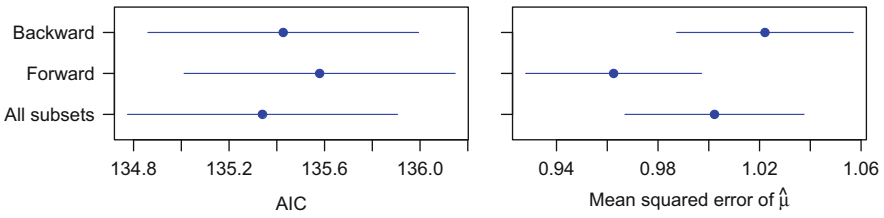

Fig. 5.4: Results of simulation looking at performance of different subset selection routines as measured using (left) AIC on training data and (right) MSE predicting the true mean response, for multiple linear regression with 8 predictors and 32 observations, of which 2 were related to the response. The mean AIC or MSE is reported across 1000 simulated datasets, together with a 95% confidence interval for the mean. All subset selection routines used AIC minimisation as their selection criterion. *All* pairwise contrasts were significant at the 0.05 level (using paired t-tests). Note that while all-subsets selection produced the smallest AIC values (indeed it always chose the model with the smallest possible AIC), it did not have the best predictive performance, as MSE was about 4% smaller for forward selection

5.6 Penalised Estimation

A modern and pretty clever way to do subset selection is to use penalised estimation. Instead of estimating model parameters (β) to minimise least squares

$$\min\left\{\sum_{i=1}^{n}(y_i-\mu_i)^2\right\}$$

we add a penalty as well, which encourages estimates towards zero, such as this one:

$$\min\left\{\sum_{i=1}^{n}(y_i-\mu_i)^2+\lambda\sum_j|\beta_j|\right\}$$

This approach is known as the LASSO (Tibshirani, 1996, least absolute shrinkage and subset selection operator). This is implemented in a lot of recently developed statistical tools, so many ecologists will have used the LASSO without realising it, e.g. in MAXENT software under default settings (Phillips et al., 2006).

This looks a bit like GIC, but the penalty term is a function of the size of the parameters in the model, rather than just the number of parameters. The effect is to push parameter estimates towards zero (so their penalty is smaller), especially for coefficients of variables that aren't very useful in predicting the response. This biases parameter estimates in order to reduce their sampling variance.

Penalised estimation is a good thing when

- the main goal is prediction—it tends to improve predictive performance (by reducing variance);
- you have lots of parameters in your model (or not a large sample size)—in such cases, reducing the sampling variance is an important issue.

The λ parameter is a *nuisance parameter* that we need to estimate to fit a LASSO model. The value of this parameter determines how hard we push the slope parameters towards zero, i.e. how much we bias estimates, in an effort to reduce variance. So this parameter is what manages the bias–variance trade-off.

λ is large $\implies$ most $\beta_j = 0$

λ is small $\implies$ few $\beta_j = 0$

The parameter λ controls model complexity, determining how many predictors are included in the model. The full range of model sizes is possible, from having no predictors included (if λ is large enough) to including all of them (as λ approaches zero and we approach the least-squares fit). We can choose λ using the same methods we used to choose model complexity previously—CV is particularly common, but BIC is known to work well also.

The LASSO can equivalently be thought of as constrained minimisation:

$$\min\left\{\sum_{i=1}^{n}(y_i-\mu_i)^2\right\}\text{ such that }\sum_j|\beta_j|\le t$$

In other words, it can be understood as a least-squares estimator that insists that the sum of the absolute values of all parameters is no larger than some nuisance parameter t (which is a known function of λ and the data).

Code Box 5.6: LASSO for Plant Height Data of Exercise 5.1

```
> library(glmnet)
> X = cbind(globalPlants$temp, globalPlants$rain,
  globalPlants$rain.wetm, globalPlants$temp.seas)
> ft_heightcv=cv.glmnet(X,log(globalPlants$height))
> plot(ft_heightcv)
> ft_lasso=glmnet(X,log(globalPlants$height),
  lambda=ft_heightcv$lambda.min)
> ft_lasso$beta
```

Some good news about the LASSO:

- It reduces the sampling variability in parameters, and in predictions, by shrinking them towards zero.
- It does model selection as part of the estimation process. This happens because some (or many, depending on the data) parameter estimates are forced to zero by the LASSO penalty term, and if a parameter estimate is zero, that term has been excluded from the model.
- It's fast compared to other model selection methods (Friedman et al., 2010).
- It predicts well (by reducing variance). In fact, use of the LASSO is probably the main reason why MAXENT software has done well in comparisons of different methods of species distribution modelling (Elith et al., 2006; Renner & Warton, 2013).
- It simplifies the problem of model selection to one of estimating a single parameter (λ). Pretty cool that a problem involving choosing between 2^p candidate models (which could be in the millions or larger, see Table 5.1) can be simplified to estimating just one nuisance parameter.

And the bad news:

- It biases parameter estimates—relationships are flatter than they should be. There are variations on the LASSO to address this (such as the *adaptive LASSO*, Zou, 2006).
- Obtaining standard errors is complicated. (And how useful are standard errors when we know our estimates are biased anyway?) There are approximate methods for getting standard errors (Fan & Li, 2001), but they are rarely implemented in software and hard to interpret without concurrent information about the bias in estimates.

The LASSO, and related developments in fitting sparse models to data (i.e. models forcing lots of parameters to zero), is one of the more exciting developments in statistics over the last couple of decades, and there is a huge and rapidly growing literature on it in statistics journals.

5.7 Variable Importance

Exercise 5.2: Relative Importance of Climate Variables
Consider again Angela's height data (Exercise 5.1) and four variables—average annual temperature (`temp`), total precipitation (`rain`), rainfall in the wettest month (`rain.wetm`), and variation in mean monthly temperature (`temp.seas`).

How important are the different climate variables in explaining plant height?

Sometimes we are interested not just in which variables best predict a response, but how important they are relative to each other, as in Exercise 5.2. There are a few options here in terms of how to approach this sort of problem.

Recall the difference between *marginal* and *conditional* effects (Sect. 3.1.2)—the estimated effect of a predictor will change depending on what other terms are included in the model, because linear models estimate conditional effects. So when measuring the relative importance of predictors, we can expect to get different answers depending on what other terms are included in the model, as indeed happens in Code Boxes 5.7 and 5.8.

One option is to use forward selection to sequentially enter the predictor that most reduces the sum of squares at each step (Code Box 5.7). This is one way to order variables from most important to least, but not the only way (e.g. backward selection, which often leads to a different ordering). The table in Code Box 5.7 is intuitive, breaking down the overall model R^2 into components due to each predictor, but it does so in a misleading way. By adding terms sequentially, the R^2 for the first predictor `temp` estimates the *marginal* effect of temperature, because it was added to the model before any other predictors. But by the time `temp.seas` was added to the model, all other predictors had been included, so its *conditional* effect was being estimated in Code Box 5.7. So we are comparing "apples with oranges"—it would be better to either include all other predictors in the model or none of them when quantifying the relative effects of predictors.

Code Box 5.7: Sequential R^2 for Variable Importance
Proportion of variance explained R^2 will be used to measure importance of each predictor; it can be calculated by dividing the sum of squares explained by the sum of squares from a model with no predictors in it (`ft_int`).

Let's enter the variables sequentially:

```
> ft_clim = lm(log(height)~temp+rain+rain.wetm+temp.seas,
      data=globalPlants)
> ft_int = lm(log(height)~1,data=globalPlants)
> stepAnova = step(ft_int, scope=formula(ft_clim),
      direction="forward", trace=0, k=0)$anova
> stepAnova$R2=stepAnova$Deviance/deviance(ft_int)
```

```
> stepAnova
          Step Df  Deviance Resid. Df Resid. Dev       AIC          R2
1             NA        NA       130   355.9206 130.93585          NA
2 + rain.wetm -1 74.598450       129   281.3221 100.12370 0.209592965
3      + temp -1 16.150298       128   265.1718  92.37867 0.045376129
4      + rain -1  2.586703       127   262.5851  91.09452 0.007267641
5 + temp.seas -1  1.912441       126   260.6727  90.13694 0.005373225
```

`Deviance` means the same thing as sum of squares for a linear model, and `deviance(ft1)` gets the total sum of squares needed to construct R^2. In the line calling the `step` function, setting `k=0` ensures no penalty when adding terms, so that all four variables get added, in decreasing order of importance.

We can see that `rain.wetm` explains about 21% of variation in plant height, `temp` adds another 5%, and the other two variables do very little.

But there are different ways we could add these terms to the model! See Code Box 5.8 for alternatives.

Code Box 5.8: Marginal and Conditional R^2 for Variable Importance

In Code Box 5.7, we sequentially calculated R^2 on adding each term to the model in a way that maximised the explained variation at each step. Alternatively, we could look at the effect of the predictors one at a time, their *marginal* effect:

```
> stepMargin=add1(ft_int,scope=formula(ft_clim))
> stepMargin$R2=stepMargin$`Sum of Sq`/deviance(ft_int)
> stepMargin
          Df Sum of Sq    RSS    AIC      R2
<none>                  355.92 132.94
temp       1    66.224 289.70 107.97 0.18607
rain       1    70.761 285.16 105.90 0.19881
rain.wetm  1    74.598 281.32 104.12 0.20959
temp.seas  1    46.401 309.52 116.64 0.13037
```

It seems that either `temp` or `rain` explains about as much variation in plant height as `rain.wetm` did, some information we were missing from Code Box 5.7.

Alternatively, we could measure the *conditional* effect of each predictor by sequentially leaving each predictor out of the model while keeping all others in:

```
> leave1out=drop1(ft_clim)
> leave1out$R2=leave1out$`Sum of Sq`/deviance(ft_int)
> leave1out
          Df Sum of Sq    RSS     AIC       R2
<none>                  260.67 100.137
temp       1   16.0581 276.73 105.968 0.045117
rain       1    3.4438 264.12  99.856 0.009676
rain.wetm  1    0.6364 261.31  98.456 0.001788
temp.seas  1    1.9124 262.58  99.095 0.005373
```

These values are much smaller because they only capture variation explained by a predictor that is not explained by anything else. Leaving out `temp` increases the sum of squares by 4.5%; leaving out other terms increases error by 1% or less.

Code Box 5.8 presents two alternatives: first, estimating the *marginal* effect of a predictor, with the variation in height explained by this predictor when included in the model by itself; or, second, estimating the *conditional* effect of a predictor,

with the variation in height it explains not being captured by other predictors. Fitting the full model and looking at standardised coefficients is another and more or less equivalent way to look at conditional effects (Code Box 5.9). The advantage of using standardised coefficients is that this method can be readily applied to other types of models (e.g. using a LASSO).

Looking at marginal vs conditional effects can give quite different answers (as in Code Box 5.8), especially when predictors are correlated. Neither of these is a perfect way to measure what is happening, and both seem to miss some details.

Code Box 5.9: Standardised Coefficients for Angela's Height Data

Looking at standardised coefficients:

```
> # first create a dataset with standardised predictors:
> globalPlantStand=globalPlants
> whichVars=c("temp","rain","rain.wetm","temp.seas")
> globalPlantStand[,whichVars]=scale(globalPlantStand[,whichVars])
> # then fit the model:
> ft_climStand = lm(log(height)~temp+rain+rain.wetm+temp.seas,
                data=globalPlantStand)
> summary(ft_climStand)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.1947     0.1257   9.507  < 2e-16 ***
temp          0.5715     0.2051   2.786  0.00616 **
rain          0.4185     0.3244   1.290  0.19934
rain.wetm     0.1860     0.3353   0.555  0.58013
temp.seas     0.2090     0.2174   0.961  0.33816
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Again we see that `temp` is the most important predictor, followed by `rain`.

The problem with looking at marginal effects only is that if two predictors are highly correlated, measuring very similar things, then both can have large marginal effects, even if one predictor is not directly related to the response. For example, the marginal effect of temperature seasonality is $R^2 = 13\%$ (Code Box 5.8, `temp.seas`), but there seems to be little effect of `temp.seas` on plant height after annual temperature (`temp`) has been added to the model (Code Box 5.7, $R^2 < 1\%$ for `temp.seas`). It seems that the marginal R^2 for `temp.seas` was as high as 13% simply because it is correlated with `temp`. The `temp` predictor on the other hand does seem to be important, because even after including other predictors in the model, it still explains about 5% of variation in plant height (Code Box 5.8).

The problem with looking at conditional effects only is that if two predictors are highly correlated, measuring very similar things, then the conditional effect of each is small. For example, total precipitation (`rain`) and rainfall in the wettest month (`rain.wetm`) are very highly correlated (Code Box 3.7). Hence the conditional effect of each, after the other has already been included in the model, is small (Code Box 5.8) because most of the information captured in `rain` has already entered the model via `rain.wetm`, and vice versa. However, rainfall is clearly important to the distribution of plant height—it was in all models produced by best-subsets selection

(Code Box 5.4) and actually explains about 8% of variation after temperature has been added to the model; a leave-one-out approach misses this part of the story because rainfall enters the model via two variables. We would need to leave both rainfall variables out to see this—as in Code Box 5.10.

Exercise 5.3: Variable Importance Output

Compare the R^2 results of Code Boxes 5.7 and 5.8. Which table(s) do you think Angela should report when describing variable importance?

Now look at the standardised coefficients in Code Box 5.9. Do these coefficients measure marginal or conditional effects? Which of the R^2 tables in Code Box 5.8 are they most similar to in relative size (e.g. ranking from largest to smallest)? Is this what you expected?

Code Box 5.10: Importance of Temperature vs Rainfall

Yet another approach is to classify predictors as either temperature or rainfall predictors and study their relative effect. Looking at the effect of rainfall after temperature:

```
> ft_onlyTemp = lm(log(height)~temp+temp.seas,data=globalPlants)
> tempAn=anova(ft_int,ft_onlyTemp,ft_clim)
> tempAn$R2=tempAn$`Sum of Sq`/deviance(ft_int)
> tempAn
Analysis of Variance Table

Model 1: log(height) ~ 1
Model 2: log(height) ~ temp + temp.seas
Model 3: log(height) ~ temp + rain + rain.wetm + temp.seas
  Res.Df    RSS Df Sum of Sq      F     Pr(>F)       R2
1    130 355.92
2    128 289.00  2    66.917 16.173 0.00000056 0.188011
3    126 260.67  2    28.331  6.847 0.00150359 0.079599
```

Looking at the effect of temperature after rainfall:

```
> ft_onlyRain = lm(log(height)~rain+rain.wetm,data=globalPlants)
> rainAn=anova(ft_int,ft_onlyRain,ft_clim)
> rainAn$R2=rainAn$`Sum of Sq`/deviance(ft_int)
> rainAn
Model 1: log(height) ~ 1
Model 2: log(height) ~ rain + rain.wetm
Model 3: log(height) ~ temp + rain + rain.wetm + temp.seas
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)       R2
1    130 355.92
2    128 279.80  2    76.118 18.3964 0.0000001 0.213863
3    126 260.67  2    19.130  4.6233 0.0115445 0.053747
```

temperature seems able to explain about 19% of global variation in plant height; then rainfall can explain about 8% more, whereas over 21% of variation can be explained by rainfall alone. This idea is visualised in Fig. 5.5.

For Angela's height data, one solution is to aggregate variables into types (temperature vs rainfall) and look at the importance of these variable types as a unit,

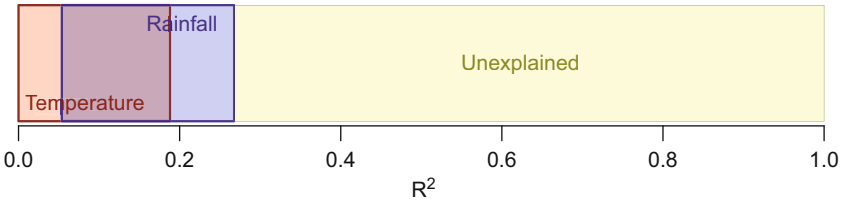

Fig. 5.5: Schematic diagram of relative importance of temperature and rainfall for Angela's height data, based on results in Code Box 5.10. Temperature and rainfall variables jointly explain about 27% of global variation in plant height for Angela's data, but temperature and rainfall each on their own explain closer to 20% of variation. About 5% of variation can be attributed to temperature, about 8% to rainfall, and the remaining 14% could be explained by either (it is *confounded*). This sort of plot, while conceptually helpful, is difficult to generalise to several predictors

as in Code Box 5.10. Each of temperature and rainfall, on its own, seems able to explain about 20% of variation in plant height, but adding the other variable type as well explains an additional 5–8% of variation, as visualised in Fig. 5.5. So we may conclude that temperature and rainfall are both important, separately, rainfall perhaps slightly more so, but we can do a better job (about 5% better) at explaining global variation in plant height by looking at temperature as well.

Plenty of alternative approaches could be used here. The simplest is to reduce multi-collinearity by removing highly correlated responses—this reduces the overlap, so conditional and marginal effects become more comparable. For Angela's data, we could have reduced the dataset to one temperature and one rainfall variable—a model with just `temp` and `rain`, for example, which ended up being suggested by `step` anyway (Code Box 5.5). Another option is to use structural equation modelling (Grace, 2006) to explicitly build into the model the idea that while temperature and rainfall are important, each is measured using multiple predictors. A more controversial option is to use a technique that averages measures of variable importance across different choices of model, which has been very popular in some parts of ecology under the name *hierarchical partitioning* (Chevan & Sutherland, 1991). The issue with that type of approach is that coefficients have different meanings depending on what other terms are included in the model—recall linear models estimate conditional effects, so changing what terms are in the model changes what we are conditioning on. So it makes little sense to average measures of variable importance across different models, which condition on different things, meaning we are measuring different things.

5.8 Summary

Say you have a model selection problem, like Angela's (Exercise 5.1). We have seen a suite of different tools that can be used for this purpose. So what should she actually do? Well, she could try a number of these methods; the important thing is to abide by a few key principles:

- Model selection is difficult and will be most successful when there are only a few models to choose between—so it is worth putting careful thought into what you actually want to compare, and deciding whether you can shortlist just a few candidate models.
- A key step is choosing model complexity—how many terms should be in the model? Too few means your model will be biased, too many means its predictions will be too variable, and we are looking to choose a model somewhere in the middle. A good way to choose the model complexity for your data is to consider how well different models predict to new, *independent data*—directly, typically using some type of CV, or indirectly, using information criteria.
- If you do have a whole heap of predictors, penalised estimation using methods like the LASSO is a nice solution that returns an answer quickly, meaning it is applicable to big data problems. Stepwise methods can also be useful, but it is worth starting them from a model near where you think the right model will be. For example, if you have many predictors but only want a few in the final model, it would be much better to use forward selection starting with a model that has no predictors in it than to use backward selection from a model with all predictors in it.

In the analyses for Angela's paper (Moles et al., 2009), there were 22 potential predictors, which we shortlisted to 10; then I specially wrote some code to use all-subsets selection and CV to choose the best-fitting model. With the benefit of hindsight I don't think the added complexity implementing an all-subsets algorithm justified the effort, as compared to, say, forward selection. A decade on, if dealing with a similar problem, I would definitely still shortlist variables, but then I would probably recommend a LASSO approach.

Model selection is an active area of research, and the methods used for this problem have changed a lot over the last couple of decades, so it is entirely possible that things will change again in the decades to come!

Exercise 5.4: Head Bobs in Lizards—Do Their Displays Change with the Environment?
Terry recorded displays of 14 male *Anolis* lizards in the wild (Ord et al., 2016). These lizards bob their head up and down (and do push-ups) in attempts to attract the attention of females. Terry measured how fast they bobbed their heads and wanted to know which environmental features (out of temperature,

light, and noisiness) were related to head bobbing speed. The data, with one observation for each lizard, can be found in the `headbobLizards` dataset.

What type of inference method is appropriate here?

What sort of model would you fit?

Load the data and take a look. Would it make sense to transform any of the variables in the data prior to analysis?

Which environmental variables best predict head bob speed?

Exercise 5.5: Plant Height Data and Precipitation

Consider Angela's global plant height data of Exercise 5.1. Angela collected data on how tall plants are in lots of different places around the globe. She also has data on eight different precipitation variables. She is interested in how plant height relates to precipitation and *which precipitation variables height relates to most closely*.

Find a subset of precipitation variables that optimally predicts plant height. Try a couple of different methods of model selection.

Any issues with multi-collinearity among the precipitation variables? Try to address any multi-collinearity by culling one or two of the main culprits. Does this affect your previous model selection results?