Tips on word frequencies

I found a slightly another approach to getting words from a plain text document using the `scan()` function.

Here is an example:

```
jamess-mbp:STAT476 superjames$ cat fox.txt
The quick brown fox jumped over the lazy dog.
Memory believes before knowing remembers.
```

I want to convert to lower case and remove all periods.

Tips on word frequencies

To get a vector of words

```
> x <- scan("fox.txt",what="char")
Read 14 items
> x
 [1] "The"         "quick"       "brown"       "fox"         "jumped"
 [6] "over"        "the"         "lazy"        "dog."        "Memory"
[11] "believes"    "before"      "knowing"     "remembers."
> x <- tolower(x)
> x <- gsub(".","",x,fixed=TRUE)
> x
 [1] "the"         "quick"       "brown"       "fox"         "jumped"     "o
 [7] "the"         "lazy"        "dog"         "memory"      "believes"   "be
[13] "knowing"     "remembers"
```

Tips on word frequencies

```
> y <- scan("romeoandjuliet.txt",what="char")
Read 23995 items
> length(unique(y))
[1] 6166
> y <- tolower(y)
> length(unique(y))
[1] 5705
> y2 <- chartr(",'.;!:[]?","          ",y)
> length(unique(y2))
[1] 4658
> y3 <- gsub(" ","",y2)
> length(unique(y3))
[1] 3714
```

Tips on word frequencies

```
> words[1:20]
y3
  and   the     i    to     a    of    my    is  that    in romeo
  682   624   540   498   419   364   337   321   319   301   301
   me   not  with    it  this   for    be
  247   237   232   210   208   204   192
> words <- sort(table(y3)/length(y3),decreasing=TRUE)
> words[1:20]
y3
        and         the           i          to           a
0.028422588 0.026005418 0.022504688 0.020754324 0.017461971 0.01516
         my          is        that          in       romeo
0.014044593 0.013377787 0.013294436 0.012544280 0.012544280 0.01091
       thou          me         not        with          it
0.010710565 0.010293811 0.009877058 0.009668681 0.008751823 0.00866
        for          be
0.008501771 0.008001667
```

Classification analysis (Chapter 9)

Now we go back to chapter 9, which deals with classification of observations to groups. You might have several groups already defined and want to classify a new observation. If you have a PCA, then you could determine the linear combinations of variables corresponding to PC1 and PC2 that was determined from an original set of data. Then you could use those linear combinations on a new data point (even if it didn't contribute to the calculation of the PCs) and see where it fits on plot of PC2 versus PC1.

If clusters have been defined from the original data, you might want to find which cluster the new observation is closest to.

Classification analysis

For a biological example, there might several known species, each of which is considered a cluster. A new organism is sampled from the field, and the question is: to which species does it belong? Sometimes species identification can be difficult. For example, there are 16,000 species of mushrooms (compared to about 5,000 species of mammals). Several thousand new species are discovered and described every year, and a difficult question is whether a new specimen is a new species or just a variety of a previously described species.

Other examples of classification might be a bank deciding whether a mortgage application should be considered high risk, medium risk, or low risk. There are numerous variables that could be used for classification: amount of the mortgage, downpayment, number of borrowers, income(s) of the borrower(s), credit rating, whether the house is for residence or rental, etc. A bank will have a large database of previous experience on these variables that can be used to make a model, and the interest is in classifying a new observation (a new customer).

Classification analysis

Some other examples of classification include:

1. classifying a tumor as benign or malignant based on a medical image
2. making a diagnosis (medical or psychiatric) on the basis of a set of symptoms (this is more open-ended than benign versus malignant)
3. classifying a fossil bone as belonging to a male or female, adult or juvenille
4. classifying student applicants as likely to complete college or drop out
5. finding the best fit for an applicant for a college major, or for a job within the military
6. determining disputed authorship (Hamilton versus Madison for the *Federalist Papers* or Shakespeare's plays
7. identifying speech patterns for automated voice recognition systems

Classification analysis

A first step is to classify observations into one of two categories. If we have two previous populations (or historical data), then we can determine their historical averages. For example, if we want to classify high school students applying to college as likely to succeed or not, we can use historical data for those who succeeded (say, graduated from college within a given number of years) and measure variables such as high school rank, high school GPA, and SAT/ACT scores.

One approach to this problem is to use the discriminant function. Recall from chapter 5 that the discriminant function is the vector $\mathbf{a}$ where

$$z = \mathbf{a}'\mathbf{y} = (\mathbf{y}_1 - \mathbf{y}_2)'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y}$$

$$\overline{z}_i = \mathbf{a}'\mathbf{y}_i = (\mathbf{y}_1 - \mathbf{y}_2)'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y}_i$$

here $\mathbf{y}_i$ is the multivariate average vector for population i, $\mathbf{S}_{\text{pl}}$ is the pooled covariance matrix, and $\mathbf{y}$ is the new observation.

Classification analysis

Fisher's approach is to classify observation **y** as coming from population 1 if z is closer to $\bar{z}_1$ and coming from population2 if z is closer to $\bar{z}_2$. The vector **a** represents the linear combination of the variables that best seperates the two groups before **y** is considered.

Because z is a linear combination of the **y** variables, it is somewhat similar to a principal component, and we can think of it as a rotated axis. Principal components is different in that the principal components procedure doesn't take into account group labels. PCA finds the axis of greatest variation in the pooled data, which might be different from the axis that best separates two groups.

Classification

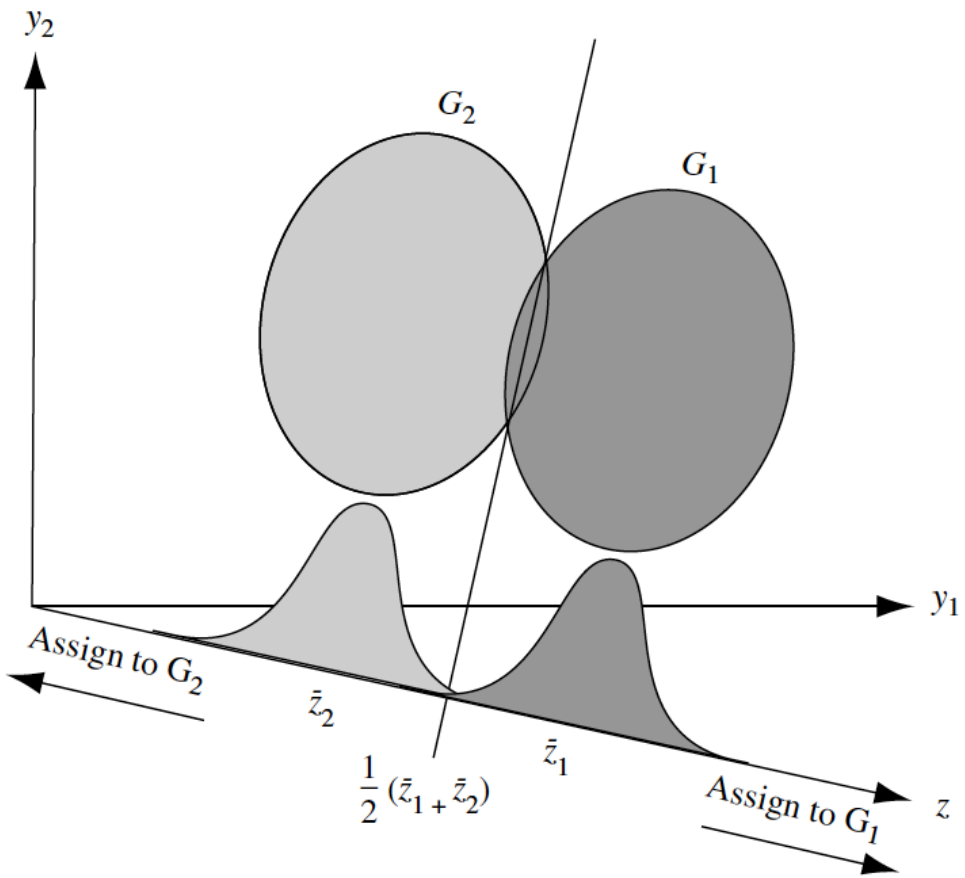


Figure 9.1. Fisher's procedure for classification into two groups.

Classification

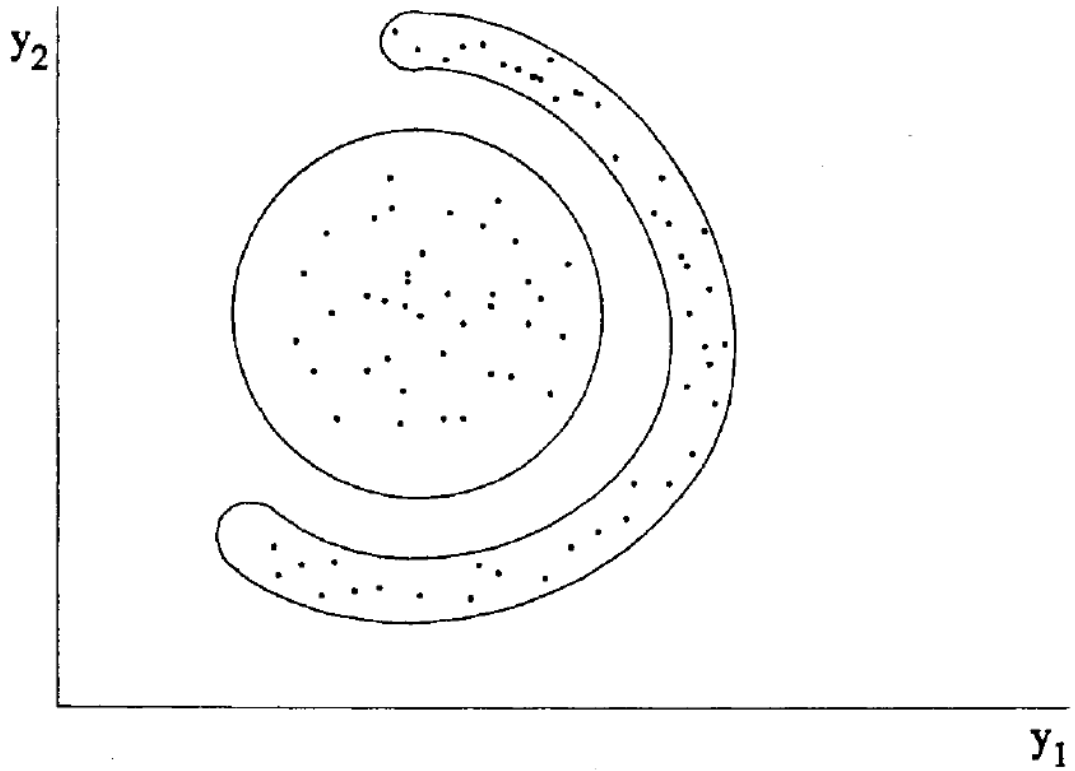


Figure 9.2. Two populations with nonlinear separation.

Classification

The way the linear combination $\mathbf{a}$ is constructed, $\overline{z}_1 - \overline{z}_2 > 0$. This might seem a bit odd, but we have that $\mathbf{a} = (\mathbf{y}_1 - \mathbf{y}_2)\mathbf{S}_{\text{pl}}^{-1}$, and

$$\begin{aligned}
\overline{z}_1 - \overline{z}_2 &= \mathbf{a}'\overline{\mathbf{y}}_1 - \mathbf{a}'\overline{\mathbf{y}} \\
&= \mathbf{a}'(\overline{\mathbf{y}}_1 - \overline{\mathbf{y}}_2) \\
&= (\overline{\mathbf{y}}_1 - \overline{\mathbf{y}}_2)'\mathbf{S}_{\text{pl}}^{-1}(\overline{\mathbf{y}}_1 - \overline{\mathbf{y}}_2) > 0
\end{aligned}$$

which is a quadratic form and is positive because $\mathbf{S}_{\text{pl}}^{-1}$ is positive definite.

Classification

As a result, z is closer to $\overline{z}_1$ if

$$z > \frac{1}{2}(\overline{z}_1 + \overline{z}_2),$$

the midpoint between $\overline{z}_1$ and $\overline{z}_2$. In terms of $\mathbf{y}$ (instead of z) we can write that $\mathbf{y}$ is classified as belonging to group 1 if

$$\mathbf{a}'\mathbf{y} = (\overline{\mathbf{y}}_1 - \overline{\mathbf{y}}_2)'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} > \frac{1}{2}(\overline{\mathbf{y}}_1 - \overline{\mathbf{y}}_2)'\mathbf{S}_{\text{pl}}^{-1}(\overline{\mathbf{y}}_1 + \overline{\mathbf{y}}_2)$$

If the inequality goes the other way, then $\mathbf{y}$ is classified as belonging to group 2, and if there is equality (which happens with probability 0 if there are any continuous variables involved), then the classification is arbitrary, or you could flip a coin, or have a default group that you prefer (default to denying or accepting mortgages).

Classification

The method described above was used by Fisher in the 1930s and doesn't assume that the data is multivariate normal. However, if the data is multivariate normal and the two groups satisfy $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$, then this classifier can be considered asymptotically optimal in the sense that the probability of misclassification is minimized by this procedure.

The method can also be used in conjunction with prior probabilities for the two groups if the densities for the groups is also known (so, usually if you are willing to assume multivariate normality). In this case, if group 1 has density $f_1(\mathbf{y})$ and prior probability p_1, and group 2 has density $f_2(\mathbf{y})$ and prior probability p_2, then you classify $\mathbf{y}$ as coming from group 1 if

$$p_1 f_1(\mathbf{y}) > p_2 f_2(\mathbf{y})$$

(the book writes $f_1(\mathbf{y})$ as $f(\mathbf{y}|G_1)$ for the density of $\mathbf{y}$ given that you come from group 1.

Classification

Assuming multivariate normality with equal covariance matrices (but unequal mean vectors) for groups 1 and 2, the classification rule is equivalent to preferring group 1 if

$$\mathbf{a}'\mathbf{y} = (\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} > \frac{1}{2}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'\mathbf{S}_{\text{pl}}^{-1}(\bar{\mathbf{y}}_1 + \bar{\mathbf{y}}_2) + \ln\left(\frac{p_1}{p_2}\right)$$

An example of using these prior probabilities is if you know that 70% of entering freshmen eventually graduate with probability 0.7, so that you use $p_1 = 0.7$ and $p_2 = 0.3$. Using prior probabilities might be more important for cases where one group is much more likely than another.

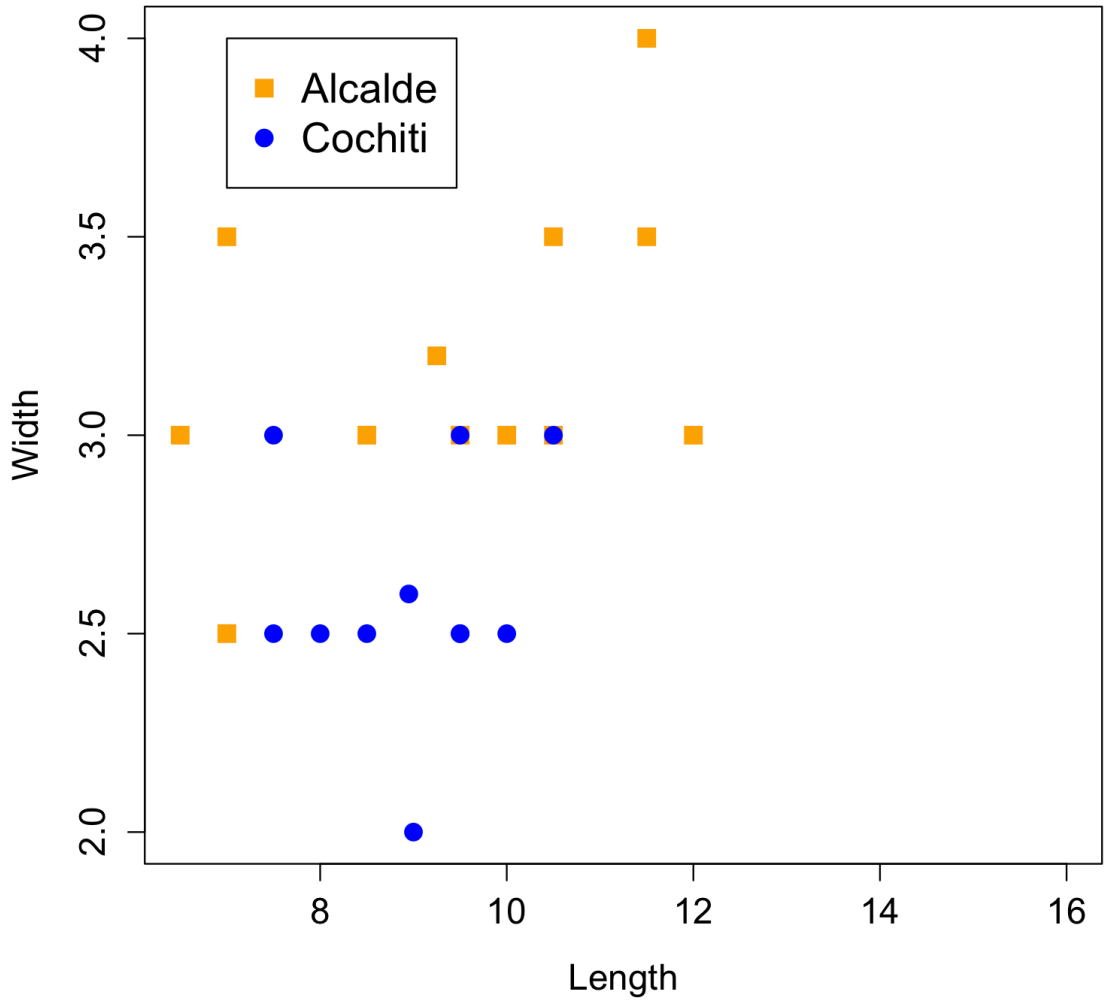
Note that if $p_1 = p_2$, then the normal-based classfication rule reduces to Fisher's linear discriminant function. Although Fisher's rule doesn't assume multivariate normality, it performs especially well with multivariate normality and $p_1 \approx p_2$.

Classification

In addition to classifying new observations, you can use discriminant functions to see how well you can classify your own data. Your misclassification rate in this case is biased in the sense that you will probably do better at classifying data used to create the model than new data, but nevertheless, a linear classifier often won't be able to perfectly classify data.

We'll do an example where we try to classify chile pods as coming from either Cochiti or Alcalde based on length and width alone (ignoring thickness of the wall).

Classification



Classification

This was the code I used to generate the plot:

```
> x <- read.table("chile.txt",header=T)
>  #first make an empty plot with the right dimensions
> plot(x$Length,x$Width,type="n",cex.axis=1.3,
cex.lab=1.3,xlab="Length",ylab="Width")
> points(x$Length[12:22],x$Width[12:22],
col="orange",pch=15,cex=1.5)
> points(x$Length[23:33],x$Width[23:33],
col="blue",pch=16,cex=1.5)
> legend(7,4,legend=c("Chimayo","Cochiti"),
pch=c(15,16),col=c("orange","blue"),cex=1.5)
```

Classification

You can see that if we just used Length or Width alone, then we'd have to misclassify several of the chile pods. So a question is whether we can use a combination of Length and Width to do a better job of classification? Of course, this is a small sample, so whatever rule we came up with might not generalize well to larger samples, but this will give the idea of the approach.

Classification

First we'll determine the discriminant function. To find $\mathbf{a}$ we set

$$\mathbf{a} = (\mathbf{y}_1 - \mathbf{y}_2)'\mathbf{S}_{\text{pl}}^{-1}$$

```
> y <- x[c(1:11,34:44),2:3]
> y1bar <- colMeans(y[1:11,])
> y2bar <- colMeans(y[12:22,])
> y1bar
Length  Width
  9.25   3.20
> y2bar
Length  Width
  8.95   2.60
```

Classification

Here are the pooled covariance matrix and its inverse

```
> n1 <- 11
> n2 <- 11
> S1 <- cov(y[1:11,])
> S2 <- cov(y[12:22,])
> Sp <- (n1*S1 + n2*S2)/(n1+n2)
> Spinv <- solve(Sp)
> Spinv
             Length      Width
Length  0.5804841 -0.9984327
Width  -0.9984327  9.7173042
> Sp
       Length Width
Length 2.0925 0.215
Width  0.2150 0.125
```

Classification

The vector **a** and classification rule criterion are

```
> a <- (y1bar-y2bar) %*% Spinv
> a
         Length    Width
[1,] -0.4249144 5.530853
> .5*(y1bar-y2bar) %*% Spinv %*% (y1bar+y2bar)
          [,1]
[1,] 12.17275
```

Thus the discriminant function is

$$-0.425 Length + 5.53 Width$$

and the classification says to assign a chile pod to Alcade if

$$-0.425 Length + 5.53 Width \; > \frac{1}{2}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)' \mathbf{S}_{\text{pl}}^{-1} (\bar{\mathbf{y}}_1 + \bar{\mathbf{y}}_2) = 12.1725$$

Classification

To see how our classifcation rule will classify the original 22 data points, we compute **Ya** where **a** is 2×1 and **Y** is 22×2. I did this in R as

```
> combo <- as.matrix(y) %*% t(a)
> classifier <- cbind(combo,(combo > 12.1725),
c(rep(1,11),rep(0,11)))
```

Classification

```
> classifier # linear combo, classification, truth
1  12.130957    0    1
2  16.383584    1    1
3  14.896384    1    1
4  17.236896    1    1
5  14.471469    1    1
6  12.555872    1    1
7  13.830615    1    1
8  12.980786    1    1
9  12.343414    1    1
10 10.852731    0    1
11 13.768271    1    1
34 10.640274    0    0
35 12.555872    1    0
36  9.790445    0    0
37 10.427817    0    0
38 12.130957    0    0
39  9.577988    0    0
40  7.237476    0    0
41  9.790445    0    0
42 13.405700    1    0
43 10.215360    0    0
44 10.577233    0    0
```

Classification

From these results, we see that four observations were missclassified, two from Alcalde would be classified as Cochiti, observations 1 and 10, and two from Cochiti would be classified as being from Alcalde, observations 35 and 42 (corresponding to the 2nd and 9th chiles from Cochiti). This was disappointing as the classification rule: "If width is greater than 2.7, assign it to Alcalde" would get an equal error rate.

However, this illustrates an automated procedure that can generalize to more dimensions.

Back to classification

In addition to classification into two groups, we often want to classify into several groups, for example classifying an employee into an appropriate training program or diagnosing a patient based on symptoms.

You can get a mean vector for each of k groups: $\mathbf{overline{y}}_1, \ldots, \overline{\mathbf{y}}_k$. The simplest approach is that a new observation $\mathbf{y}$ can be assigned to the group i for which the distance

$$d_i(\mathbf{y}) = d(\mathbf{y}, \overline{\mathbf{y}}_i)$$

is minimized.

Classification with more than two groups

More generally, however, you might want to account for the variation in the different groups. In this case, we find the group i that minimizes

$$D_i^2(\mathbf{y}) = (\mathbf{y} - \bar{\mathbf{y}})' \mathbf{S}_{\text{pl}}^{-1} (\mathbf{y} - \bar{\mathbf{y}})$$

Note that minimizing a squared distance is equivalent to minimizing a distance even though a squared distance is not generally a distance.

Classification with more than two groups

If we expand D_i^2, we get

$$D_i^2 = (\mathbf{y} - \overline{\mathbf{y}}_i)'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i)$$

Classification with more than two groups

If we expand D_i^2, we get

$$\begin{aligned} D_i^2 &= (\mathbf{y} - \overline{\mathbf{y}}_i)'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) - \overline{\mathbf{y}}_i'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \end{aligned}$$

Classification with more than two groups

If we expand D_i^2, we get

$$\begin{aligned} D_i^2 &= (\mathbf{y} - \overline{\mathbf{y}}_i)'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) - \overline{\mathbf{y}}_i'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i - \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} + \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i \end{aligned}$$

Classification with more than two groups

If we expand D_i^2, we get

$$\begin{aligned} D_i^2 &= (\mathbf{y} - \overline{\mathbf{y}}_i)'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) - \overline{\mathbf{y}}_i'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i - \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} + \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i \\ &= \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - 2\overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} + \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i \end{aligned}$$

Classification with more than two groups

If we expand D_i^2, we get

$$\begin{aligned} D_i^2 &= (\mathbf{y} - \overline{\mathbf{y}}_i)'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) - \overline{\mathbf{y}}_i'(\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i) \\ &= \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i - \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} + \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i \\ &= \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} - 2\overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\mathbf{y} + \overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i \end{aligned}$$

The first term, doesn't depend on i, so it can be ignored. The second term is a linear combination of the $\mathbf{y}$ terms, which is why this is called a linear classification. The third term doesn't depend on $\mathbf{y}$ is therefore constant with respect to $\mathbf{y}$. Therefore, minimizing D_i^2 is equivalent to minimizing

$$-2\mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i$$

Classification with more than two groups

To make the classification rule more similar to the two-group case, you can multiply by $-\frac{1}{2}$ and maximize

$$L_i(\mathbf{y}) = \mathbf{y}'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i - \frac{1}{2}\overline{\mathbf{y}}_i'\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i$$

(Maximizing $-\frac{1}{2}f(x)$ is equivalent to minimizing $f(x)$.) We can also write $L_i(\mathbf{y})$ as

$$L_i(\mathbf{y}) = \mathbf{c}_i'\mathbf{y} + c_{i0} = c_{i1}y_1 + c_{i2}y_2 + \cdots, c_{ip}y_p + c_{i0}$$

where

$$\mathbf{c}_i' = \overline{\mathbf{y}}_i\mathbf{S}_{\text{pl}}^{-1}, \quad c_{i0} = -\frac{1}{2}\overline{\mathbf{y}}_i\mathbf{S}_{\text{pl}}^{-1}\overline{\mathbf{y}}_i$$

Classification with more than two groups

If there are prior probabilities $p_1, \ldots, p_g$, on groups, one would assign an observation to the group that minimizes

$$p_i f_i(\mathbf{y})$$

If group i is $N_p(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ (i.e., all covariances are the same), then using prior probabilities is equivalent to minimizing

$$L_i(\mathbf{y}) = \ln p_i + \mathbf{y}' \mathbf{S}_{\text{pl}}^{-1} \overline{\mathbf{y}}_i - \frac{1}{2} \overline{\mathbf{y}}_i' \mathbf{S}_{\text{pl}}^{-1} \overline{\mathbf{y}}_i$$

Classification with more than two groups

The previous approach is sensitive to the assumption that the variances are equal — otherwise the new observation $\mathbf{y}$ might appear too close to spread out groups. An alternative is to use the individual covariance matrices instead of the pooled covariance matrix. In this case, you pick the group i that minimizes

$$D_i^2(\mathbf{y}) = (\mathbf{y} - \overline{\mathbf{y}})' \mathbf{S}_i^{-1} (\mathbf{y} - \overline{\mathbf{y}})$$

The only difference here is that $\mathbf{S}_{\text{pl}}$ was replaced with $\mathbf{S}_i$. In this case D_i^2 is quadratic in $\mathbf{y}$ rather than linear, so it is called a quadratic classification rule.

It is computationally a bit less efficient than a linear classification rule (as you might expect) because it requires computing i inverses of $p \times p$ matrices rather than pooling the covariances first and then computing the inverse of only one $p \times p$ matrix.

Classification with more than two groups

If prior probabilities are used, the rule is to minimize

$$Q_i(\mathbf{y}) = \ln p_i + \frac{1}{2} \ln |\mathbf{S}_i| + \mathbf{y}' \mathbf{S}_i^{-1} \overline{\mathbf{y}}_i - \frac{1}{2} \overline{\mathbf{y}}_i' \mathbf{S}_{\text{pl}}^{-1} \overline{\mathbf{y}}_i$$

For the quadratic classification to work, the group sample sizes, n_i, must all be larger than p for each $\mathbf{S}_i$ to have an inverse. For linear classification, we just need $n > p$ for the inverse to exist.

Misclassification rates

How well a classifcation scheme does is measured by misclassification rates. If the data is used to create a classifier, and then the same data is used to test the classifier, this is called **resubstitution**. The resulting error rate is the number of misclassified observations, and is called the **apparent error rate**. This is an optimistic assessment of the error rate because the classifier has been "trained" on the data being tested, and might not perform as well on data that wasn't part of its training set. Cross-validation methods can also be used to get less biased error rates.

Some of the language used to describe classifiers comes from Artificial Intelligence and Machine Learning, where we think of a function doing the classification as having learned from experience (the data used to generate the classifier). If you've learned something well, you can apply it to new situations, not just the examples already presented, and this is often called **generalization**.

Misclassification

For now, we consider all errors to be equally important. In some cases, you might care about some errors more than others. For classifying tumors as benign or malignant, there are two types of errors: false positives, in which a benign tumor is classified as malignant; and (2) false negatives, in which a malignant tumor is classified as benign. The fasle negatives in this case might be more serious than false negatives if it means that a malignant tumor is not appropriately treated.

For now, we consider classification with two groups and consider both kinds of errors equally.

Misclassification

Let n_1 be the number of observations in group G_1, and let n_2 be the number of observations in group G_2. Then let

- n_{11} be the number of observations from group 1 classified as group 1
- n_{12} be the number of observations from group 1 classified as group 2
- n_{21} be the number of observations from group 2 classified as group 1
- n_{22} be the number of observations from group 2 classified as group 2

The total number of misclassifications is $n_{12} + n_{21}$, and the number of correct classifications is $n_{11} + n_{22}$. The apparent error rate is

$$\frac{n_{12} + n_{21}}{n_1 + n_2}$$

Misclassification

The results can be summarized in a **classification table**, which is also someimtes called a **confusion matrix**:

Actual Group	Number of Observations	Predicted Group	
		1	2
1	n_1	n_{11}	n_{12}
2	n_2	n_{21}	n_{22}

Misclassification

For the chile data, we had two chiles from Alcalde classified as being from Cochiti and two from Cochiti calssified as being from Alcalde. There were 11 chiles in each group, so the misclassification rate is $(2+2)/(11+11) = 2/11 = 18\%$.

Now we'll try the quadratic classification on the chile data to see how that performs

Misclassification

```
> D2.1 <- 1:22
> for(i in 1:22) {
+ D2.1[i] <- as.matrix(y[i,]-ybar1) %*% S1inv %*% t(as.matrix(y[i,]-
+ }
> D2.2 <- 1:22
> for(i in 1:22) {
+ D2.2[i] <- as.matrix(y[i,]-ybar2) %*% S2inv %*% t(as.matrix(y[i,]-
+ }
> D2.1-D2.2
 [1]  -2.2287341  -9.6294436  -9.9478823 -22.2589087 -12.6802911  -1
 [7]  -6.1732641  -1.8530645  -1.6945790  -0.7523556  -4.0103896   0
[13]  -1.4539157   4.4057631   2.1913674  -2.2287341   4.5569114   8
[19]   4.4057631  -3.4261807   3.2229911   2.9008475
> (D2.1-D2.2 < 0)
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 TRUE  TRUE  TRUE FALSE
[13]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
TRUE FALSE FALSE
```

Miclassification

For the quadratic function, we classify the chile as being from group 1 (Alcalde) if $D_1^2 < D_2^2 \Rightarrow D_1^2 - D_2^2 < 0$ for each observation. The quadratic rule correctly classifies all the Alcalde chiles but incorrectly classifies three of the Cochiti chiles. The quadratic function therefore had an apparent error rate of $(0 + 3)/(11 + 11) = 3/22 = 13.6\%$. This is one better than the linear rule, but with such a small sample, it is difficult to say that this is "significantly" better.

Misclassification

To summarize the results in a table, we can write

		Predicted group	
Actual group	number of observations	Alcalde	Cochiti
Alcalde	11	11	0
Cochiti	11	3	8

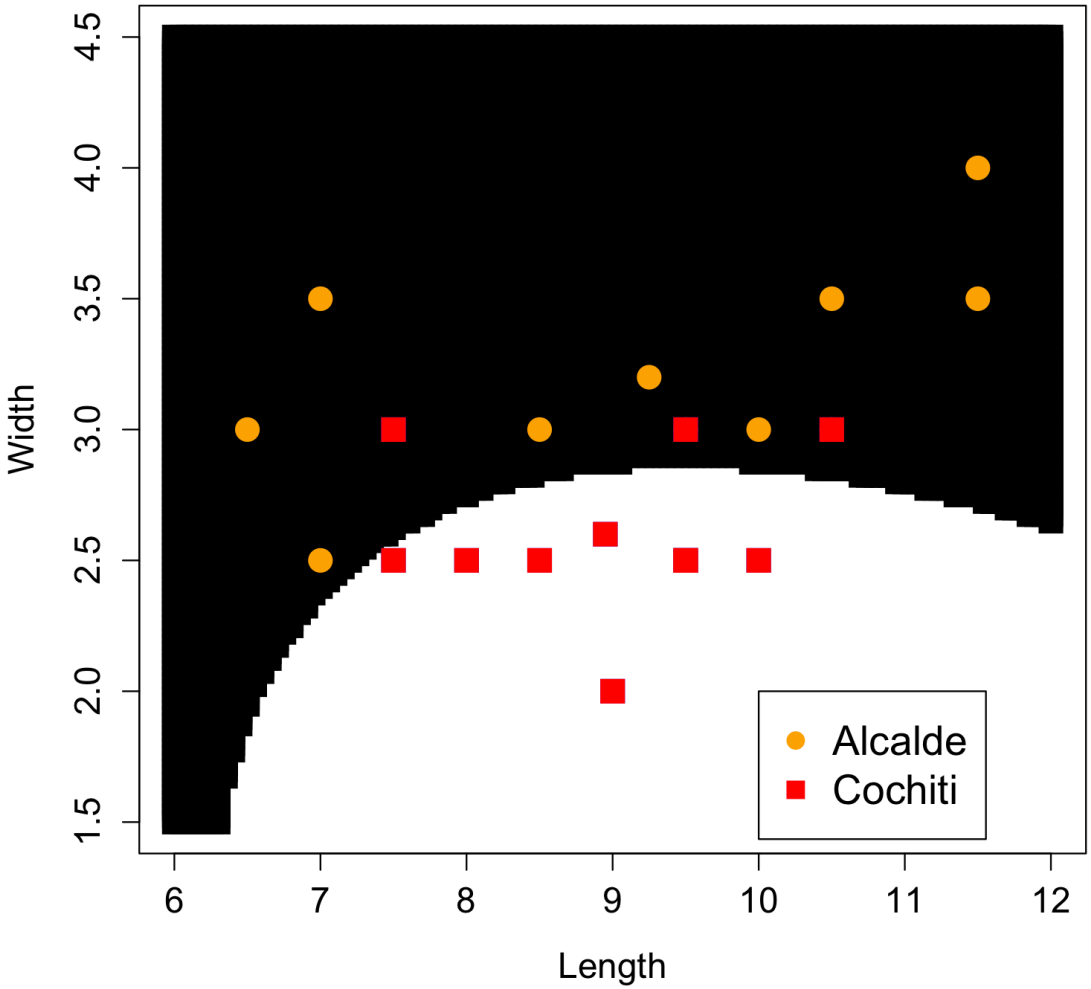
Plotting the quadratic rule

```
#create ranges for length and width of chiles
> y1 <- seq(6,12,.05)
> length(y1)
[1] 121
> y2 <- (y1-6)/6 *3 + 1.5
> mymatrix <- matrix(nrow=121,ncol=121)
> for(i in 1:121) {
+ for(j in 1:121) {
+ D2.1 <-  as.matrix(t(c(y1[i],y2[j])-ybar1)) %*% S1inv %*%
as.matrix(c(y1[i],y2[j])-ybar1)
+ D2.2 <-  as.matrix(t(c(y1[i],y2[j])-ybar2)) %*% S2inv %*%
as.matrix(c(y1[i],y2[j])-ybar2)
+ if(D2.1 < D2.2) mymatrix[i,j] = 1
+ else mymatrix[i,j] = 0
+ }
+ }
```

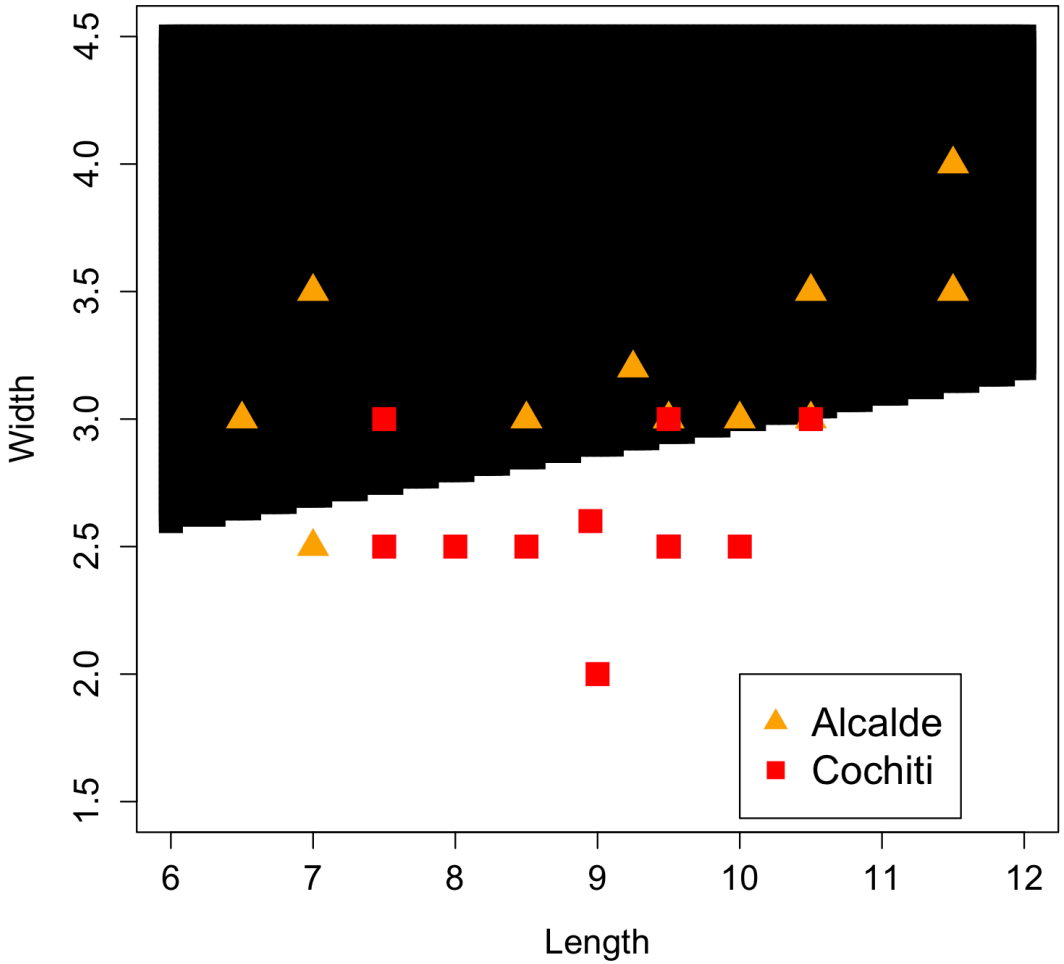
Plotting the quadratic rule

```
> plot(c(6,12),c(1.5,4.5),type="n",xlab="Length",ylab="Width",cex.la
> for(i in 1:121) {
+ for(j in 1:121) {
+ points(y1[i],y2[j],pch=15,cex=2,col=paste(mymatrix[i,j]))
+ }}
> points(x[1:11,2],x[1:11,3],pch=16,col="orange",cex=2)
> points(x[34:44,2],x[34:44,3],pch=15,col="red",cex=2)
> legend(10,2,legend=c("Alcalde","Cochiti"),col=c("orange","red"),ce
```

Plotting the quadratic rule



Plotting the linear rule



Getting less biased error rates

The resubstitution methods leads to overly optimistic apparent error rates. To get a better idea of how a classification rule generalizes to new data, an idea is to randomly split up your data into two portions, a **training** and a **validation**. There are different ways of doing this. By reducing the sample size in the training set, the classification rule is made from less data, so might not be as good as a classifier based on all the data, but it still might give a better sense of how well the classifier would work on new data, and could be used, for example, to help choose between a linear and quadratic classifer, or to help choose between Euclidean versus other distance functions.

In splitting up the data, you are evaluating the performance of a classifier that is not quite the classifier that would want to use a final product — eventually you would want to use the classifier built on the entire data set. Another issue is that reducing the sample size might be unreasonable for small samples. Particularly for the quadratic classifier, you could end up with $n_i < p$

Partitioning samples

A simple example of partitioning the data is to randomly pick 50% of the data to be the training set and the remaining 50% to be the validation set. The results will depend on the random subset of the data used, so you could repeat the procedure with different random subsets and take the average misclassification rates. Instead of 50%, you could also use a number such as 80% or 90%. This will make the classification rule more similar between random samples (which would tend to reduce the variability in the number of misclassifications). On the other hand, the number of misclassifications is binomial where the parameter n is size of the validation set, so this would tend to increase the variability in the misclassification rates.

Partitioning samples

Another approach is to just leave out one observation at a time. In this case, you construct N classification rules, each based on $N-1$ observations, and classify the one observation left out on the basis of the previous $N-1$ observations. This way, the one observation mbeing classified is independent of the classification rule. You then calculate the misclassification rate based on the number of misclassified observations out of the N observations. This method is called the **holdout**, **leave-one-out**, or **cross-validation**. Note that it is a similar idea to studentized residuals from regression.

Partitioning samples

A generalization of this approach are to leave out more than one observation at a time (for example leave out two), but this would require leaving out $\binom{N}{2}$ observations, but this is computationally intensive for large N.

Another generalization is k-fold cross-validation, in which you partition the data in k sets (e.g., $k = 10$) and train the classifer using $N(1 - 1/k)$ observations to classify the remaining N/k observations. This is repeated k times, one for each partition, and the misclassification rates can be averaged. $k = 2$ is the original idea of splitting the data into two halves, and $k = N$ is the leave-one-out-method. $k = 10$ is the most common choice for k-fold cross-validation.

Creating random subsets in R

The `sample()` command in R makes it pretty easy to create random subsets. If you have N total observations, then you can shuffle a vector with the values $\{1, \ldots, N\}$. Here are some examples:

```
> n1 <- round(n/2)
> N <- 1:length(x[,1]) # assume data is in x
> training <- sample(N,n1,replace=F)
> x.training <- x[training,]
> x.validation <- x[-training,] #indices
from 1,...,N NOT in the training set
```

Creating random subsets in R

```
> k <- 10
> x.temp <- x[sample(1:n,replace=F),]
> for(i in 1:10) {
> first.obs <- floor((i-1)*N/k) + 1
> last.obs <- floor(i*N/k)
> x.temp.training <- x[first.obs:last.obs,]
```

To understand the loop, suppose we have 325 observations and want to do k-fold cross-validation. When `i=1`, `first.obs` is set to $\lfloor 0 * 325/10 \rfloor + 1 = 1$ and `last.obs` is set to $\lfloor 325/10 \rfloor = 32$. Thus, the first temporary data set has observations 1 through 32 of the shuffled data. When `i=2`, `first.obs` is set to $\lfloor 325/10 \rfloor + 1 = 33$ and `last.obs` is set to $2(325)/10 = 65$. Thus the second partition has observations 33 through 65 of the shuffled data set. Note that partition 2 has one more observation than partition 1 because it isn't possible for all partitions to have exaclty the same number of observations.

Variable selection

You can also use misclassification rates as a basis for variable selection. If a small number of variables has similar misclassification rates as a larger number of variables, then the larger set of variables isn't helping predictive ability, so we might decide to go with the smaller set of variables. Different subsets of variables can be tried with cross-validation applied to each subset to help determine a reasonable subset of variables when the goal is to classify using a small number of variables.

Fisher Iris data

A famous data set used to illustrate classification is Fisher's Iris data from 1936. There are three species of iris with 50 observations each. The species are:

1. Iris setosa
2. Iris versicolour
3. Iris virginica

The variables are (in cm)

1. sepal length
2. sepal width
3. petal length
4. petal width

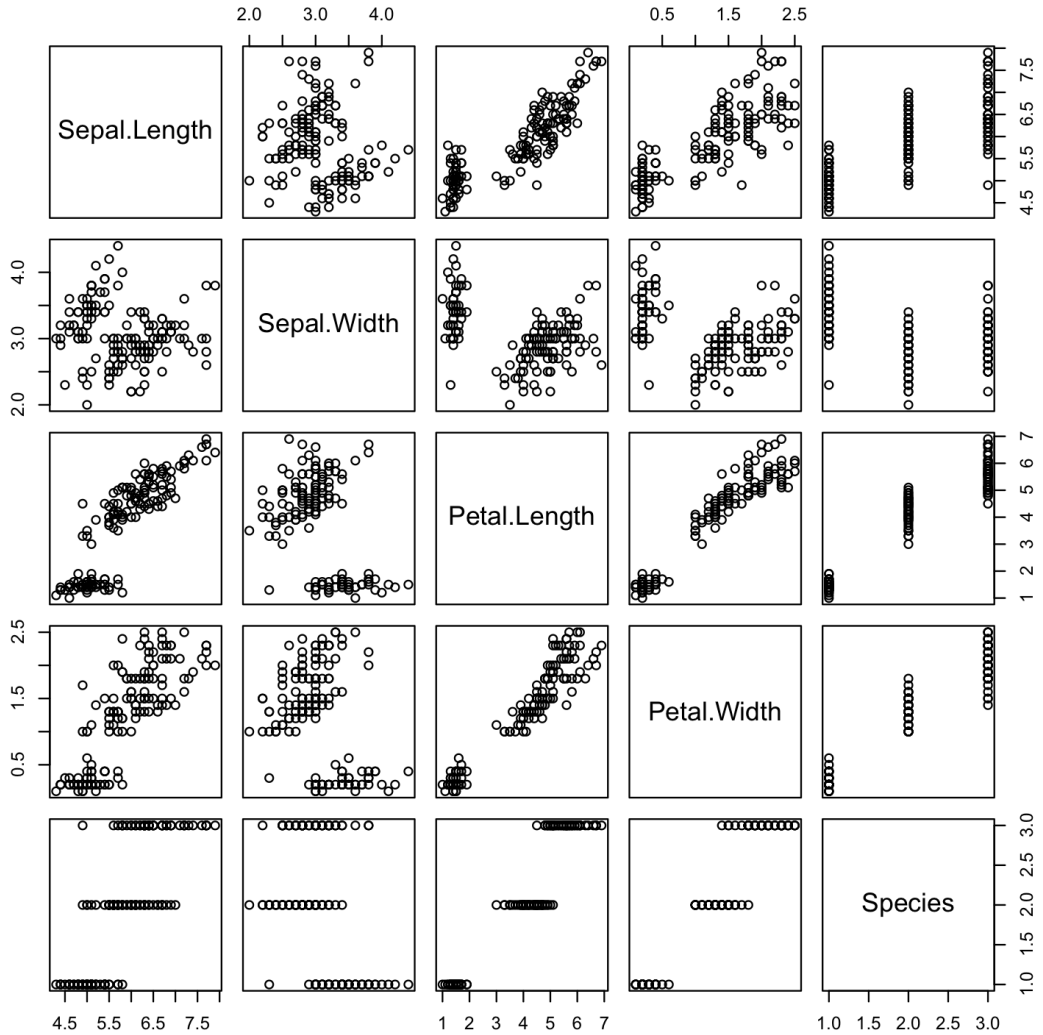
Fisher Iris data

Fisher Iris data

The data is famous enough to be built into R

```
> data(iris)
> iris
> head(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Fisher Iris data



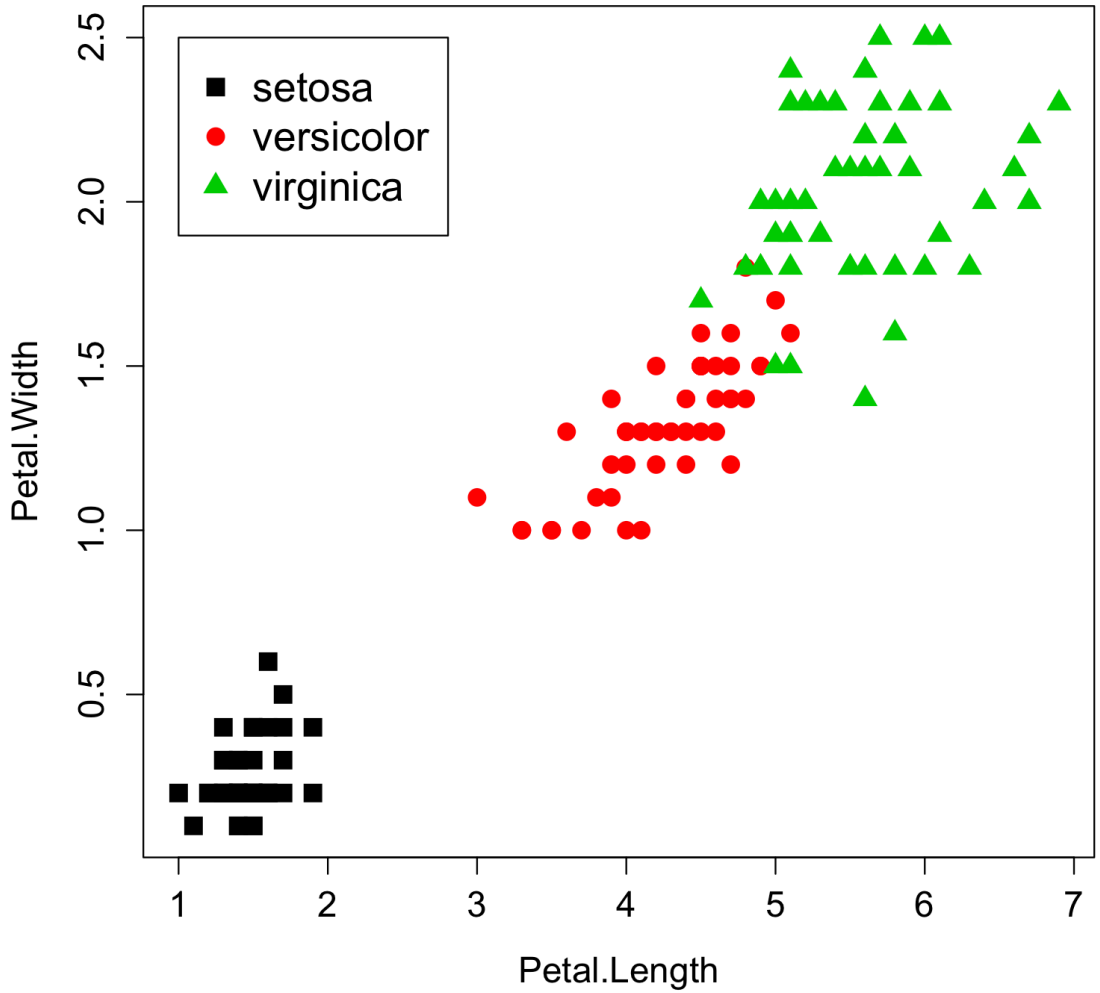
Fisher Iris data

Species 1 (setosa) seems clearly separated from species 2 and 3 on petal length and width, but not on sepal length and width. Species 2 and 3 are less clearly separated, but also seem to have more separation for petal attributes than the sepals.

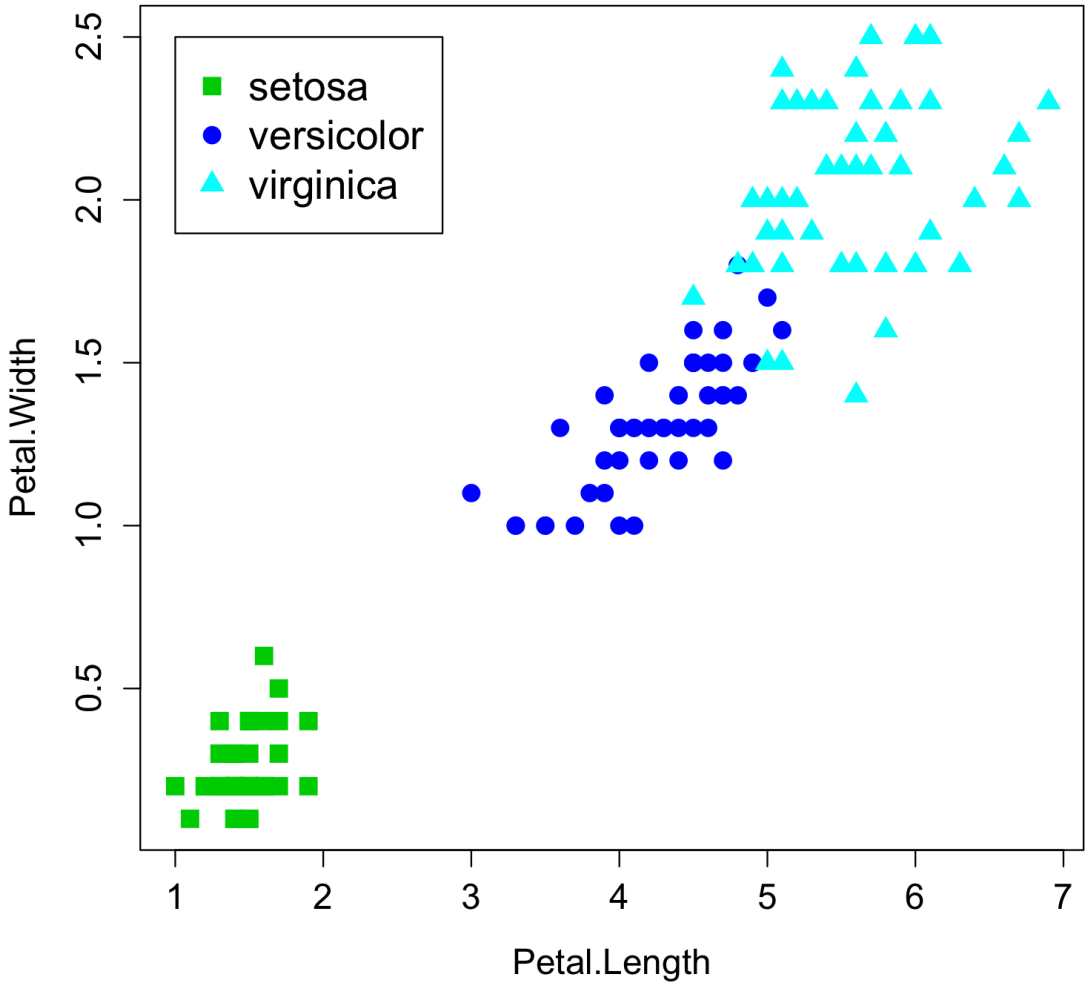
```
> plot(Petal.Length,Petal.Width,col=species,pch=14+species,
cex.lab=1.3,cex.axis=1.3,cex=1.5)
> legend(1,2.5,legend=c("setosa","versicolor","virginica"),
pch=c(15,16,17),col=c(1,2,3),cex=1.5)
> plot(Petal.Length,Petal.Width,col=species+2,pch=14+
species,cex.lab=1.3,cex.axis=1.3,cex=1.5)
> legend(1,2.5,legend=c("setosa","versicolor","virginica"),
pch=c(15,16,17),col=c(1,2,3)+2,cex=1.5)
```

can also use `col=mycolor` in plot where `mycol=c(rep("orange",50),rep("red",50),rep("brown",50))`, for example to customize colors.

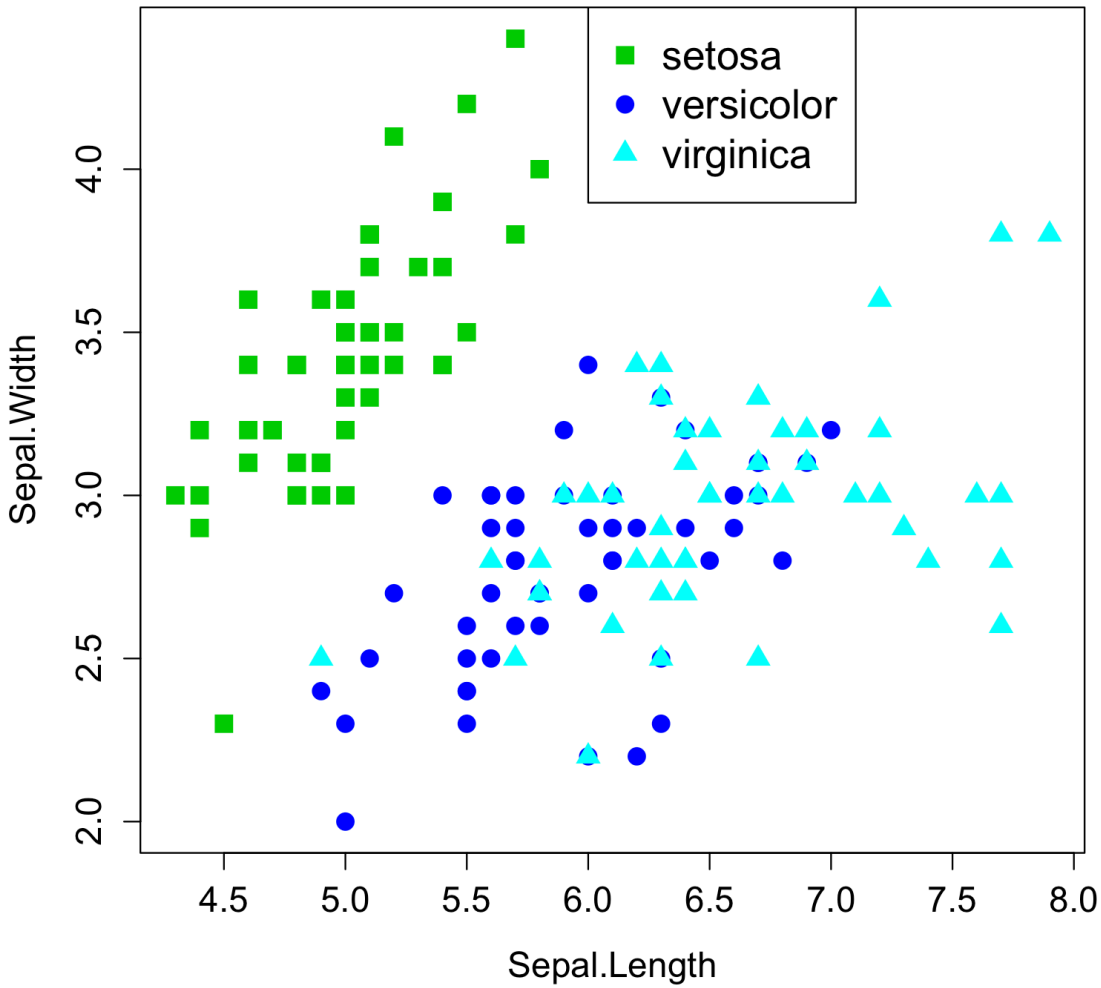
Fisher Iris data



Fisher Iris data: easier colors for me



Fisher Iris data: less separation on sepal variables



Fisher Iris data

```
> S1 <- cov(iris[1:50,1:4])
> S2 <- cov(iris[51:100,1:4])
> S3 <- cov(iris[101:150,1:4])
> Sp <- (S1+S2+S3)/3
> S <- cov(iris[,1:4])
```

Is the covariance matrix S the same as the pooled covariance matrix when all sample sizes are equal?

Fisher Iris data

No, the pooled covariance matrix tends to have smaller variances than the variance matrix on all the data when there are differences between groups.

```
> D2.1 <- 1:150
> D2.2 <- 1:150
> D2.3 <- 1:150
> for(i in 1:150) {
+ D2.1[i] <-  as.matrix(iris[i,1:4]-ybar1) %*%
Sinv %*% t(as.matrix(iris[i,1:4]-ybar1))
+ D2.2[i] <-  as.matrix(iris[i,1:4]-ybar2) %*%
Sinv %*% t(as.matrix(iris[i,1:4]-ybar2))
+ D2.3[i] <-  as.matrix(iris[i,1:4]-ybar3) %*%
Sinv %*% t(as.matrix(iris[i,1:4]-ybar3))
+ }
```

```
> classifier <- 1:150
> for(i in 1:150) {
+ if(D2.1[i] < min(D2.2[i],D2.3[i])) classifier[i] <- 1
+ else if(D2.2[i] < min(D2.1[i],D2.3[i])) classifier[i] <- 2
+ else if(D2.3[i] < min(D2.1[i],D2.2[i])) classifier[i] <- 3
+ }
> classifier
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3
[149] 3 3
> truth <- c(rep(1,50),rep(2,50),rep(3,50))
>  which(as.numeric(classifier==truth)==0)
[1]  71  84 134
```

So observations 71, 84, and 134 were misclassified, with two versicolor being classified as virginica and one virginica being misclassified as versicolor.

For a quadratic classifier, we use

```
> for(i in 1:150) {
+ D2.1[i] <-  as.matrix(iris[i,1:4]-ybar1) %*% solve(S1)
 %*% t(as.matrix(iris[i,1:4]-ybar1))}
> for(i in 1:150) {
+ D2.2[i] <-  as.matrix(iris[i,1:4]-ybar2) %*% solve(S2)
%*% t(as.matrix(iris[i,1:4]-ybar2))}
> for(i in 1:150) {
+ D2.3[i] <-  as.matrix(iris[i,1:4]-ybar3) %*% solve(S3)
%*% t(as.matrix(iris[i,1:4]-ybar3))}
> for(i in 1:150) {
+ if(D2.1[i] < min(D2.2[i],D2.3[i])) classifier[i] <- 1
+ if(D2.2[i] < min(D2.1[i],D2.3[i])) classifier[i] <- 2
+ if(D2.3[i] < min(D2.1[i],D2.2[i])) classifier[i] <- 3
+ }
>  which(as.numeric(classifier==truth)==0)
[1] 71 73 84
```

So again 71 and 84 are misclassified, but now 73 is misclassified instead of 134.

Fisher Iris data

We can't really plot the classification rule in the nice way as was done for two variables since we now have 4 variables. The classification decision depends on the value of all four variables. We could write a function that gave the decision as a function of the four variables, and we could make a 2D plot of the decision rule by fixing two of the other variables, but there would a separate plot for each combination of the fixed variables.

Fisher Iris data

We might want to look at the individual flowers that were misclassified.

```
> iris[c(71,73,84,134),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
71           5.9         3.2          4.8         1.8 versicolor
73           6.3         2.5          4.9         1.5 versicolor
84           6.0         2.7          5.1         1.6 versicolor
134          6.3         2.8          5.1         1.5  virginica
> y2bar
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
> ybar3
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       6.588        2.974        5.552        2.026
```

So generally virginica has larger values on all variables, on average, and the misclassified versicolor irises had larger than average values for versicolor, and the misclassified virginica had smaller than average values for virginica.