PS1Q5 (PCA) Show that $\text{tr}(\tilde{V}_q^T X^T X \tilde{V}_q)$ is maximized (up to permutation) over matrices $\tilde{V}_q = (\tilde{v}_1, \ldots, \tilde{v}_q) \in \mathbb{R}^{p \times q}$ with q orthonormal columns (i.e. $\tilde{V}_q^T \tilde{V}_q = 1$) by $\tilde{v}_i = v_i$, where v_i, $i = 1, \ldots, q$ are the q eigenvectors of $X^T X v_i = \lambda_i v_i,\ i = 1, 2...q$ corresponding to the first (i.e. the largest) q eigenvalues of $X^T X$.

Answer: Let V be the orthogonal matrix consisting of eigenvectors v_i. Since $\text{col}(V)$ is a basis for the column space of $\tilde{V}_q$, there is a $p \times q$ matrix $A = (a_{ij})$ so that $\tilde{V}_q = VA$ and since $\tilde{V}_q$ and V are both orthogonal, so is A. Now $X^T X = V D^2 V^T$, with $D^2 = \text{diag}(\lambda_1, ..., \lambda_p)$ the diagonal matrix of eigenvalues of $X^T X$.

$$\begin{aligned}
\text{tr}(\tilde{V}_q^T X^T X \tilde{V}_q) &= \text{tr}(A^T V^T V D^2 V^T V A) && (1)\\
&= \text{tr}(A^T D^2 A) && (2)\\
&= \sum_{j=1}^{q} \sum_{i=1}^{p} \lambda_i a_{ij}^2 && (3)\\
&= \sum_{i=1}^{p} \lambda_i b_i && (4)
\end{aligned}$$

where $b_i = \sum_{j=1}^{q} a_{ij}^2$. Now $0 \leq b_i \leq 1$ (since AA^T is a projection into $\text{col}(A)$, and the length of the projection of a unit vector is smaller than one) and $\sum_i b_i = q$ (since $\sum_i b_i = \sum_{ij} a_{ij}^2 = \text{tr}(A^T A) = \text{tr}(I_q) = q$). The maximum of $\sum_{i=1}^{p} \lambda_i b_i$ over $b_i, i = 1, ..., p$ is achieved by the choice $[b_i = 1, i = 1, ..., q,\ b_j = 0, j = q+1, ..., p]$, since $\lambda_i \geq \lambda_{i+1}$, so any other choice can be beaten by moving mass from $b_j, j > q$ into $b_i, i \leq q$. Hence the maximum of $\text{tr}(\tilde{V}_q^T X^T X \tilde{V}_q)$ is $\lambda_1 + \lambda_2 + ... + \lambda_q$ which is achieved by the choice $\tilde{v}_i = v_i,\ i = 1...q$.

Note that maximising over $\tilde{V}_q$ is clearly equivalent to maximizing over A, but we didn't show that this was again equivalent to maximizing over $b_i, i = 1, ..., p$ subject to $0 \leq b_i \leq 1$, $\sum_i b_i = q$; there might be no A to achieve a given b. However since the solution we obtained was achieved within the space of matrices $\tilde{V}_q$, we must have the correct upper bound.

PS2Q4 (EDA) Obtain `http://www.stats.ox.ac.uk/\%7Eteh/teaching/datamining/cognate.txt` and load it using something like `X <- read.table("cognate.txt")`. It contains an 87×2665 matrix of observations on each of 87 Indo-European languages where the presence (1) or absence (0) of 2665 homologous traits has been recorded.

Historical linguists have grouped these languages into clades. Most large scale groupings are contested, but something like

$$\{Indic, Iranian\}$$

$$\{Balto - Slav, (Germanic, Italic, Celtic)\}$$

is not too controversial. The position of the Armenian, Greek, Albanian, Tocharian and Hittite groups is in doubt (though not within the second of the above super-clade).

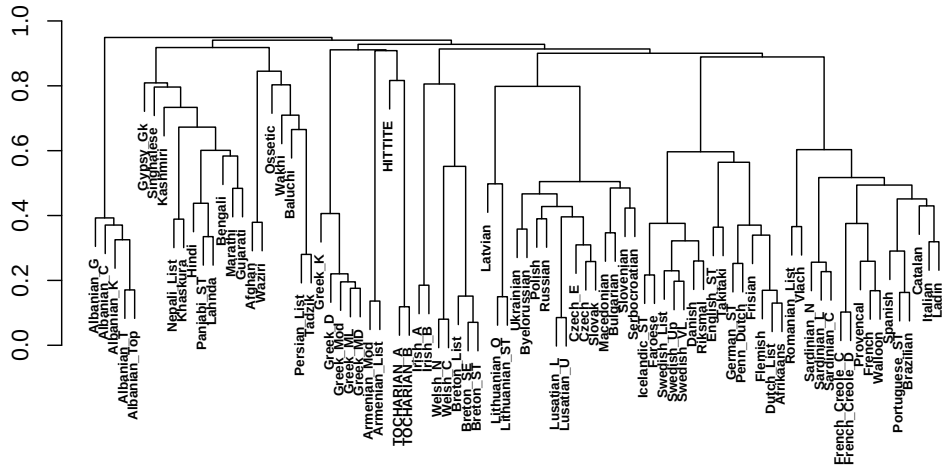
We would like to cluster the languages into groups on the basis of these data. It is also of interest to represent the languages in a planar map in order to visualise similarities between languages.

(a) These data are categorical. The **S**imple **M**atching **C**oefficient for two data vectors is the proportion of variables which are unequal. The Jaccard coefficient for two language data vectors is the proportion of variables with at least one present which are unequal (so 1100 and 1010 have SMC 2/4 and JC 2/3). Which dissimilarity measure is appropriate for these data?

Answer: Probably Jaccard. These are trait data. If two objects both lack many irrelevant traits, that should not make them more similar. So dist(1100,1000) and d(1111,1100) should be the same, a vote for Jaccard. In this data 1's make up only about 5% of the data values, so shared absence of traits is much more common than shared presence, again pointing to Jaccard.

(b) Compute an agglomerative clustering of the data, and plot a dendrogram with language labels on the leaves. You will need to specify a distance measure between clusters. Include your R code for generating the dendrogram.

Answer: Using Jaccard for the distance measure between data-vectors, and "`average`" between clusters. In R, "`average`" is the average distance between points in different clusters. This measure generates trees with no inversions, and reproduces most of the prior clade structure. "`complete`" is similar. "`single`" produces strange groupings. Replacing Jaccard with SMC ("`manhattan`" in `dist()`) seems to do little damage: in fact "`manhattan`" with "`complete`" predicts all prior clade structure.




```
rm(list = ls(all = TRUE))
X <- read.table("cognate.R")

## calculate Jaccard distances
D <- dist(X,method="binary")

## form clusters by agglomerative cluster using "average"
## or "complete". The choice "single" is not robust.
hi <- hclust(D, method="average")
plot(hi,labels=row.names(X),cex=0.6,font=2,ann=FALSE)
```

(c) Compute K-means clustering with 10 groups. Include your R code.

Answer:

```
## make 10 clusters with kmeans
K <- 10
km <- kmeans(X, K, nstart=100, iter.max=100)
```

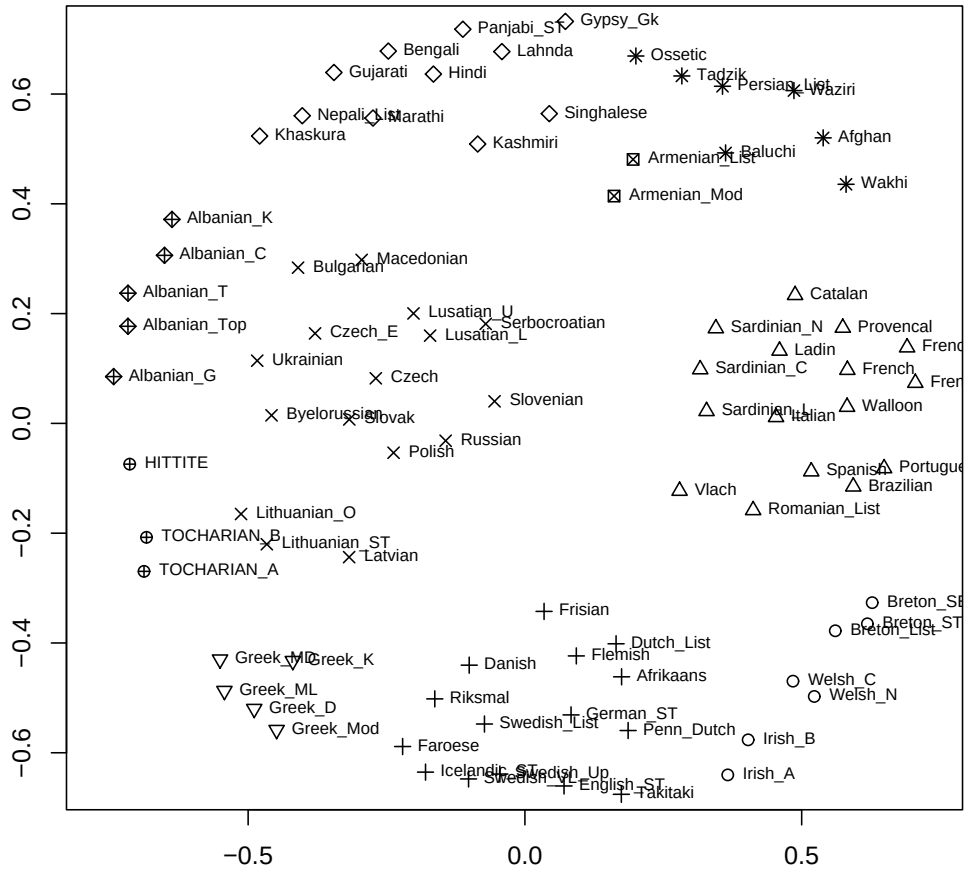
Showing the clustering result

```
> str(km)
List of 4
 $ cluster : Named int [1:87] 7 7 7 7 7 7 7 4 4 4 ...
  ..- attr(*, "names")= chr [1:87] "Irish_A" "Irish_B" "Welsh_N" ...
 $ centers : num [1:10, 1:2665] 0 0 0 0 0 0 0 1 1 0 ...
```

```
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:10] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2665] "V1" "V2" "V3" "V4" ...
 $ withinss: num [1:10] 153 336 923 933 838 ...
 $ size    : int [1:10] 5 5 15 16 11 5 7 13 3 7
 - attr(*, "class")= chr "kmeans"

> km$cluster
       Irish_A        Irish_B        Welsh_N        Welsh_C ...
             7              7              7              7
     Breton_SE      Breton_ST  Romanian_List          Vlach
             7              7              4              4
         Ladin      Provencal         French       Walloon
             4              4              4              4
French_Creole_D    Sardinian_N    Sardinian_L    Sardinian_C
             4              4              4              4
  Portuguese_ST      Brazilian        Catalan      German_ST
             4              4              4              3
     Dutch_List      Afrikaans        Flemish        Frisian
             3              3              3              3
     Swedish_VL   Swedish_List         Danish        Riksmal
             3              3              3              3
        Faroese      English_ST       Takitaki   Lithuanian_O
             3              3              3              9
        Latvian      Slovenian     Lusatian_L     Lusatian_U
             9              8              8              8
         Slovak        Czech_E      Ukrainian   Byelorussian
             8              8              8              8
        Russian      Macedonian       Bulgarian  Serbocroatian
             8              8              8              8
     Singhalese       Kashmiri        Marathi        Gujarati
             5              5              5              5
         Lahnda          Hindi        Bengali    Nepali_List
             5              5              5              5
       Greek_ML       Greek_MD       Greek_Mod       Greek_D
             6              6              6              6
   Armenian_Mod   Armenian_List        Ossetic         Afghan
             2              2             10             10
   Persian_List         Tadzik        Baluchi          Wakhi
            10             10             10             10
   Albanian_Top      Albanian_G     Albanian_K     Albanian_C   ...
             1              1              1              1
```

(d) Using MDS (the Sammon map may be best), represent the languages in a 2D plot. Plot the clusters obtained in part **??** using different symbols, or colors and superpose the language name. Can you see any geographical grouping in the layout?

Answer: MDS gives a nice visualization if we use `sammon`. This MDS scheme minimizes a stress function with terms like $\frac{(d_{i(j)}-\tilde{d}_{i(j)})^2}{d_{i(j)}}$ as opposed to "classical" MDS (i.e. `cmdscale`) which minimizes $(d^2_{i(j)} - \tilde{d}^2_{i(j)})^2$. Sammon thereby puts more weight on reproducing the separation of points which are close by forcing them apart. Projection by MDS(Jaccard/sammon) with cluster discovery by k-means (Jaccard): There is an obvious east to west (top-left to bottom-right) separation of languages in the MDS and the clusters in the MDS grouping agree with the clusters discovered by agglomerative clustering and k-means. The two clustering methods group languages slightly differently with k-means splitting the Germanic languages.



```
## (alternative/MDS) make a field to display the clusters
## use MDS - sammon does this nicely
di.sam <- sammon(D,magic=0.20000002,niter=1000,tol=1e-8)
eqscplot(di.sam$points,pch=km$cluster,col=km$cluster)
text(di.sam$points,labels=row.names(X),pos=4,col=km$cluster)
```

PS5Q2 (Bayesian classification) Consider some training data $D = \{(x_i, z_i), y_i\}_{i=1}^n$ where $(x_i, z_i) \in \mathbb{R}^p \times \mathbb{R}$ is the vector of inputs and $y_i \in \{0, 1\}$ the response. We adopt the following regression model for class k

$$Z = \beta_k^{\mathrm{T}} X + \varepsilon$$

where $\varepsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_k^2)$ if $Y = k$. Hence we have for the class conditional density $f_k(z|x) = \mathcal{N}\left(z; \beta_k^{\mathrm{T}} x, \sigma_k^2\right)$ so that the unconditional density of Z follows a so-called mixture of regressions model. Note that this model differs conceptually from the examples discussed in lectures as we do not model X. We adopt the notation $P(Y = k) = \pi_k$ and denote $\theta = (\pi_1, \beta_0, \beta_1, \sigma_0^2, \sigma_1^2)$ the set of unknown parameters.

(a) Give an expression of the estimate $\widehat{\theta}$ of θ maximizing the conditional log-likelihood

$$l(\theta) = \sum_{i=1}^n \log\ p(y_i, z_i|\, x_i, \theta).$$

What happens when $n < p$?

Answer: We have simply

$$\begin{aligned}
\widehat{\pi}_k &= \frac{n_k}{n} \\
\widehat{\sigma}_k^2 &= \frac{1}{n_k} \sum_{i=1}^n \mathbb{I}(y_i = k) \left(z_i - \widehat{\beta}_k^{\mathrm{T}} x_i\right)^2 \\
\widehat{\beta}_k &= \left[\sum_{i=1}^n \mathbb{I}(y_i = k)\, x_i x_i^{\mathrm{T}}\right]^{-1} \left[\sum_{i=1}^n \mathbb{I}(y_i = k)\, z_i x_i\right].
\end{aligned}$$

When $n < p$, then $\widehat{\beta}_k$ is not defined as the $p \times p$ matrix $\left[\sum_{i=1}^n \mathbb{I}(y_i = k)\, x_i x_i^{\mathrm{T}}\right]$ is at most of rank n.

(b) Consider a Bayesian approach with $\pi_1 \sim$Beta(a, b) and

$$p\left(\sigma_0^2, \beta_0, \sigma_1^2, \beta_1\right) = p\left(\sigma_0^2, \beta_0\right) p\left(\sigma_1^2, \beta_1\right)$$

where $p(\sigma_k^2, \beta_k)$ satisfies a normal inverse-Gamma distribution; e.g.

$$\begin{aligned}
p\left(\sigma_k^2, \beta_k\right) &= p\left(\sigma_k^2\right) p\left(\beta_k|\, \sigma_k^2\right) \\
&= \mathcal{IG}\left(\sigma_k^2; \frac{\nu}{2}, \frac{\kappa}{2}\right) \mathcal{N}\left(\beta_k; 0, \sigma_k^2 \Sigma\right)
\end{aligned}$$

with some hyperparameters $(\nu, \kappa, \delta\Sigma)$ such that $\nu, \kappa > 0$ and Σ is a positive definite matrix. $\mathcal{IG}\left(\sigma^2; \frac{\nu}{2}, \frac{\kappa}{2}\right)$ denotes the inverse-Gamma density given by

$$\mathcal{IG}\left(\sigma^2; \frac{\nu}{2}, \frac{\kappa}{2}\right) = \frac{\left(\frac{\kappa}{2}\right)^{\frac{\nu}{2}}}{\Gamma\left(\frac{\nu}{2}\right)} \left(\sigma^2\right)^{-\frac{\nu}{2}-1} \exp\left(-\frac{\kappa}{2\sigma^2}\right).$$

The posterior distribution $p\left(\theta \middle| D\right)$ satisfies

$$p\left(\theta \middle| D\right) = p\left(\pi_1 \middle| D\right) p\left(\sigma_0^2, \beta_0 \middle| D\right) p\left(\sigma_1^2, \beta_1 \middle| D\right).$$

Show that $p\left(\pi_1 \middle| D\right)$ is a Beta distribution and $p\left(\sigma_k^2, \beta_k \middle| D\right)$ a normal inverse-Gamma distribution.

Answer: We have $p\left(\pi_1 \middle| D\right) = Beta\left(\pi_1; a + n_1, b + n - n_1\right)$ and

$$p\left(\beta_k, \sigma_k^2 \middle| D\right) = \mathcal{IG}\left(\sigma_k^2; \frac{\nu_k}{2}, \frac{\kappa_k}{2}\right) \mathcal{N}\left(\beta_k; \mu_k, \sigma_k^2 \Sigma_k\right)$$

where

$$\begin{aligned} p\left(\beta_k, \sigma_k^2 \middle| D\right) \quad \propto \quad & \left(\sigma_k^2\right)^{-\frac{\nu}{2}-1} \exp\left(-\frac{\kappa}{2\sigma_k^2}\right) \frac{\exp\left(-\frac{1}{2\sigma_k^2}\beta_k^{\mathrm{T}}\Sigma^{-1}\beta_k\right)}{\sigma_k^{p/2}\left|\Sigma_k\right|^{1/2}} \\ & \times \frac{1}{\left(\sigma_k^2\right)^{n/2}} \exp\left(-\frac{\sum_{i=1}^n \mathbb{I}\left(y_i = k\right)\left(z_i - \beta_k^{\mathrm{T}} x_i\right)^2}{2\sigma_k^2}\right) \end{aligned}$$

but

$$\begin{aligned} & \frac{1}{2\sigma_k^2}\beta_k^{\mathrm{T}}\Sigma^{-1}\beta_k + \frac{\sum_{i=1}^n \mathbb{I}\left(y_i = k\right)\left(z_i - \beta_k^{\mathrm{T}} x_i\right)^2}{2\sigma_k^2} \\ = \quad & \frac{1}{2\sigma_k^2}\beta_k^{\mathrm{T}}\Sigma^{-1}\beta_k + \frac{\left(Z_k - X_k\beta_k\right)^{\mathrm{T}}\left(Z_k - X_k\beta_k\right)}{2\sigma_k^2} \\ = \quad & \frac{1}{2\sigma_k^2}\beta_k^{\mathrm{T}}\Sigma^{-1}\beta_k + \beta_k^{\mathrm{T}}\frac{X_k^{\mathrm{T}}X_k}{2\sigma_k^2}\beta_k - 2\beta_k^{\mathrm{T}}\frac{X_k^{\mathrm{T}}Z_k}{2\sigma_k^2} + \frac{Z_k^{\mathrm{T}}Z_k}{2\sigma_k^2} \\ = \quad & \frac{1}{2\sigma_k^2}\beta_k^{\mathrm{T}}\Sigma_k^{-1}\beta_k - 2\beta_k^{\mathrm{T}}\frac{\Sigma_k^{-1}\mu_k}{2\sigma_k^2} + \frac{Z_k^{\mathrm{T}}Z_k}{2\sigma_k^2} \\ = \quad & \frac{1}{2\sigma_k^2}\left(\beta_k - \mu_k\right)^{\mathrm{T}}\Sigma_k^{-1}\left(\beta_k - \mu_k\right) + \frac{Z_k^{\mathrm{T}}Z_k}{2\sigma_k^2} - \frac{\mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k}{2\sigma_k^2} \end{aligned}$$

with

$$\begin{aligned} \Sigma_k \quad &= \quad \left(\Sigma^{-1} + \sum_{i=1}^n \mathbb{I}\left(y_i = k\right) x_i x_i^{\mathrm{T}}\right)^{-1}, \\ \mu_k \quad &= \quad \Sigma_k\left(\sum_{i=1}^n \mathbb{I}\left(y_i = k\right) z_i x_i\right) = \Sigma_k X_k^{\mathrm{T}} Z_k \end{aligned}$$

and as $\mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k = Z_k^{\mathrm{T}}X_k\Sigma_k X_k^{\mathrm{T}}Z_k$

$$\begin{aligned}\nu_k &= \nu + n,\\ \kappa_k &= \kappa + Z_k^{\mathrm{T}}\left(I_{n_k} - X_k\Sigma_k X_k^{\mathrm{T}}\right)Z_k.\end{aligned}$$

(c) Given a new test data (x, z), establish the expression of $p\left(z|\, D, x, y = k\right)$ and explain how you would use this expression to obtain a Bayesian classifier. What are the potential benefits of this approach over using $p\left(z|\,\widehat{\theta}, x, y = k\right)$?

Answer: We have

$$\begin{aligned}p\left(z|\, D, x, y = k\right) &= \int p\left(z|\,\theta, x, y = k\right) p\left(\theta|\, D\right) d\theta\\ &= \int \mathcal{N}\left(z; \beta_k^{\mathrm{T}}x, \sigma_k^2\right)\mathcal{IG}\left(\sigma_k^2; \frac{\nu_k}{2}, \frac{\kappa_k}{2}\right)\mathcal{N}\left(\beta_k; \mu_k, \sigma_k^2\Sigma_k\right) d\theta\end{aligned}$$

where

$$\begin{aligned}&\left(z - \beta_k^{\mathrm{T}}x\right)^2 + \left(\beta_k - \mu_k\right)^{\mathrm{T}}\Sigma_k^{-1}\left(\beta_k - \mu_k\right)\\ =\ &\beta_k^{\mathrm{T}}\left(\Sigma_k^{-1} + xx^{\mathrm{T}}\right)\beta_k - 2\beta_k\left(\Sigma_k^{-1}\mu_k + xz\right) + z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k\\ =\ &\beta_k^{\mathrm{T}}\Sigma_k^{-1}\left(x\right)\beta_k - 2\beta_k\Sigma_k^{-1}\left(x\right)\mu_k\left(x, z\right) + z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k\\ =\ &\left(\beta_k - \mu_k\left(x, z\right)\right)^{\mathrm{T}}\Sigma_k^{-1}\left(x\right)\left(\beta_k - \mu_k\left(x, z\right)\right) + z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k - \mu_k^{\mathrm{T}}\left(x, z\right)\Sigma_k^{-1}\left(x\right)\mu_k\left(x, z\right)\end{aligned}$$

so

$$\begin{aligned}&p\left(z, \sigma_k^2\middle|\, D, x, y = k\right)\\ =\ &\mathcal{IG}\left(\sigma_k^2; \frac{\nu_k}{2}, \frac{\kappa_k}{2}\right)\frac{\left|\Sigma_k\left(x\right)\right|^{1/2}}{\sqrt{2\pi}\sigma_k\left|\Sigma_k\right|^{1/2}}\\ &\times\exp\left(-\frac{1}{2\sigma_k^2}\left(z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k - \mu_k^{\mathrm{T}}\left(x, z\right)\Sigma_k^{-1}\left(x\right)\mu_k\left(x, z\right)\right)\right)\end{aligned}$$

where

$$\begin{aligned}\Sigma_k^{-1}\left(x\right) &= \Sigma_k^{-1} + xx^{\mathrm{T}},\\ \mu_k\left(x, z\right) &= \Sigma_k\left(x\right)\left(\Sigma_k^{-1}\mu_k + xz\right).\end{aligned}$$

So

$$\begin{aligned}
& \mathcal{IG}\left(\sigma_k^2; \frac{\nu_k}{2}, \frac{\kappa_k}{2}\right) \\
& \times \frac{\left|\Sigma_k(x)\right|^{1/2}}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}\left(z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k - \mu_k^{\mathrm{T}}(x,z)\,\Sigma_k^{-1}(x)\,\mu_k(x,z)\right)\right) \\
= & \frac{\left|\Sigma_k(x)\right|^{1/2}}{\sqrt{2\pi}} \frac{\left(\frac{\kappa}{2}\right)^{\frac{\nu}{2}}}{\Gamma\left(\frac{\nu}{2}\right)} \left(\sigma_k^2\right)^{-\frac{\nu_k+1}{2}-1} \\
& \times \exp\left(-\frac{1}{2\sigma_k^2}\left(z^2 + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k - \mu_k^{\mathrm{T}}(x,z)\,\Sigma_k^{-1}(x)\,\mu_k(x,z) + \kappa_k\right)\right) \\
= & \frac{\left|\Sigma_k(x)\right|^{1/2}}{\sqrt{2\pi}} \frac{\left(\frac{\kappa}{2}\right)^{\frac{\nu}{2}}}{\Gamma\left(\frac{\nu}{2}\right)} \left(\sigma_k^2\right)^{-\frac{\nu_k+1}{2}-1} \\
& \times \exp\left(-\frac{1}{2\sigma_k^2}\left(\kappa + z^2 + Z_k^{\mathrm{T}}Z_k - \mu_k^{\mathrm{T}}(x,z)\,\Sigma_k^{-1}(x)\,\mu_k(x,z)\right)\right)
\end{aligned}$$

but

$$\int \left(\sigma^2\right)^{-\frac{\nu+1}{2}-1} \exp\left(-\frac{\kappa}{2\sigma^2}\right) d\sigma^2 = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\left(\frac{\kappa}{2}\right)^{\frac{\nu+1}{2}}}$$

so

$$\begin{aligned}
p(z|\,D, x, y=k) &= \frac{\left|\Sigma_k(x)\right|^{1/2}}{\sqrt{2\pi}} \frac{\left(\frac{\kappa_k}{2}\right)^{\frac{\nu_k}{2}}}{\Gamma\left(\frac{\nu_k}{2}\right)} \Gamma\left(\frac{\nu_k+1}{2}\right) \\
& \times \left(\frac{\kappa + z^2 + Z_k^{\mathrm{T}}Z_k - \mu_k^{\mathrm{T}}(x,z)\,\Sigma_k^{-1}(x)\,\mu_k(x,z)}{2}\right)^{-\frac{\nu_k+1}{2}}
\end{aligned}$$

which is a t-student. (Final calculations on the student to check Yuanyuan...)

PS5Q4 Load the Vanveer gene expression data used in a previous practical. Make use of the 20 'best' genes (according to a marginal t-test) by using the following commands.

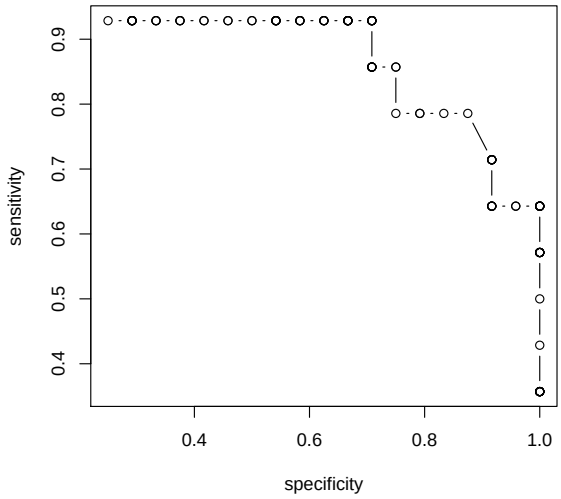
```
load(url("http://www.stats.ox.ac.uk/%7Eteh/MS1b/PracticalObjects.RData"))
vanv<- vanveer.4000[,2:21]
prog<- vanveer.4000[,1]
```

Your X matrix is thus `vanv` and the response Y is `prog`. Split the data into a test and training set (of equal size). Using logistic regression, plot a ROC curve.

Answer: Here is a R-script that plots the ROC curve. Many variations are possible, clearly.

```
load(url("http://www.stats.ox.ac.uk/~doucet/MS1/PracticalObjects.RData"))
vanv <- vanveer.4000[,2:21]
prog <- vanveer.4000[,1]

n <- nrow(vanv)  ## Number of samples
p <- ncol(vanv)  ## Number of variables (genes)
```

```
indtrain <- sort(sample(1:n,round(n/2)))  ## which observations
                                          ## to use for training?
indtest <- (1:n)[ -indtrain]              ## remaining for testing

Y <- as.numeric(prog=="good")    ## make binary response
                                 ##  vector 0="bad" and 1="good"
Ytest <- Y[indtest]              ## response for training data
Ytrain <- Y[indtrain]            ## response for test data

X <- data.frame(scale(vanv))     ## scaled predictor matrix X
Xtrain <- X[indtrain,]           ## for training and for
Xtest <- X[indtest,]             ## test data

fitglm <- glm(Ytrain ~ ., data=Xtrain,family=binomial)  ## fit GLM model

                                  ## estimate probabilities for "good"
                                  ## on test data
predicted <- predict(fitglm, newdata=Xtest,type="response")

cutoff <- seq(0,1,length=100)   ## where to put the decision boundary ?
specificity <- numeric(length(cutoff))
sensitivity <- numeric(length(cutoff))


## loop over all decision boundaries
for (cutc in 1:length(cutoff)){

  specificity[cutc] <- mean( predicted[ Ytest==1] > cutoff[cutc] )
  sensitivity[cutc] <- mean( predicted[ Ytest==0] <=cutoff[cutc] )

}

## plot results
plot(sensitivity, specificity, type="b")
```



Reading just one point on the curve: if 80% of all "good" patients are thus correctly classified as "good", also roughly 80% of all "bad" patients are classified correctly as "bad". The ROC curve here is not very smooth as there are not very many samples in the test data set.

PS6Q3 Load the Vanveer gene expression data used in a previous practical and the previous problem sheet. Make use of the 20 'best' genes (according to a marginal t-test) by using the following commands.

load(url("http://www.stats.ox.ac.uk/%7Eteh/MS1b/PracticalObjects.RData"))

vanv <- vanveer.4000[,2:21]

prog <- vanveer.4000[,1]

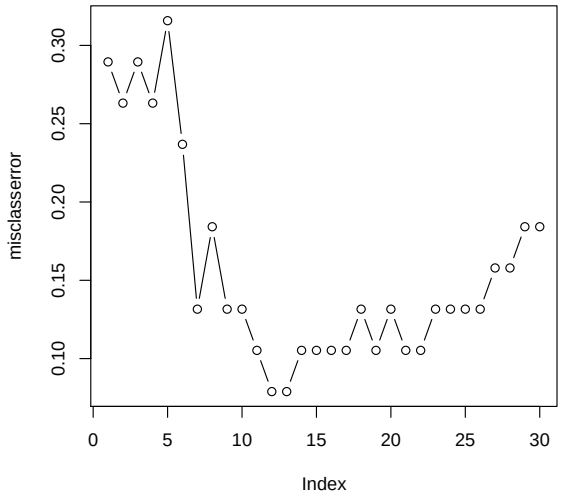
Your X matrix is thus `vanv` and the response Y is `prog`. Split the data into a test and training set (of equal size).

Use k-nearest neighbour classification. Find an estimate of the test error rate as you vary k, the number of nearest neighbours. What seems to be a good choice of k, the number of nearest neighbours? What is the estimated misclassification error under an optimal choice of k? Is it possible to produce a ROC curve for k-nearest neighbour classification?

Answer:

```
kvec <- 1:30    ## test k-NN for k=1,...,30
                ## record miscl. error in vector misclasserror
misclasserror <- numeric(length(kvec))

## loop over k=1,...,30
for (k in kvec){
```

```
  predict <- numeric(length(Ytest))  ## vector with prediction

  ## loop over test samples i
  for (i in 1:length(Ytest)){

    ## vector with distances between test sample i and
    ## all training samples j
    distance <- numeric(length(Ytrain))
    for (j in 1:length(Ytrain)){
      distance[j] <- mean( (Xtrain[j,]-Xtest[i,])^2 )
    }
    ## which are the nearest neighbours?
    nearestneighbour <- order(distance)[1:k]

    ## what is the most common value among them ?
    meanpred <- mean(Ytrain[nearestneighbour])

    ## take this to be the prediction
    predict[i] <- if( meanpred>0.5) 1 else 0

  }

  ## record misclassification error for this value of k
  misclasserror[k] <- mean(predict != Ytest)

}

plot(misclasserror,type="b")
```



A good choice of k is a choice that minimizes the misclassification error rate. From the estimated curve, $k = 12$ seems a reasonable choice. The misclassification error for this choice is in general, however, larger than the estimated curve would suggest (as we picked this value to be the minimum) and can only be honestly assessed with new test data or a more elaborate cross-validation scheme.

K-nearest neighbours does not produce estimated probabilities (at least for $k = 1$), and so it is very difficult to impossible to adjust the tradeoff between sensitivity and specificity (and thus produce a ROC curve). For large values of k, the empirical proportion of samples among the k nearest neighbours could be taken as a probability estimate, albeit a very coarse one.

PS7Q3 The files `wine.txt` is available on the course website. You can again read it directly with

```
td <- read.table(
        "http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/wine.data",
        sep=",")
```

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

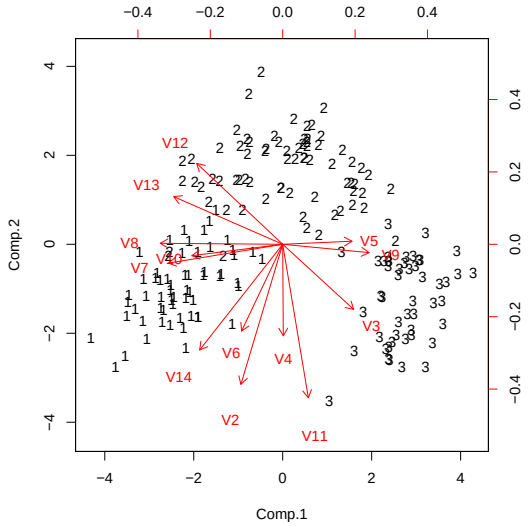
The first column contains the class label (1, 2 or 3), which is denoting the grower the wine came from. The goal is to predict the grower (the class 1, 2 or 3) given, some predictor variables. These are, in columns 2-14,

2) Alcohol

3) Malic acid

4) Ash
5) Alcalinity of ash
6) Magnesium
7) Total phenols
8) Flavanoids
9) Nonflavanoid phenols
10) Proanthocyanins
11) Color intensity
12) Hue
13) OD280/OD315 of diluted wines
14) Proline.

(a) Make a biplot using the `scale=0` option, and then use the `xlabs=as.numeric(td$Type)` option in `biplot()` to label points by their `$Type`. The output should look like:



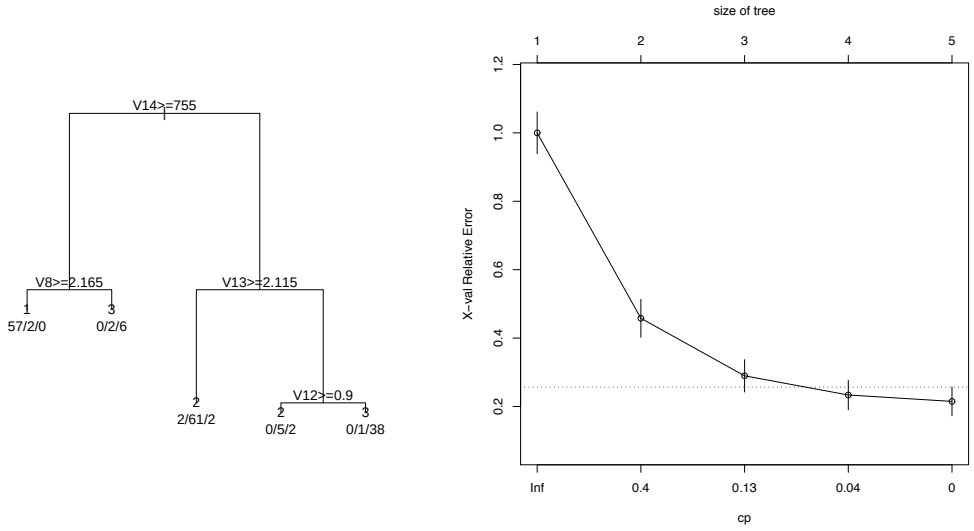
Answer:

```
library(rpart)
library(MASS)
td <- read.table("http://www.stats.ox.ac.uk/%7Eteh/teaching/datam
names(td)[1] <- "Type"
td[,1] <- as.factor(td[,1])

td.pc <- princomp(td[,2:14],scale=0,cor=T)
biplot((td.pc),xlabs=as.numeric(td$Type))
```

(b) Now make a classification tree analysis, and relate the decision rules discovered there to the projections of the original variable axes displayed in the biplot.

Answer:

```
td.tree <- rpart(Type~.,td,parms=list(split='gini'),
                 control = rpart.control(xval=10),method="class")
plot(td.tree)
text(td.tree,use.n=TRUE,digits=4,pretty=0);
plotcp(td.tree)
```



The tree was pruned at CP=0.04 (reading from the graph, and choosing the smallest tree withing 1 sd of the minimum). The relative cross validation error rate was 0.23 relative to an apparent/training root error rate of 0.6.

Referring to the biplot, variable V14 increases towards the 1's. The tree routes TRUE to the left. Splitting on large V14 we capture all the 1's and also a few 3's from small values of the 2nd principal component (PC). Variable V8 is aligned with the 1st PC (or rather, it's -ve), so splitting on large values of V8 (-ve 1st PC) removes these residual 3's. Variable 13 has a component in the direction of positive 2nd PC, which splits the 2's and 3's routed to the right at the first split. Note that this is really just a consistency check - we cannot trust that the points at the end of a particular variable vector in a biplot will have a large value of that component (since there are more than two variable vectors in the biplot, they are linearly dependent).

(c) Now produce a Random Forest fit, calculating the out-of-bag estimation error and compare with the tree analysis. You could start like:

```
library(randomForest)
rf <- randomForest(td[,2:14],td[,1],importance=TRUE)
print(rf)
```

Experiment with the parameter `mtry`, the random number of variables the best splitpoint is searched over. Use `varImpPlot` to determine what are the most important variables.

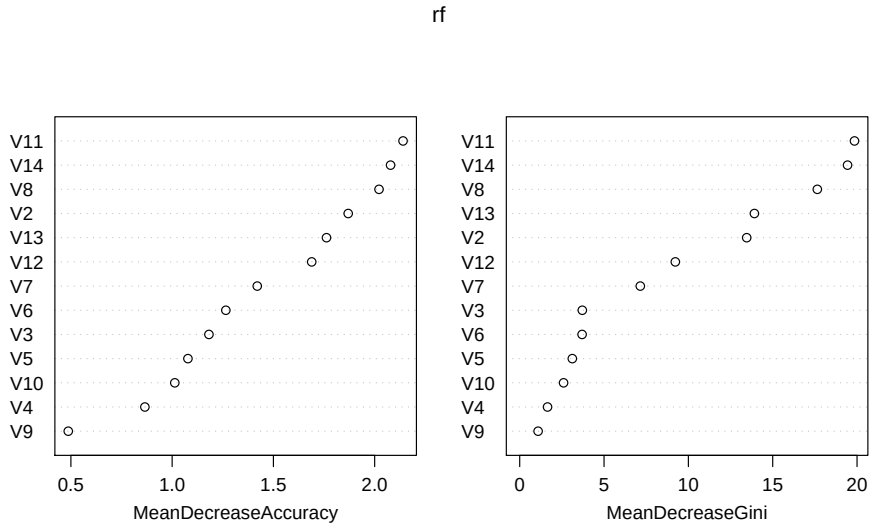
Answer: The out-of-bag estimation error is lower than for a single tree, as the following output shows

```
> print(rf)

Call:
 randomForest(x = td[, 2:14], y = td[, 1], importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.69%
Confusion matrix:
   1  2  3 class.error
1 59  0  0  0.00000000
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

The variable importance plot is obtained by

```
> varImpPlot(rf)
```

Two measures of importance are computed, which give a way of interpreting the forest (which is harder than for a single tree). The left panel shows the decrease in accuracy if permuting variable x. A larger decrease in accuracy means a higher "importance".



Experimenting with the number of variables over which the best splitpoint is searched, one obtains in all cases better values than with a single tree, even though the choice matters. If searching over 9 variables, we get

```
> rf <- randomForest(td[,2:14],td[,1],mtry=9)
> print(rf)
> > >
Call:
 randomForest(x = td[, 2:14], y = td[, 1], mtry = 9)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 2.25%
Confusion matrix:
   1  2  3 class.error
1 58  1  0  0.01694915
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

```
rf <- randomForest(td[,2:14],td[,1],mtry=6)
print(rf)
```

while searching over 3 variables (the default) gives a similar answer as the initial Random Forests (Remember that the trees are random as the variables over which the best splitpoint is searched are chosen randomly. The answer will only be the same if the number of trees is chosen very large.)

```
> rf <- randomForest(td[,2:14],td[,1],mtry=6)
> print(rf)
> > >
Call:
 randomForest(x = td[, 2:14], y = td[, 1], mtry = 6)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 2.25%
Confusion matrix:
   1  2  3 class.error
1 58  1  0  0.01694915
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

This search can be done more systematically but the default value is close to optimal for this dataset.