Ecologia Numérica - Componente Teórico-Prática

Resolução do exame de 22-01-2018

Tiago A. Marques e Hugo Anjos

January 21, 2019

This resolution is in English. Note that since this is a dynamic exam, where the data changes for each student, this is just a possible resolution, using my own `set.seed` commands

In some exercises we show two different corrections, both being correct. Often one is the one that the professor had, the other, a version that Hugo thought would be easier to understand for the students.

Exercise 1

One should begin by creating the required .Rmd. Since you are reading this as the output of a dynamic report, you know we did that!

Exercise 2

Here we run the code with Tiago's day of the month, the 22^{nd}, as an argument to `set.seed`. Note this is not because we think that Tiago's birthday is in any way more important than Hugo's, but because Hugo only joined this resolution later in the game and this was already done!

```
set.seed(22)
# **** dias do mês em que o aluno faz anos,
#e.g. Carlos e Maria com anos a 1 e 13 de Maio, 0113
b0=rnorm(1,0,0.5)
b1=rnorm(1,0,1)
b2=rnorm(1,0,0.5)
b3=rnorm(1,0,1)
b4=rnorm(1,0,0.5)
b5=rnorm(1,0,1)
b6=rnorm(1,0,2)
n1=rpois(1,30)
n2=2*n1
n=4*n1
hab= sample(x=c("H1", "H2"),size=n,prob=c(0.5,0.5),replace=T)
est= sample(x=c("P", "V", "O", "I"),size=n,prob=rep(0.25,4),replace=T)
x1=c(runif(n2,0,10), runif(n2,10,20))
x2=c(runif(n2,0,10), runif(n2,5,15))
x3= c(rnorm(n1,0,1), rnorm(n1,1,1),rnorm(n1,2,1),rnorm(n1,3,1))
x4= c(rnorm(n1,0,1), rnorm(n1,0,1),rnorm(n1,0,1),rnorm(n1,1,1))
x5=rnorm(n,15,2)
x6=rnorm(n,0,5)
torf=sample(x=0:1,size=6,prob=c(0.2,0.8),replace=T)
ys=b0+b1*x1*torf[1]+b2*x2*torf[2]+b3*x3*torf[3]+
b4*x4*torf[4]+b5*x5*torf[5]+b6*x6*torf[6]+rnorm(n,0,5)
```

We now have the data required for the following exercises.

Exercise 2.1

```
length(ys)
```

```
## [1] 116
```

I generated 116 observations of `ys`.

Exercise 2.2

It will not be the same, most likely it will not be the same, but, by coincidence, it could be the same. It depends of the argument used inside the function `set.seed`. We can only be sure that all people which have birthday's on a 22 will get the same number (and we note that, just by coincidence, even a person that does not have their birthday on a 22^{nd} might have the same sample size).

Exercise 3

Exercise 3.1

The number of observations per habitat is shown below

```
table(hab)
```

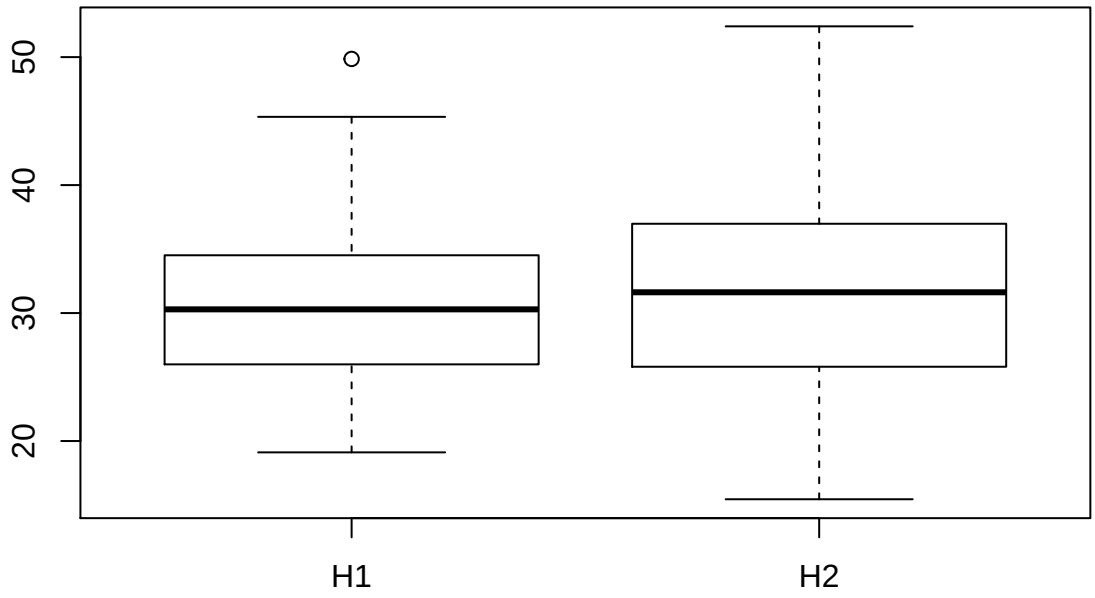
```
## hab
## H1 H2
## 48 68
```

so 48 observations in habitat 1 (H1) and 68 observations in habitat 1 (H2).

Exercise 3.2

The best plot to compare the values of two samples as a function of the level of a factor is a boxplot

```
boxplot(ys~hab)
```



There does not seem to be any large difference between the value of the index in the two habitats. The median and variance in H2 are both larger. One of the values in habitat 1 is a potential outlier.

Exercise 3.3

A formal test will be either a t-test (if parametric assumptions hold) or a Wilcoxon test (if they do not): Here the assumptions are that the values follow a Gaussian distribution, with equal variances (and naturally, that the observations are independent!).

Therefore, first we test if the data might reject these assumptions. First, the Gaussian assumption

```
shapiro.test(ys[hab=="H1"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ys[hab == "H1"]
## W = 0.97196, p-value = 0.3015
```

and

```
shapiro.test(ys[hab=="H2"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ys[hab == "H2"]
## W = 0.98411, p-value = 0.5398
```

and then the variance equality

```
bartlett.test(x=ys,g=hab)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  ys and hab
## Bartlett's K-squared = 1.2053, df = 1, p-value = 0.2723
```

Clearly, none of the tests is significant using any of the usual significance levels, hence we use the t-test

```
t.test(ys~hab)
```

```
##
##  Welch Two Sample t-test
##
## data:  ys by hab
## t = -0.75763, df = 109.5, p-value = 0.4503
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.614108  1.615080
## sample estimates:
## mean in group H1 mean in group H2
##         30.52970         31.52921
```

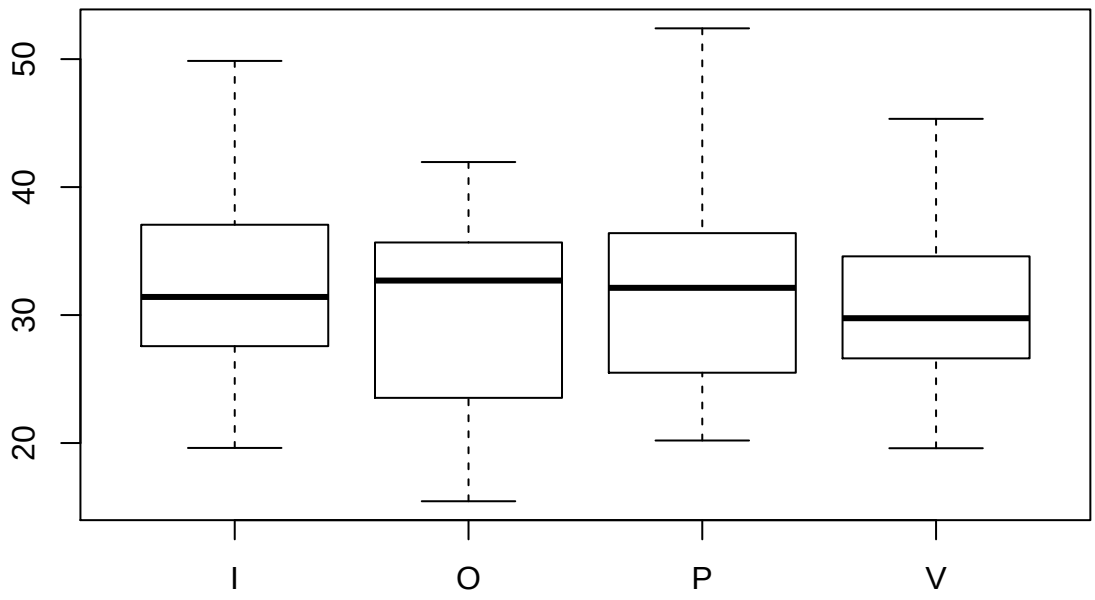
as anticipated from the plots above, there are no evidences that the indexes are different for each habitat, for any significance level considered.

Exercise 4

Exercise 4.1

The best plot to compare the values of several samples as a function of the level of a factor is, as for two samples, a boxplot

```
boxplot(ys~est)
```



The medians in Verão and Inverno seem smaller than in Outono and Primavera. There are no outliers. In general, no clear differences are present.

Exercise 4.2

As before, we test for Normality and variance equality (but note this was strictly not required, as we were told to use a parametric approach, hence an ANOVA). First, the Gaussian assumption

```
shapiro.test(ys[est=="V"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ys[est == "V"]
## W = 0.97297, p-value = 0.6427
```

```
shapiro.test(ys[est=="I"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ys[est == "I"]
## W = 0.97733, p-value = 0.7509
```

```
shapiro.test(ys[est=="P"])
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  ys[est == "P"]
## W = 0.96034, p-value = 0.4453
```

```
shapiro.test(ys[est=="O"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ys[est == "O"]
## W = 0.95441, p-value = 0.1787
```

and then the variance equality

```
bartlett.test(x=ys,g=est)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  ys and est
## Bartlett's K-squared = 1.8199, df = 3, p-value = 0.6106
```

No evidence for assumption failure, and hence, we can test the equality of the means with an ANOVA. The H0 is that $\mu_V = \mu_P = \mu_O = \mu_I$, and H1 that at least one mean is different from the others

```
aov1=aov(ys~est)
summary(aov1)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## est           3    111   36.87   0.712  0.547
## Residuals   112   5797   51.76
```

We cannot reject the H0, hence there is no evidence that the different seasons present different values for the index.

Exercise 4.3

In a non parametric context, after comparing several means (using a Kruskal-Wallis test), if there is evidence that at least one mean is different, one can conduct the Dunn test. The result will allow us to evaluate which of the groups are different.

Exercise 5

We begin by creating the required data.frame

```
datays=data.frame(ys,x1,x2,x3,x4,x5,x6)
```

Exercise 5.1

Before looking at the model results, we decide on a significance level of 1%.

Then, I fit the relevant linear model

```
lm1=lm(ys~.,data=datays)
```

Exercise 5.2

Now we can look at the corresponding summary:

```
summary(lm1)
```

```
##
## Call:
## lm(formula = ys ~ ., data = datays)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8598  -3.2878  -0.1353   3.1723  14.1256
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.81371    3.71861  -1.294   0.1982
## x1           0.09972    0.11503   0.867   0.3879
## x2           0.39830    0.15962   2.495   0.0141 *
## x3          -0.20220    0.44245  -0.457   0.6486
## x4          -0.09054    0.48942  -0.185   0.8536
## x5           2.17705    0.22854   9.526 5.17e-16 ***
## x6          -0.20314    0.10053  -2.021   0.0458 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.247 on 109 degrees of freedom
## Multiple R-squared:  0.492,  Adjusted R-squared:  0.4641
## F-statistic:  17.6 on 6 and 109 DF,  p-value: 3.658e-14
```

At the 1% significance level, only `x5` seems to be relevant to explain the `ys`. This would have been different if I had chosen the 5% significance level, since then `x2` and `x6` would then also be considered significant. Remember, to some extent the significance value one chooses is an arbitrary choice that can be changed depending on the objective of the test.

Exercise 5.3

Option 1

First, we need to find the required means for each independent variable

```
colMeans(datays)
```

```
##         ys         x1         x2         x3         x4         x5
## 31.1156215  9.6489823  7.0210410  1.4811780  0.3381589 14.9389998
##         x6
##  0.1091822
```

```
vp=as.numeric(colMeans(datays)[-1])
vp
```

```
## [1]  9.6489823  7.0210410  1.4811780  0.3381589 14.9389998  0.1091822
```

and then we can make the prediction (This is the formal way of doing it and it would work, not only for a lm, but for every other type of glm)

```
lmi1pMean=predict(lm1,newdata=data.frame(x1=vp[1],x2=vp[2],x3=vp[3],x4=vp[4],x5=vp[5],x6=vp[6]))
```

which is 31.12.

Option 2

We could also calculate, by hand, using the linear regression equation (Similar to one of the exercises in the theoretical exam). First we need the estimates of each coefficient

```
coef<-as.numeric(summary(lm1)$coefficients[,1])
coef
```

```
## [1] -4.81370799  0.09972422  0.39830312 -0.20220104 -0.09053624  2.17704537
## [7] -0.20313990
```

and then we simply calculate the index for the means of each variable

```
coef[1]+coef[2]*mean(x1)+coef[3]*mean(x2)+coef[4]*mean(x3)+coef[5]*mean(x4)+coef[6]*mean(x5)+coef[7]*mea
```

```
## [1] 31.11562
```

leading to the exact same result, naturally.

Reminder

This was this simple because we were considering a `lm`. In a `glm` or a `gam` one would have to be careful when predicting. This because if one is using any link function that is NOT the `identity` link, standard prediction will be obtained on the link scale (which we humans can't really understand) and we need to make predictions on the response scale. If using `predict`, we need to define the argument `type` to be `type="response"` so that the results come in the response scale. If doing it manually, we need to apply the inverse link function to get the result in the response scale. As an example, the inverse of the `log` link is the exponential function. This was in fact a trick required in one of the questions of the theoretical exam.

Exercise 5.4

```
names(summary(lm1))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

The R^2 value, as shown in the summary above, is 0.492. This means that while the variables can be used to some extent to predict the value of the index, there might be a significant amount of unexplained variability. Naturally, the adjusted R-squared is even lower (0.464).

Exercise 5.5

The variable `torf` was a vector of length 6, the number of variables available, and each element was an indicator variable, taking a value 0 or 1. Because each `torf` was multiplying by the corresponding parameter associated with each variable, e.g. `torf[3]*b3`, a 0 would mean that the parameter was 0, irrespective of the true value of the parameter. In our sample

```
torf
```

```
## [1] 0 1 0 1 1 1
```

which means that all of `x2`, `x4`,`x5` and `x6` actually had coefficients different from 0

```
torf*c(b1,b2,b3,b4,b5,b6)
```

```
## [1]  0.0000000  0.5039131  0.0000000 -0.1044797  1.8580924 -0.1320528
```

Assuming a significance level of 5% we would make 3 right calls (significant `x2`, `x5` and `x6`) and 2 right calls (non significant `x1` and `x3`), but considering `x4` as non-significant was a type II error: we did not reject the null when we should have. (we note however that given the true value of the coefficient, -0.1, the power to detect so was very small with this sample size, so the result was not surprising).

This was a question that would tell me if you can think - it was not a direct answer. Questions like these are required in an exam where students can consult their material and the internet. It tells me who can actually "join the dots".

Exercise 6

We run the requested code

```
set.seed(22)
# **** dias do mês em que o aluno faz anos,
#e.g. Carlos e Maria com anos a 1 e 13 de Maio, 0113
file=ceiling(runif(1,0,100))
```

Exercise 6.1

The value in file can be any integer between 1 and 100, and not 0 and 100 as most students said. Because the function `ceiling` rounds up !(so, read questions carefully and think about your answer!)

Exercise 6.2

It is a discrete uniform distribution, each number has a 1/100 probability of being sampled.

Exercise 6.3

It's 50/100=0.5

```
50/100
```

```
## [1] 0.5
```

Exercise 6.4

In my case inside `file` was the number 31.

```
file
```

```
## [1] 31
```

Exercise 7

We read in the correct file

```
dt1=read.table(file="data4EPENg31.txt",sep="\t")
```

and check it read OK

```
str(dt1)
```

```
## 'data.frame':    27 obs. of  10 variables:
##  $ hab : Factor w/ 27 levels "eucaliptal1",..: 18 20 21 22 23 24 25 26 27 19 ...
##  $ Ctat: int  5 9 4 4 9 5 7 3 9 3 ...
##  $ Cpat: int  4 1 1 1 4 3 4 4 4 3 ...
##  $ Came: int  3 4 3 4 3 2 1 7 4 3 ...
##  $ Pcac: int  2 3 2 0 4 4 2 2 2 3 ...
##  $ Pbev: int  6 8 4 6 6 9 8 6 3 1 ...
##  $ Pset: int  9 12 22 18 13 17 13 19 15 13 ...
##  $ Bcla: int  4 7 2 3 4 7 5 2 6 4 ...
##  $ Bste: int  10 7 4 1 5 1 8 5 4 4 ...
##  $ Blec: int  13 5 13 12 10 13 9 17 13 7 ...
```

Exercise 7.1 !!!!!!!!!!!!!!

The number of sites per habitat would be easier to count from the list below

```
dt1$hab
```

```
##  [1] pinhal1     pinhal2     pinhal3     pinhal4     pinhal5
##  [6] pinhal6     pinhal7     pinhal8     pinhal9     pinhal10
## [11] eucaliptal1 eucaliptal2 eucaliptal3 eucaliptal4 eucaliptal5
## [16] eucaliptal6 eucaliptal7 eucaliptal8 mato1       mato2
## [21] mato3       mato4       mato5       mato6       mato7
## [26] mato8       mato9
## 27 Levels: eucaliptal1 eucaliptal2 eucaliptal3 eucaliptal4 ... pinhal9
```

of course, R allows you to do almost everything, and the code below, despite you not being necessarily supposed to know about it (but this was used in one of the TP's to put suitable legends in a cluster analysis), would help and then you would not even have to count as it's done automatically

```
table(substr(dt1$hab,1,4))
```

```
##
## euca mato pinh
##    8    9   10
```

Exercise 7.2

We calculate the distance requested (note you need to remove the labels in the first column, most of you did not!)

```
dists=dist(dt1[-1],method="euclidean")
```

we can round the distances for easier printing

```
round(dists,1)
```

```
##       1    2    3    4    5    6    7    8    9   10   11   12   13   14
## 2  11.0
## 3  14.9 15.5
## 4  13.4 13.2  6.0
## 5   8.4  7.3 11.7  9.8
## 6  13.0 12.3  9.7  7.2  8.8
## 7   7.0  6.7 12.6 11.0  5.2  9.8
## 8  12.8 16.8  7.5  8.1 12.0 10.2 13.2
## 9  10.1 10.9 10.0  8.7  5.7  8.7  8.7 10.1
## 10 10.9 10.6 11.7 10.0  8.5 11.8  9.6 13.6  9.3
## 11 19.0 15.4 11.4 12.6 13.5 12.5 15.1 14.8 15.4 13.5
## 12 16.4 12.3 10.2  9.2 10.7  9.6 12.0 12.4 12.9 13.8  7.7
## 13 19.0 13.9 12.5 12.4 13.3 11.4 13.4 16.2 15.9 15.4  7.5  5.8
## 14 18.5 17.4  6.6  8.3 15.0 11.9 14.9 12.2 14.8 13.4 10.7 10.5 10.4
## 15 12.0 10.9  6.9  7.1  6.8  8.1  7.8  9.1  8.0 10.1  9.9  6.9  9.3  9.5
## 16 15.7 13.1 10.4 10.5 11.2 10.8 12.2 14.5 12.4  7.5  7.9 11.3 10.9 10.0
## 17 24.2 21.7 13.2 16.1 19.5 15.2 20.1 16.5 19.6 20.0  9.8 13.0 11.2 11.5
## 18 14.4  9.9 13.5 12.3  9.9 10.5  9.1 15.2 14.0 12.6  9.4  7.4  6.4 12.8
## 19 27.8 22.8 18.7 21.1 22.7 21.0 22.6 23.3 24.2 22.2 11.7 16.0 12.0 15.3
## 20 27.5 23.7 18.3 21.1 22.5 20.9 23.2 22.4 23.9 21.7 10.1 16.2 13.2 15.6
## 21 25.9 20.2 19.3 20.3 20.7 18.7 21.0 23.1 23.3 21.0  9.6 13.9 10.1 16.7
## 22 22.3 17.3 17.3 18.5 17.2 17.3 17.8 20.0 20.3 18.4  7.3 11.3  9.0 16.0
## 23 25.3 19.6 18.8 19.3 20.0 18.6 20.0 22.3 23.0 20.8  9.9 12.0  7.9 15.5
## 24 24.6 19.3 16.2 17.3 19.3 17.4 19.1 19.9 21.2 20.4 10.4 10.9  7.3 12.8
## 25 28.9 23.2 20.6 22.1 23.3 21.2 23.4 24.7 25.6 24.4 12.6 15.2 11.0 17.3
## 26 35.5 31.5 25.4 28.2 30.2 27.5 30.7 28.9 31.1 30.1 18.7 23.2 19.9 22.3
## 27 23.2 18.4 16.7 16.8 17.7 16.1 18.4 19.6 20.5 18.1  7.3 10.2  7.1 13.7
##      15   16   17   18   19   20   21   22   23   24   25   26
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16  9.7
## 17 14.1 14.2
## 18  8.3 10.4 14.9
## 19 17.9 16.1  9.8 15.9
## 20 17.9 15.3  9.3 16.6  5.1
## 21 17.6 14.5 12.0 13.2  8.7  8.9
## 22 14.2 13.3 12.1 10.2  9.8  9.2  6.5
## 23 16.2 15.4 12.5 11.5  8.5  9.8  6.0  6.5
## 24 14.1 15.5  9.9 11.9  7.4 10.0  9.8  9.1  5.9
## 25 18.9 18.3 11.7 15.5  6.1  8.5  7.1  9.2  5.4  6.3
## 26 25.1 23.9 13.5 23.9 10.2  9.8 15.6 16.8 15.6 14.6 11.7
## 27 14.1 12.6 11.4 10.0 10.1  9.7  7.3  6.5  4.7  7.5  8.7 16.3
```

and check what is the minimum value (minimum value, minimum distance, most similar locations)

```
min(dists)
```

```
## [1] 4.690416
```

you could now simply look on the above table for the value (And this would be correct. The exercise would be considered correct if you would tell me that you look the minimum number on the matrix and the two correspondent habitats), but an even more elegant way would be to use the function `which` to check which element in the matrix is equal to the minimum value on the matrix

```
which(as.matrix(dists)==min(dists),arr.ind = TRUE)
```

```
##    row col
## 27  27  23
## 23  23  27
```

and we see the maximum occurs between sites 27 and 23 (As this matrix is symmetrical, those two results indicate the same two habitats). That is actually

```
as.character(dt1[23,1])
```

```
## [1] "mato5"
```

```
as.character(dt1[27,1])
```

```
## [1] "mato9"
```

it makes sense that the closest locations belong to the same type of habitat!

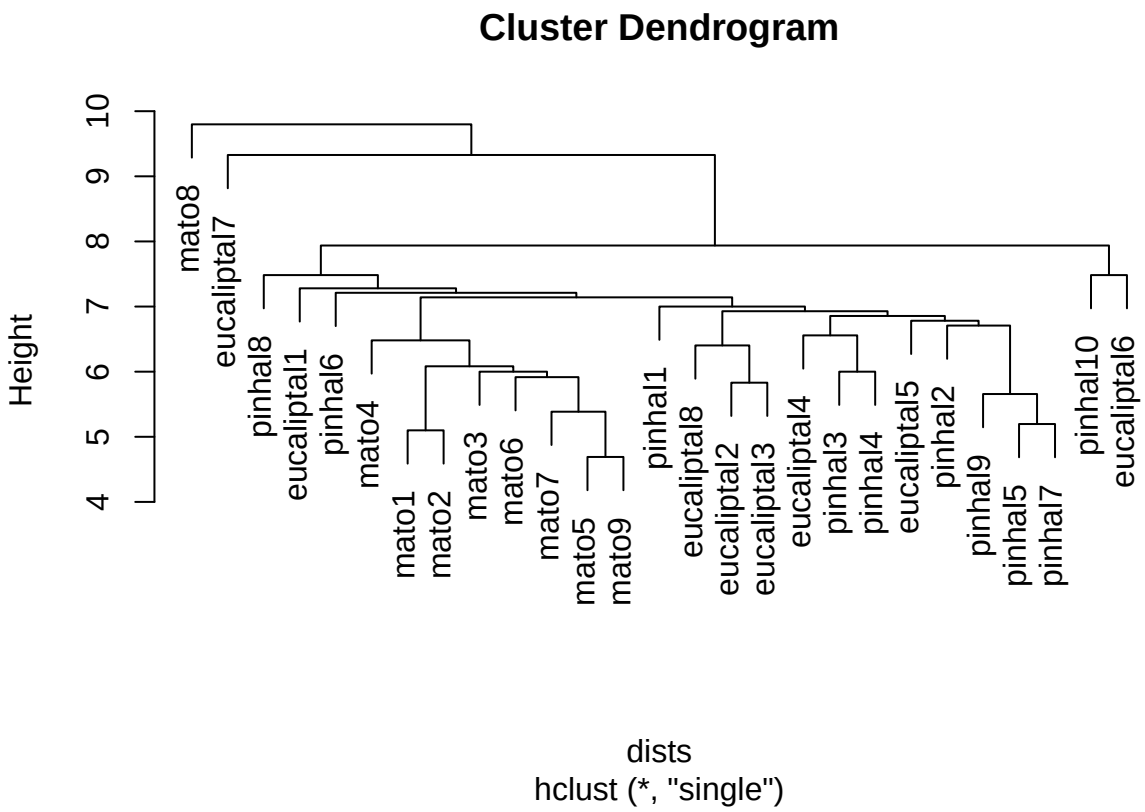
Exercise 7.3

We do the two analysis as requested

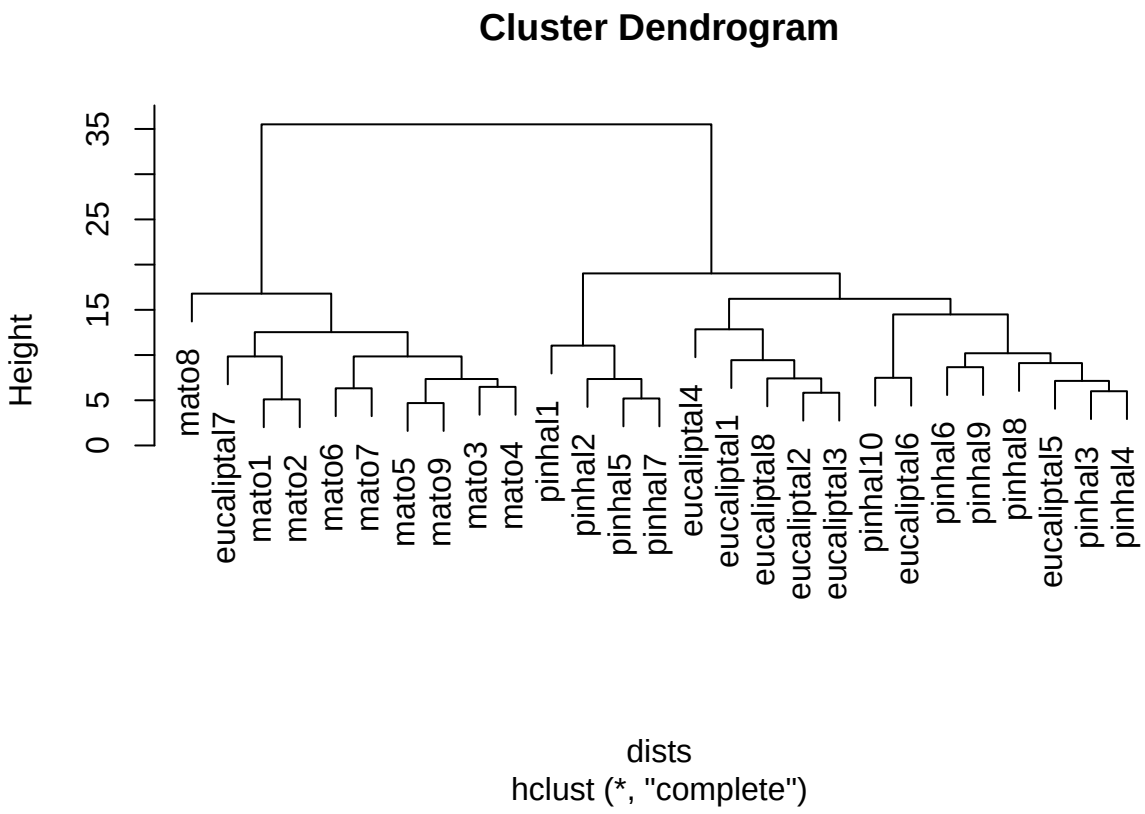
```
cl.s=hclust(dists,method="single")
cl.c=hclust(dists,method="complete")
```

We can look at the outputs

```
plot(cl.s,labels=dt1[,1])
```




```
plot(cl.c,labels=dt1[,1])
```



Exercise 7.4

The `complete` method provides a much cleaner habitat separation. All the `matos` are in a single clear group, where a single `eucaliptal` was present. The `eucaliptal` 1 to 4 and 8 where in a cluster. The pinhal sites where in a cluster, where nonetheless 2 `eucaliptal` were present. No such clear structure is present in the `single` method. This is a consequence of on `single` one joining groups by the smallest difference between any two elements, while on `complete` we join the groups by the smallest of the largest distance between all the elements in existing groups.

Exercise 8

Read the data in

```
dt2=read.table(file="data4EPENdt31.txt",sep="\t")
```

and check it read OK

```
str(dt2)
```

```
## 'data.frame':    40 obs. of  9 variables:
##  $ loc : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ prof: num  1.222 0.263 8.383 5.859 9.025 ...
##  $ alt : num  0.875 2.613 -0.582 4.892 3.494 ...
##  $ O2  : num  179 271 201 178 237 ...
##  $ pH  : num  6.17 6.68 8.02 6.72 8.22 ...
```

```
##  $ sal : num  34.9 34.8 35.6 34.3 35 ...
##  $ sus : num  9.69 8.94 9.7 8.33 8.28 ...
##  $ Mg  : num  0.0873 0.0745 0.0191 0.0103 0.0077 ...
##  $ Pb  : num  8.78 9.96 9.43 11.4 11.24 ...
```

Exercise 8.1

Here is the PCA, making sure we use the correlation matrix since the different covariates are measured in different units

```
pca1=prcomp(dt2[,-1],scale.=TRUE)
```

Exercise 8.2

Option 1

We can see how much variance was explained by the analysis

```
names(pca1)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

inside `pca1$sdev`.

```
pca1$sdev
```

```
## [1] 1.9621713 1.1357532 1.0288127 0.8531864 0.6620405 0.6016324 0.4388088
## [8] 0.2841719
```

Strictly, these are standard deviations, so we square them for getting variances

```
pca1$sdev^2
```

```
## [1] 3.85011614 1.28993536 1.05845550 0.72792702 0.43829758 0.36196157
## [7] 0.19255317 0.08075366
```

For percentages, we need to divide by the sum and multiply by 100

```
100*pca1$sdev^2/sum(pca1$sdev^2)
```

```
## [1] 48.126452 16.124192 13.230694  9.099088  5.478720  4.524520  2.406915
## [8]  1.009421
```

and the 2 first axis explained respectively

```
(100*(pca1$sdev^2)/sum(pca1$sdev^2))[1:2]
```

```
## [1] 48.12645 16.12419
```

in a total of

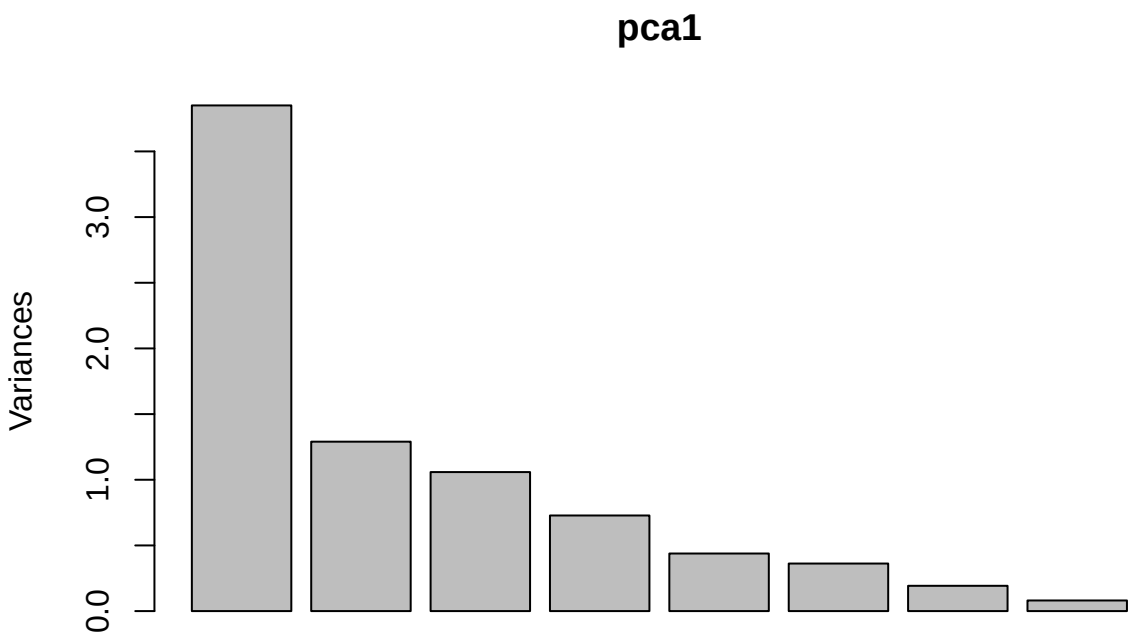
```
(100*sum(pca1$sdev[1:2]^2)/sum(pca1$sdev^2))
```

```
## [1] 64.25064
```

percent.

For completeness, we can see all axis variances in a scree plot

```
plot(pca1)
```



Option 2

Other ways to implement the PCA would have been say using `vegan`'s rda.

```
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-3
```

```
pca2<-rda(dt2[,-1],scale=TRUE)
summary(pca2)
```

```
##
## Call:
## rda(X = dt2[, -1], scale = TRUE)
##
## Partitioning of correlations:
##               Inertia Proportion
## Total               8          1
## Unconstrained       8          1
##
## Eigenvalues, and their contribution to the correlations
##
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Eigenvalue            3.8501 1.2899 1.0585 0.72793 0.43830 0.36196 0.19255
```

```
## Proportion Explained  0.4813 0.1612 0.1323 0.09099 0.05479 0.04525 0.02407
## Cumulative Proportion 0.4813 0.6425 0.7748 0.86580 0.92059 0.96584 0.98991
##                           PC8
## Eigenvalue            0.08075
## Proportion Explained  0.01009
## Cumulative Proportion 1.00000
##
## Scaling 2 for species and site scores
## * Species are scaled proportional to eigenvalues
## * Sites are unscaled: weighted dispersion equal on all dimensions
## * General scaling constant of scores:  4.202799
##
##
## Species scores
##
##           PC1      PC2      PC3       PC4      PC5      PC6
## prof  1.17246 -0.42795  0.02016 -0.604784  0.01523  0.33620
## alt   1.40357 -0.16819 -0.13825 -0.144068 -0.08869 -0.15446
## O2   -0.03345 -1.21942 -0.09064  0.821559 -0.01947  0.17671
## pH   -0.55618 -0.65571 -1.05298 -0.523163  0.18751 -0.04307
## sal  -1.29895  0.08704 -0.05113  0.005259  0.60488 -0.16021
## sus   0.36593  0.78833 -1.08148  0.436708 -0.16726  0.17813
## Mg    1.19423  0.27081  0.08136  0.208866  0.70885  0.32854
## Pb   -1.26426  0.13169  0.14720 -0.194102 -0.16662  0.68109
##
##
## Site scores (weighted sums of species scores)
##
##         PC1      PC2       PC3       PC4      PC5      PC6
## 1  -0.59799  0.97143  0.463813  0.675127  0.53912  0.34248
## 2  -0.81563 -0.09731  0.503144  1.047480  0.39539  0.87458
## 3  -0.96907 -0.01145 -0.601229 -0.524769  0.14396  0.38355
## 4  -0.94436  0.18409  0.851752 -0.668642 -0.95693  0.88520
## 5  -1.10848 -0.78929 -0.005278 -0.902334 -0.23280  1.02885
## 6  -0.61065  0.62545 -0.273631  0.637701 -0.03697 -0.02740
## 7  -0.79254 -0.04342 -0.126672  0.184660  0.05352  0.13953
## 8  -0.95297 -0.42906 -0.062496 -0.002322 -0.60267  0.94333
## 9  -0.31405  1.20849  0.175723  1.199257  0.01982  0.33079
## 10 -0.82179  0.61150 -0.392998  0.049680  0.07640  0.01359
## 11 -0.63630 -1.22779 -1.051897  1.069725  0.07348 -1.23207
## 12 -0.50967  0.69757 -0.622219 -0.093237  0.83963  0.07338
## 13 -0.56113  0.01408 -0.178785 -0.302394 -1.35449  0.15565
## 14 -0.50957  0.35736  0.450303 -0.263376 -1.27000 -0.54884
## 15 -0.38608  0.23412 -0.368532 -0.181947 -0.02685 -1.82605
## 16 -0.30484  0.76877  0.360358 -0.191612 -1.08179 -1.38959
## 17 -0.39886 -0.45335  0.921359 -0.826440  0.54188 -1.04698
## 18  0.08965 -1.12106  0.905333  0.373345  0.96039 -0.10760
## 19 -0.47072 -0.89865 -0.403891 -0.033890 -0.04206 -0.64966
## 20 -0.18721  0.08971  1.220595  0.209705  0.55479 -0.32704
## 21  0.03042  0.46601 -0.578021  0.604274  0.51303 -0.67266
## 22  0.28073  0.01681 -0.145522  0.440353 -0.77016 -0.01359
## 23  0.31747 -0.67168 -1.700389  0.237178  0.55922  0.59963
## 24  0.09955 -0.57295  0.352367 -1.568738  1.57631  0.17911
## 25  0.25788 -0.25295  0.386103  1.031153  0.93328  0.71451
```

```
## 26  0.47422  1.02144 -0.533749 -0.368200  0.93312  0.04985
## 27  0.52975 -0.32395  0.701967 -0.875016 -0.35532  0.89561
## 28  0.46724  0.67253  0.761462 -0.128950  0.92848 -0.66409
## 29  0.17289 -0.24252 -0.558644 -1.218596  0.20980  0.33470
## 30  0.41409 -0.99772 -0.357533  0.024166  0.54241  0.04639
## 31  0.80292 -0.01767  0.692805  0.869447 -0.10109  0.04055
## 32  0.98155  1.40422  0.090621 -1.178915  0.18681  0.13634
## 33  0.99425  0.36625 -0.402858  0.023212 -0.37665  0.70196
## 34  0.85880 -0.06927  1.145230  0.416142 -0.07218 -0.29945
## 35  0.72977 -0.64813  0.761893 -0.291228 -0.60159 -0.37566
## 36  1.12223 -0.88231  0.551255  0.991272 -0.94622  0.66902
## 37  0.85537  0.53246 -1.111013 -0.010712 -0.34315  0.60648
## 38  0.89033  0.33472 -0.686459  0.334475 -0.31060  0.17335
## 39  0.74017 -1.05889 -0.561763 -0.181765 -0.48748 -0.75636
## 40  0.78263  0.23241 -0.572506 -0.605269 -0.61186 -0.38140
```

It gives slightly different results because its a different function

```
summary(pca2)$cont$importance
```

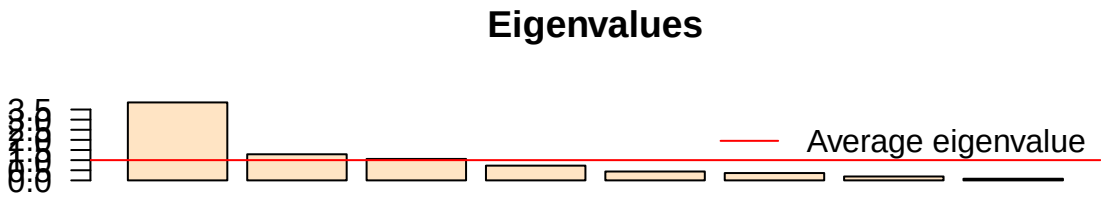
```
## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Eigenvalue            3.8501 1.2899 1.0585 0.72793 0.43830 0.36196 0.19255
## Proportion Explained  0.4813 0.1612 0.1323 0.09099 0.05479 0.04525 0.02407
## Cumulative Proportion 0.4813 0.6425 0.7748 0.86580 0.92059 0.96584 0.98991
##                            PC8
## Eigenvalue             0.08075
## Proportion Explained   0.01009
## Cumulative Proportion  1.00000
```

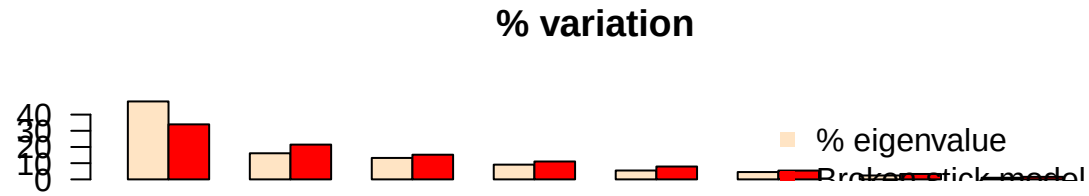
And from here we take that, the first axis explained 0.4812645 and the second 0.1612419. The total variance explained being 64.2506437

Exercise 8.3

We can use function `evplot` which provides two criteria

```
library(vegan)
source("brocardfunctions.R")
evplot(pca1$sdev^2)
```



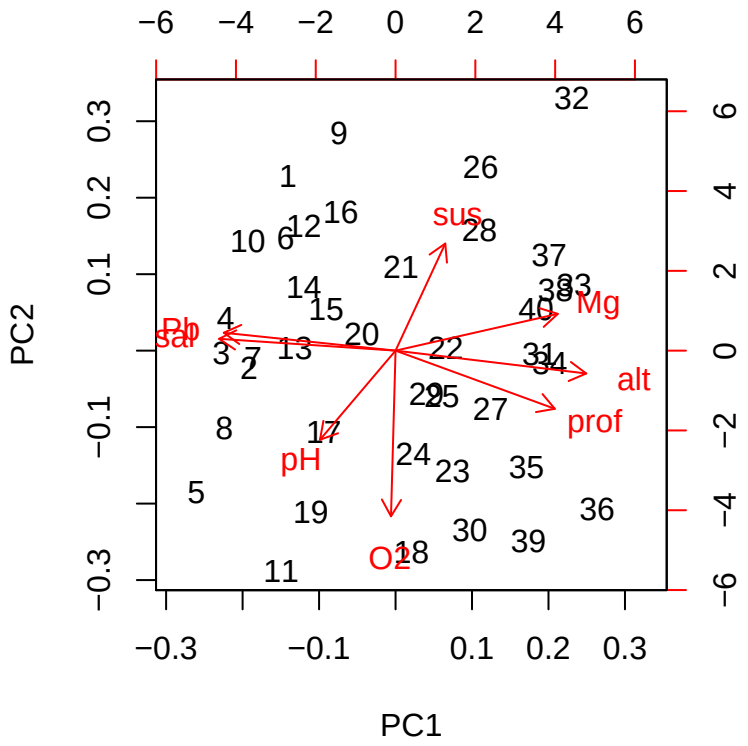


in this case,the first criteria suggests 3 axis, while the second suggests 1 axis.

Exercise 8.4

We can look at the biplot

```
biplot(pca1)
```



There seems to be an increase in the number of the site along the first axis, which implies that there is estuary to the source gradient. Given that the altitude is higher on the right side of the plot and salinity is high on the left side, we can infer that the sites with lower numbers were the ones near the estuary. Conversely, larger number should correspond to sites higher up in the mountain, near the river spring.

Exercise 8.5

Given the arrow corresponding to MG, and since the interpretation is done by projecting the site onto the variable arrow, sites 32, 37,33,and 40 are the ones with the highest values of Mg. Therefore, I would ask that the first places to be surveyed would be those near those sites, i.e., most upstream.