Using R for scientific computing - ANSWERS

Karline Soetaert
Centre for Estuarine and Marine Ecology
Netherlands Institute of Ecology
The Netherlands
June 2009

Abstract

The answers to the exercises from the document:
"Using R for scientific computing" (Soetaert 2008).

Keywords: scientific computing, lecture notes, R.

This document gives the answers to the exercises in the lecture notes:

"Using R for scientific computing" (Soetaert 2008).

These notes are an introduction to R - at beginners level. They can be found in package `marelacTeaching`.

In R, write:

```
require(marelacTeaching)

browseURL(paste(system.file(package="marelacTeaching"),
                "/lecture/R_for_scientific_computing.pdf", sep=""))
```

In order to make this vignette more readable, the questions are repeated.

Chapter 1

No exercises in this chapter

Chapter 2 - R as a scientific calculator

```
> (4/6*8-1)^(2/3)

[1] 2.657958

> log(20)

[1] 2.995732

> log2(4096)

[1] 12
```

```
> 2*pi*3

[1] 18.84956

> exp(2+cos(0.5*pi))

[1] 7.389056

> # length of 3rd side of a triangle with size 2.3 and 5.4 and angle  pi/8
> sqrt(2.3^2+5.4^2-2*2.3*5.4*cos(pi/8))

[1] 3.391288
```

Chapter 3 - computing with R-variables

Chapter 3.8.1

Use R-function `mean` to estimate the mean of two numbers, 9 and 17.

```
> mean(c(9,17))

[1] 13
```

- Create a vector, called V, with even numbers, between 16 and 56. Do not use loops.
- Display this vector
- What is the sum of all elements of V?
- Display the first 4 elements of V
- Calculate the product of the first 4 elements of V
- Display the 4th, 9th and 11th element of V.

```
> (V<-seq(16,56,by=2)) # creates AND displays the vector

 [1] 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56

> # or:
> V <- 16+2*(0:20)  ; V

 [1] 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56

> sum(V)

[1] 756

> V[1:4]

[1] 16 18 20 22

> prod(V[1:4])

[1] 126720

> V[c(4,9,11)]

[1] 22 32 36
```

- Create a new vector, W, which equals vector V, multiplied with 3; display its content.
- How many elements of W are smaller than 100?

```
> W<-V*3; W

 [1]  48  54  60  66  72  78  84  90  96 102 108 114 120 126 132 138
[17] 144 150 156 162 168

> W100<-W[W<100] ; length(W100)

[1] 9

> # or
> length(W[W<100])

[1] 9
```

- Create a sequence that contains the values (1,1/2,1/3,1/4,Ě,1/10)
- Compute the square root of each element
- Compute the square (^{2}) of each element
- Create a sequence with values (0/1,1/2,2/3,3/4,Ě,9/10)

```
> 1/1:10

 [1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000 0.1666667
 [7] 0.1428571 0.1250000 0.1111111 0.1000000

> sqrt(1/1:10)

 [1] 1.0000000 0.7071068 0.5773503 0.5000000 0.4472136 0.4082483
 [7] 0.3779645 0.3535534 0.3333333 0.3162278

> (1/1:10)^2

 [1] 1.00000000 0.25000000 0.11111111 0.06250000 0.04000000 0.02777778
 [7] 0.02040816 0.01562500 0.01234568 0.01000000

> (0:9)/(1:10)    # or : 0:9/1:10

 [1] 0.0000000 0.5000000 0.6666667 0.7500000 0.8000000 0.8333333
 [7] 0.8571429 0.8750000 0.8888889 0.9000000
```

- Create a vector, U, with 100 random numbers, uniformly distributed between -1 and 1.
- Check the range of U; all values should be within -1 and +1.
- Calculate the sum and the product of the elements of U

- How many elements of U are positive?
- Zero all negative values of U.
- Sort U

```
> U <- runif(100,-1,1)
> range(U)

[1] -0.9482796  0.9979586

> sum(U);prod(U)

[1] -4.869676

[1] 1.026173e-41

> length(U[U>0]) # or: sum(U>0)

[1] 42

> U[U<0]<-0
> sort(U)

  [1] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
  [6] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [11] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [16] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [21] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [26] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [31] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [36] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [41] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [46] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [51] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
 [56] 0.000000000 0.000000000 0.000000000 0.006788657 0.118877748
 [61] 0.147843781 0.157916529 0.164303903 0.213894379 0.230844138
 [66] 0.245174473 0.248841365 0.250033712 0.252987817 0.272850985
 [71] 0.298405104 0.385280319 0.424213514 0.447795870 0.459468868
 [76] 0.469037065 0.503672618 0.506756812 0.536059591 0.572214745
 [81] 0.590132813 0.622358007 0.628413523 0.639734149 0.674366430
 [86] 0.681879642 0.736476790 0.757944353 0.763706447 0.768675297
 [91] 0.803451715 0.815741586 0.870592395 0.886384066 0.895676645
 [96] 0.897811215 0.912143833 0.971938197 0.977795146 0.997958575
```

- Create two vectors: vector x, with the elements: 2,9,0,2,7,4,0 and vector y with the elements 3,5,0,2,5,4,6 (in that order).

- Divide all the elements of y by the elements of x.
- Select all values of y that are larger than the corresponding values of x
- Select all values of y for which the corresponding values of x are 0.
- Remove all values of y for which the corresponding values of x equal 0.
- Zero all elements of x that are larger or equal than 7. Show x.

```
> x<- c(2,9,0,2,7,4,0)
> y<- c(3,5,0,2,5,4,6)
> y/x

[1] 1.5000000 0.5555556       NaN 1.0000000 0.7142857 1.0000000
[7]       Inf

> x>y

[1] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE

> x==0

[1] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE

> y[y>x]

[1] 3 6

> y[x==0]

[1] 0 6

> y<-y[x!=0]
> x[x>=7]<-0 ; x

[1] 2 0 0 2 0 4 0
```

Chapter 3.8.2

- Use R-function "matrix" to create a matrix with the following contents:

$$\begin{bmatrix} 3 & 9 \\ 7 & 4 \end{bmatrix}$$

- display it to the screen

- Use R-function "matrix" to create a matrix called "A":

$$\begin{bmatrix} 3 & 9 \\ 7 & 4 \end{bmatrix}$$

- Take the transpose of A.
- Create a new matrix, B, by extracting the first two rows and first two columns of A. Display it to the screen.

```
> A<-matrix(nrow=2,data=c(3,7,9,4)) ; A

     [,1] [,2]
[1,]    3    9
[2,]    7    4

> A<-matrix(nrow=3,data=1/1:9,byrow=TRUE)  # or: 1/matrix(nrow=3,data=1:9,byrow=TRUE)
> t(A)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.2500000 0.1428571
[2,] 0.5000000 0.2000000 0.1250000
[3,] 0.3333333 0.1666667 0.1111111

> B <- A[1:2,1:2] ; B

     [,1] [,2]
[1,] 1.00  0.5
[2,] 0.25  0.2
```

Matrix D

- Use `diag` to create the following matrix, called "D":

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

- Use `cbind` and `rbind` to augment this matrix, such that you obtain:

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 2 & 0 & 4 \\ 0 & 0 & 3 & 4 \\ 5 & 5 & 5 & 5 \end{bmatrix}$$

- Remove the second row and second column of the previous matrix

```
> D   <- diag(nrow=3,c(1,2,3))
> DD  <- cbind(D,rep(4,3))  # or: cbind(D,4)
> DDD <- rbind(DD,rep(5,4)) # or: rbind(DD,5)
> DDD

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    4
[2,]    0    2    0    4
[3,]    0    0    3    4
[4,]    5    5    5    5

> # same, in one sentence
> DD <- rbind(cbind(D,4),5)
> DD[-2,-2]

     [,1] [,2] [,3]
[1,]    1    0    4
[2,]    0    3    4
[3,]    5    5    5
```

Chapter 3.8.3 - nematode diversity

- Select the data from station M160b (the 2nd column of Nemaspec); put these data in a vector called "dens".
- Remove from vector dens, the densities that are 0. Display this vector on the screen.
- Calculate N, the total nematode density of this station.
- Divide the values in vector dens by the total nematode density N. Put the results in vector p. The sum of all values in p should equal 1.
- Calculate S, the number of species.
- Estimate the values of diversity indices N1 and N2 and Ni, given by the following formulae:

$$N1 = e^{\sum -p_i \cdot \log_e(p_i)}$$
$$N2 = 1/\left(\sum p_i^2\right)$$
$$Ni = 1/\max(p_i)$$

- The expected number of species in a sample with size n, drawn from a population which size N, which has S species is given by:

$$ES(n) = \sum_{i=1}^{S} \left[1 - \frac{\binom{N-Ni}{n}}{\binom{N}{n}}\right]$$

  What is the expected number of species per 100 individuals ?

- Print all diversity indices to the screen, which should look like:

```
> head(Nemaspec)

                SPECIES M160a    M160b    M280a    M280b    M530a
1         Acantholaimus     0 6.580261 0.000000 1.120782 1.315487
2 Acantholaimus elegans     0 0.000000 1.439706 0.000000 3.956836
3 Acantholaimus iubilus     0 0.000000 0.000000 0.000000 0.000000
4      Acantholaimus M1     0 5.919719 0.000000 3.628518 0.000000
5     Acantholaimus M10     0 0.000000 0.000000 1.120782 0.000000
6     Acantholaimus M11     0 0.000000 0.000000 0.000000 0.000000
     M530b    M820a    M820b    M990a    M990b   M1220a   M1220b
1 1.727387 3.417313 3.748096 2.447545 3.728838 4.369345 4.787512
2 0.000000 0.000000 2.198407 5.080900 5.330997 3.644567 3.494481
3 1.193131 0.000000 0.000000 1.270450 1.372789 0.000000 0.000000
4 1.193131 1.155307 0.000000 5.052036 6.225598 0.000000 1.166825
5 0.000000 0.000000 0.000000 0.000000 0.000000 1.115981 0.000000
6 0.000000 0.000000 0.000000 0.000000 2.285694 0.000000 0.000000

> dens <- Nemaspec[,2]
> dens <- dens[dens>0]
> N    <- sum(dens);
> p    <- dens/N
> N0   <- length(p)
> N1   <- exp(sum(-p*log(p)))
> N2   <- sum(p*p)^(-1)
> Ni   <- 1/max(p)
> ESS  <- N0-1/choose(N,100)*sum(choose(n=(N-dens),k=100))
> c(N=N,N0=N0,N1=N1,N2=N2,Ni=Ni,ESS=ESS)

         N         N0         N1         N2         Ni        ESS
576.000000  97.000000  27.782793   8.364525   3.162870  40.502318
```

Chapter 4 user-defined functions

Chapter 4.4.1

```
> ## Sphere function
> Sphere <- function(radius)
+ {
+  vol   <- 4/3*pi*radius^3
+  surf  <- 4 *pi*radius^2
+  circ  <- 2*pi*radius
+  return(list(volume=vol,surface=surf,circumference=circ))
+ }
> Sphere(6371)

$volume
[1] 1.083207e+12

$surface
[1] 510064472

$circumference
[1] 40030.17
```

Chapter 4.4.2

The saturated oxygen concentration in water ($molkg^{-1}$), as function of temperature (T), and salinity (S) can be calculated by: $SatOx = e^A$ where : A= -173.9894 + 25559.07/T + 146.4813* loge(T/100) -22.204*T/100 + S * (-0.037362+0.016504*T/100-0.0020564 *T/100*T/100) and T is temperature in Kelvin (Tkelvin = Tcelsius+273.15).

- Make a function that implements this formula; the default values for temperature and salinity are $20°C$ and 35 respectively.
- What is the saturated oxygen concentration at the default conditions?
- Estimate the saturated oxygen concentration for a range of temperatures from 0 to $30°C$, and salinity 35.

```
> SatOx <- function(T=20,S=35)
+   {
+     T <- T+273.15
+     A= -173.9894 + 25559.07/T + 146.4813* log(T/100) -22.204*T/100 + S *
+       (-0.037362+0.016504*T/100-0.0020564 *T/100*T/100)
+     exp(A)
+   }
> SatOx()

[1] 225.2346
```

```
> SatOx(0:30)

 [1] 349.6542 340.6019 331.9557 323.6924 315.7901 308.2286 300.9890
 [8] 294.0533 287.4051 281.0288 274.9098 269.0344 263.3897 257.9638
[15] 252.7452 247.7235 242.8884 238.2306 233.7412 229.4118 225.2346
[22] 221.2020 217.3070 213.5431 209.9038 206.3833 202.9759 199.6764
[29] 196.4796 193.3808 190.3755
```

Chapter 4.4.3

The Fibonacci numbers are calculated by the following relation: $F_n = F_{n-1} + F_{n-2}$ With $F_1 = F_2 = 1$

- Compute the first 50 Fibonacci numbers; store the results in a vector.
- For large n, the ratio Fn/Fn-1 approaches the "golden mean"
- What is the value of F50/F49; is it equal to the golden mean?
- When is n large enough? (i.e. sufficiently close (<1e-6) to the golden mean)

```
> Fibo<-vector()
> Fibo[1:2]<-1
> for (i in 3:50) Fibo[i]<-Fibo[i-1]+Fibo[i-2]
> (1+sqrt(5))/2

[1] 1.618034

> Fibo[50]/Fibo[49]

[1] 1.618034

> Fibo[2:50]/Fibo[1:49]- (1+sqrt(5))/2

 [1] -6.180340e-01  3.819660e-01 -1.180340e-01  4.863268e-02
 [5] -1.803399e-02  6.966011e-03 -2.649373e-03  1.013630e-03
 [9] -3.869299e-04  1.478294e-04 -5.646066e-05  2.156681e-05
[13] -8.237677e-06  3.146529e-06 -1.201865e-06  4.590718e-07
[17] -1.753498e-07  6.697766e-08 -2.558319e-08  9.771908e-09
[21] -3.732537e-09  1.425702e-09 -5.445699e-10  2.080072e-10
[25] -7.945178e-11  3.034772e-11 -1.159184e-11  4.427569e-12
[29] -1.691314e-12  6.459278e-13 -2.466916e-13  9.414691e-14
[33] -3.597123e-14  1.376677e-14 -5.329071e-15  1.998401e-15
[37] -8.881784e-16  2.220446e-16 -2.220446e-16  0.000000e+00
[41]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
[45]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
[49]  0.000000e+00
```

Chapter 4.4.5 - nematode diversity all stations

- Make a function that will calculate the diversity indices for any data matrix.
- Calculate and show diversity on species level

```
> Diversity <- function (Dens,   # density, each column a station
+                        S=100)  # common number of individuals on which
+                                # to estimate expected number of species
+ {
+
+   nstat <- NCOL(Dens)                              # number of stations
+   if(is.vector(Dens)) Dens <- matrix(ncol=nstat,Dens)
+   div   <- matrix(nrow=nstat,ncol=6,data=NA)     # create matrix for results
+   rownames(div) <- colnames(Dens)
+   colnames(div) <- c("N","N0","N1","N2","Ninf",paste("ESS",S,sep=""))
+
+   for (i in 1:nstat)
+    {
+      dens<- Dens[,i]
+      dens<- dens[dens>0]  # selection of species present
+      N   <- sum(dens)     # N, total density
+      p   <- dens/N        # relative proportion
+      N0  <- length(p)     # N0 = number of species present
+      N1  <- exp(sum(-p*log(p)))   # N1 = exp(Shannon-Wiener)
+      N2  <- sum(p*p)^(-1)         # Na = sum(sp^a)^(1/(1-a)
+      Ni  <- 1/max(p)               # Ninf
+      ESS <- N0-1/choose(N,S)*sum(choose(n=(N-dens),k=S))
+      div[i,] <- c(N,N0,N1,N2,Ni,ESS)
+    }
+   return(div)
+ }
> summary(Nemaspec)                  # calculate summary characteristics
```

```
                 SPECIES        M160a             M160b
 Acantholaimus        :  1   Min.   :  0.000   Min.   : 0.000
 Acantholaimus elegans:  1   1st Qu.:  0.000   1st Qu.: 0.000
 Acantholaimus iubilus:  1   Median :  0.000   Median : 0.000
 Acantholaimus M1     :  1   Mean   :  1.226   Mean   : 1.487
 Acantholaimus M10    :  1   3rd Qu.:  0.000   3rd Qu.: 1.341
 Acantholaimus M11    :  1   Max.   :182.113   Max.   :30.982
 (Other)              :464
    M280a            M280b            M530a            M530b
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000
 1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000
 Median : 0.000   Median : 0.000   Median : 0.000   Median : 0.0000
 Mean   : 1.143   Mean   : 1.047   Mean   : 0.849   Mean   : 0.8553
```

```
 3rd Qu.: 1.183   3rd Qu.: 1.121   3rd Qu.: 0.000   3rd Qu.: 0.0000
 Max.   :54.031   Max.   :33.500   Max.   :45.938   Max.   :29.2128

     M820a             M820b             M990a
 Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
 1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
 Median : 0.0000   Median : 0.0000   Median : 0.0000
 Mean   : 0.9234   Mean   : 0.9383   Mean   : 0.9596
 3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 1.1771
 Max.   :78.3521   Max.   :38.0481   Max.   :47.8478

     M990b            M1220a            M1220b
 Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
 1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000
 Median : 0.000   Median : 0.0000   Median : 0.0000
 Mean   : 1.217   Mean   : 0.7213   Mean   : 0.7085
 3rd Qu.: 1.213   3rd Qu.: 0.0000   3rd Qu.: 0.0000
 Max.   :51.271   Max.   :37.5895   Max.   :25.6069

> # remove species names
> (divspec<-Diversity(Nemaspec[,-1]))

          N  N0       N1        N2      Ninf   ESS100
M160a   576  97 27.78279  8.364525  3.162870 40.50232
M160b   699 126 90.15358 66.778414 22.561569 60.68971
M280a   537 148 83.57356 43.779249  9.938717 59.44306
M280b   492 140 87.18598 51.988253 14.686371 61.17590
M530a   399 107 61.03511 29.166239  8.685560 54.97142
M530b   402 105 66.41245 44.308181 13.761081 54.25870
M820a   434 102 47.47238 20.190796  5.539098 48.29562
M820b   441 115 60.48626 33.178177 11.590593 52.14167
M990a   451 121 72.89122 40.725866  9.425726 57.11996
M990b   572 148 88.37314 47.535985 11.156321 61.16281
M1220a 339 106 65.53017 37.188517  9.018488 55.97925
M1220b 333  96 63.58659 41.009214 13.004317 54.90511
```

Chapter 4.4.6 - rarefaction diversity An alternative way of estimating the number of species per 100 individuals is by taking random 'subsamples' of 100 individuals and estimating the number of species from this subsample.

```
> dens <- Nemaspec[,2]
> dens <- dens[dens>0]  # selection of species present
> cs   <- round(dens)  # rarefaction method can only work with integer numbers
> ind  <- NULL         # individual organisms; each one belonging to a species
> for (i in 1:length(cs)) ind <- c(ind,rep(i,times=cs[i]))
> ind100 <-sample(ind,size=100)      # take 100 random individuals
> Spec   <-table(ind100)             # table of counts: speciesnr versus nr ind
```

```
> ESS100 <-length(Spec)              # length of Spec = number of species
> # or, three sentences combined in 1!
> length(table(sample(ind,size=100)))

[1] 38

> ESS100 <- vector()
> for (i in 1:1000) ESS100[i] <- length(table(sample(ind,size=100)))
> mean (ESS100)

[1] 40.527
```

Chapter 5 - statistics

- Perform a hierarchic clustering of the Nemaspec dataset and plot the dendrogram
- Perform a principal component analysis (PCA) and plot the results
- repeat the PCA analysis, with the first two stations removed

```
> nemaspec <- Nemaspec[,-1]
> hc <- hclust(dist(t(nemaspec)), "ave")
> par(mfrow=c(2,2))
> plot(hc)
> plot(hc, hang = -1)
> x <- prcomp(t(nemaspec))
> biplot(x)
> x2 <- prcomp(t(nemaspec[,-(1:2)]))
> biplot(x2)
> par(mfrow=c(1,1))
```

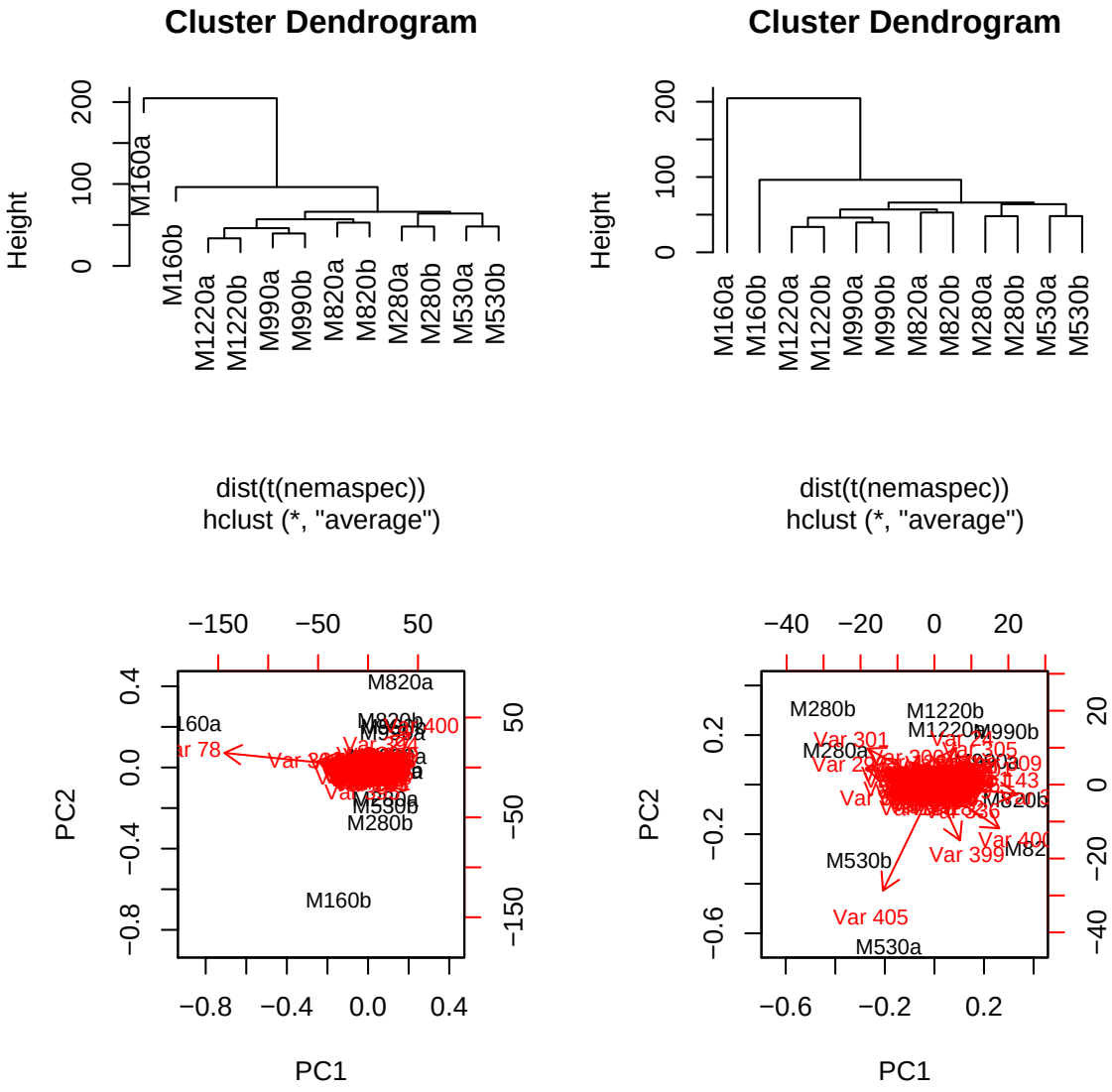

Figure 1: Cluster analysis and PCA of nematode data

Chapter 6 - graphics

- Create a script file which draws a curve of the function $y = x^3 sin^2(3\pi x)$ in the interval [-2 , 2].
- Make a curve of the function $y = 1/cos(1 + x^2)$ in the interval [-5,5].
- The relative importance of ammonia

$$p_{[NH_3]} = \frac{K_N}{K_N + [H^+]}$$

- Plot the relative fraction of toxic ammonia to the total ammonia concentration as a function of pH, where pH = -log10([H+]) and for a temperature of 30°C. Use a range of pH from 4 to 9. The value of KN is 810^{-10} at a temperature of 30°C.
- Add to this plot the relative fraction of ammonia at 0°C; the value of KN at that temperature is $810^{-11} molkg^{-1}$.
- For the US, the population density in 1900 (N0) was 76.1 million; the population growth can be described as:

$$N(t) = \frac{K}{1 + [\frac{K - N_{t0}}{N_{t0}}]e^{-a \cdot (t - t0)}}$$

  a=0.02 yr-1, K = 500 million of people.

  Actual population values are:

  1900 1910 1920 1930 1940 1950 1960 1970 1980

  76.1 92.4 106.5 123.1 132.6 152.3 180.7 204.9 226.5

- Plot the population density curve as a thick line, using the US parameter values.
- Add the measured population values as points. Finish the graph with titles, labels etcĚ

```
> par(mfrow=c(2,2))
> # simple curves
> curve(x^3*sin(3*pi*x)^2,-2,2)
> curve(1/cos(1+x^2),-5,5)
> # ammonia
> pN <- function(pH,Kn=8*10^-10) Kn/(Kn+10^-pH)
> curve(pN(x),4,9,main="fraction toxic ammonium")
> curve(pN(x,Kn=8*10^-11),4,9,add=TRUE,col="red")
> legend("topleft",lty=1,col=c("black","red"),c("30 dg","0 dg"))
> # US population
> K  <- 500
> N0 <- 76.1
> a  <- 0.02
> curve(K/(1+((K-N0)/N0*exp(-a*(x-1900)))),1900,1980,main="US population",
+       xlab="year",ylab= "million",lwd=2)
> N <- matrix(ncol=2,data=c(
```

```
+ seq(1900,1980,by=10), 76.1,92.4,106.5,123.1,132.6,152.3,180.7,204.9,226.5
+                          )  )
> points(N)
```

- Have a look at the `iris` data; What is the class and dimension of the data set?
- Produce a scatter plot of petal length against petal width
- Repeat the same graph, using different symbol colours for the three species.
- Create a box-and whisker plot for sepal length where the data values are split into species groups
- Now produce a similar box-and whisker plot for all four morphological measurements, arranged in two rows and two columns.

```
> 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

> class(iris)

[1] "data.frame"

> dim(iris)

[1] 150   5

> par(mfrow=c(2,2))
> plot(iris$Petal.Length,iris$Petal.Width,cex=1.5,pch=15,
+       xlab="Petal length", ylab=" Petal width")
> plot(iris$Petal.Length,iris$Petal.Width,cex=1.5,pch=15,
+      xlab="Petal length", ylab=" Petal width",
+      col=c("red","blue","green")[iris$Species])
> legend("bottomright",pch=15,col=c("red","blue","green"),
+      legend=levels(iris$Species))
> boxplot(Petal.Width~Species,data=iris)


> par(mfrow=c(2,2))
> boxplot(Sepal.Length~Species, data=iris,main="sepal length")
> boxplot(Sepal.Width~Species, data=iris,main="sepal width")
> boxplot(Petal.Length~Species, data=iris,main="petal length")
> boxplot(Petal.Width~Species, data=iris,main="petal width")
> mtext(outer=TRUE,side=3,line=-2,"Iris data set",cex=1.5)
```

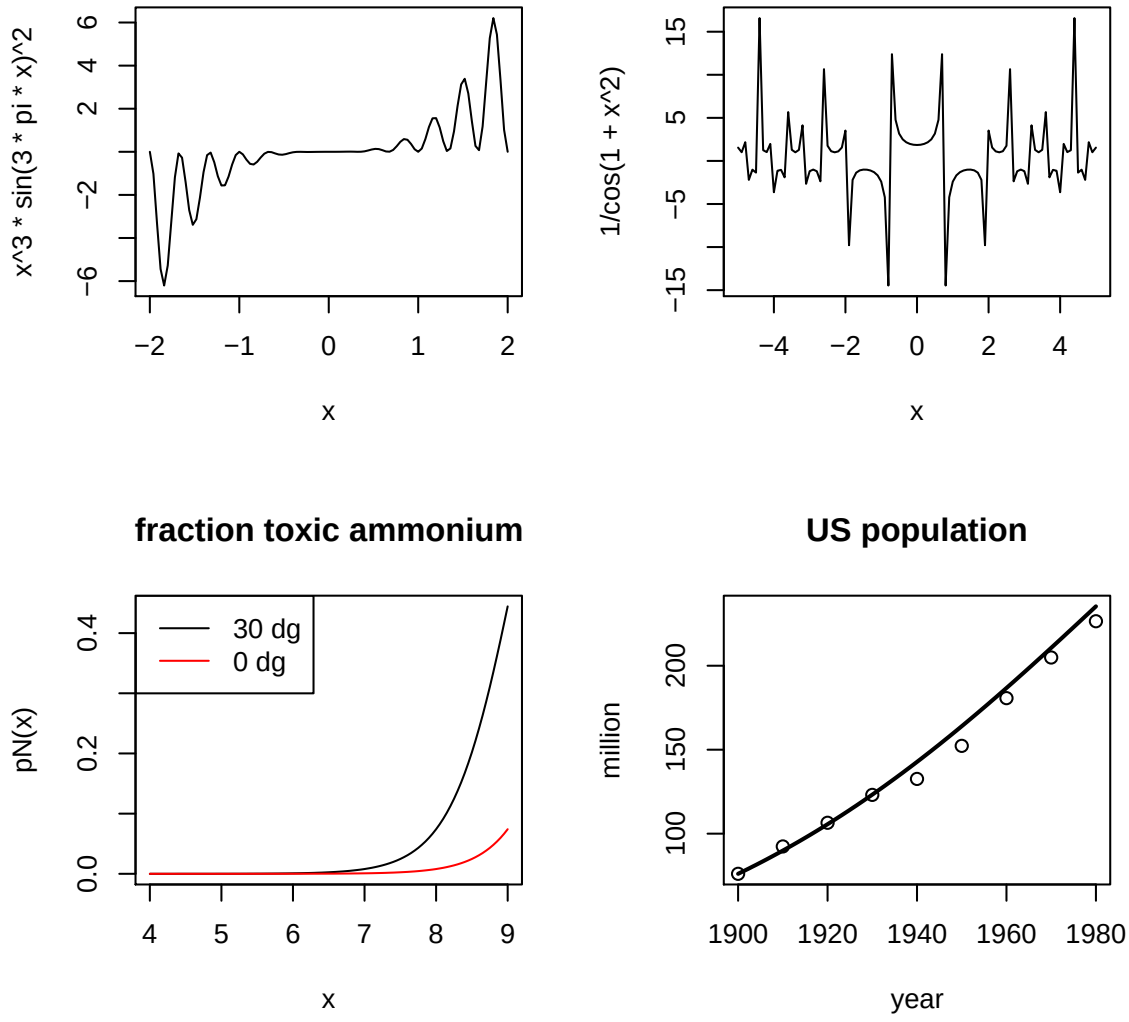

Figure 2: Use of R-function curve to plot simple functions

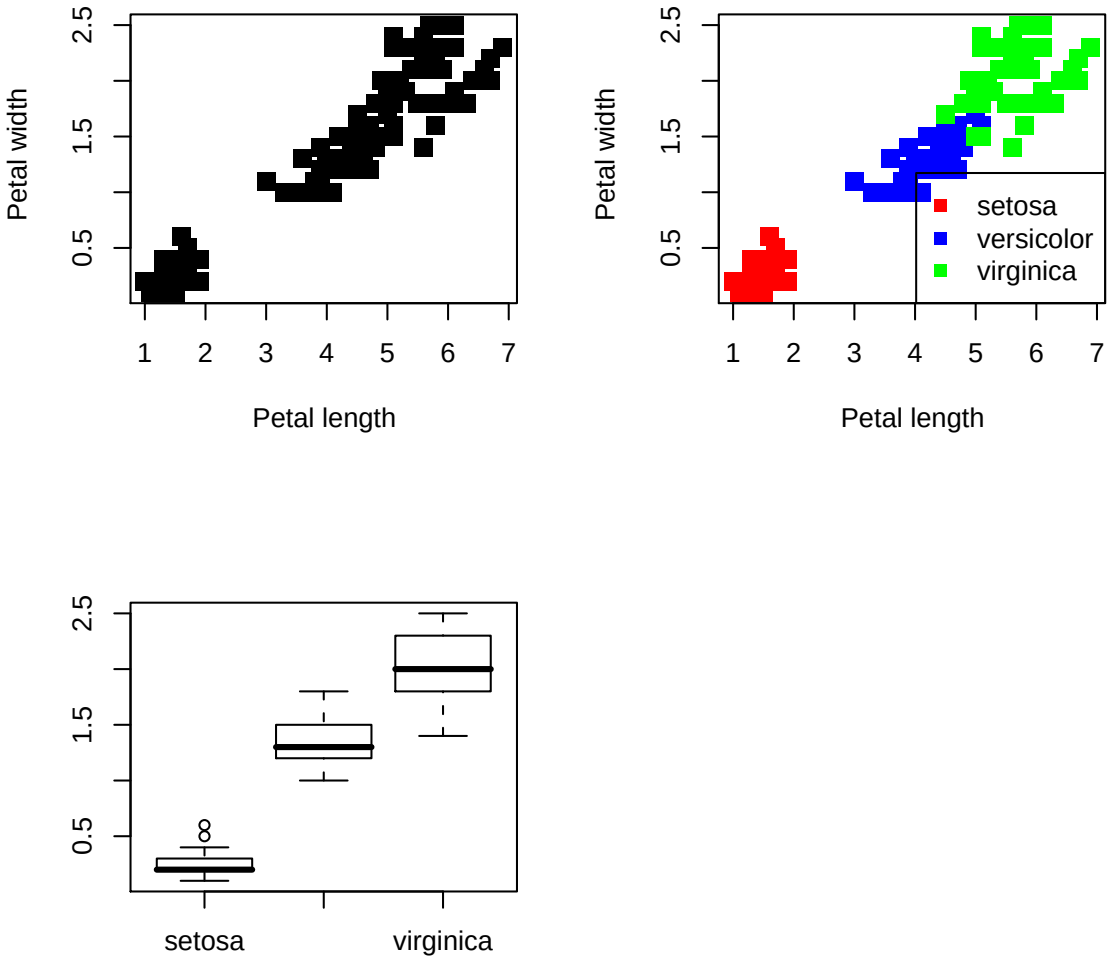

Figure 3: The iris data set

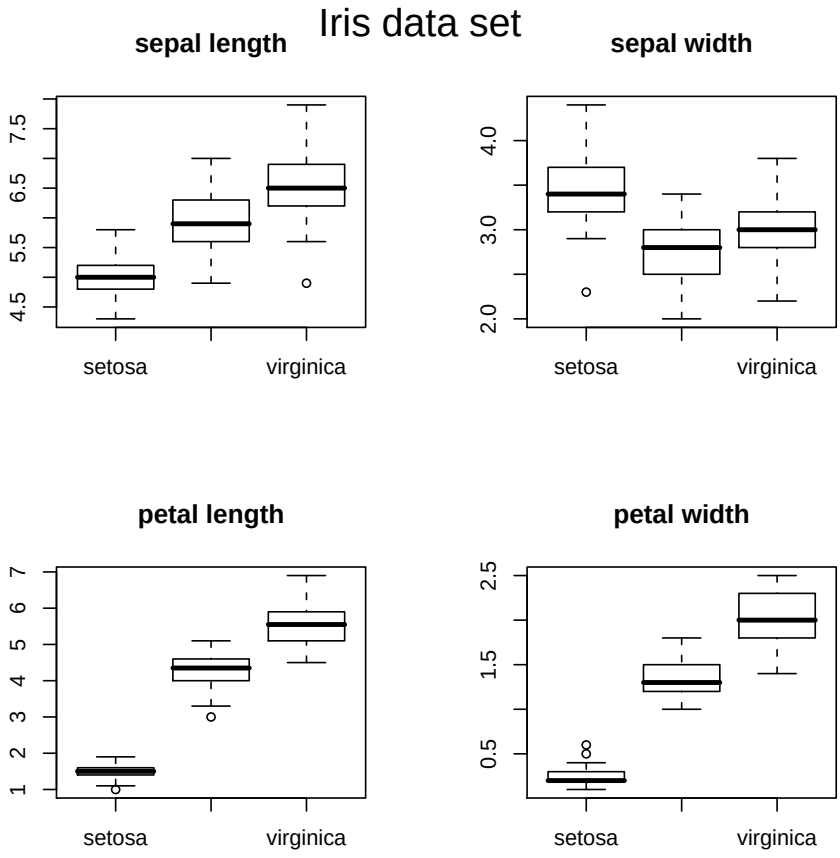

Figure 4: The iris data set

Chapter 7 - matrix algebra

Chapter 7.1.1

- Create matrices called "A" and "B":

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 1 \\ -2 & 1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

- Take the inverse of A and the transpose of A.
- Multiply A with B.
- Estimate the eigenvalues and eigenvectors of A.
- For a matrix A, x is an eigenvector, and ? the eigenvalue of a matrix A, if Ax =?x. Test it!

```
> A <- matrix(nrow=3, data=c(1,6,-2,2,4,1,3,1,-1))
> B <- matrix(nrow=3, data=1:9)
> solve(A); t(A)

            [,1]       [,2]       [,3]
[1,] -0.11111111  0.1111111 -0.2222222
[2,]  0.08888889  0.1111111  0.3777778
[3,]  0.31111111 -0.1111111 -0.1777778

     [,1] [,2] [,3]
[1,]    1    6   -2
[2,]    2    4    1
[3,]    3    1   -1

> A%*%B

     [,1] [,2] [,3]
[1,]   14   32   50
[2,]   17   50   83
[3,]   -3   -9  -15

> eigen(A)

$values
[1]  6.366696+0.000000i -1.183348+2.380697i -1.183348-2.380697i
```

```
$vectors
               [,1]                   [,2]                   [,3]
[1,] -0.36275602+0i -0.0725936-0.5240033i -0.0725936+0.5240033i
[2,] -0.93146469+0i -0.2632991+0.4856291i -0.2632991-0.4856291i
[3,] -0.02795726+0i  0.6441961+0.0000000i  0.6441961+0.0000000i
```

```
> ee<-eigen(A)
> A%*%ee$vectors[,1]

               [,1]
[1,] -2.3095572+0i
[2,] -5.9303521+0i
[3,] -0.1779954+0i
```

```
> ee$values[1]*ee$vectors[,1]

[1] -2.3095572+0i -5.9303521+0i -0.1779954+0i
```

Chapter 7.1.2 killer whale model

- Create a matrix, called P:

$$\begin{bmatrix} 0 & 0.0043 & 0.1132 & 0 \\ 0.9775 & 0.9111 & 0 & 0 \\ 0 & 0.0736 & 0.9534 & 0 \\ 0 & 0 & 0.0452 & 0.9804 \end{bmatrix}$$

- What is the value of the largest eigenvalue (the so-called dominant eigenvalue) and the corresponding eigenvector?.
- Create a new matrix, T, which equals P, except for the first row, where the elements are 0.
- Now estimate $N = (I - T)^{-1}$, where I is the identity matrix.

```
> A<- matrix (nrow=4,data=c(0,     0.9775,0,     0,
+                           0.0043,0.9111,0.0736,0,
+                           0.1132,0,     0.9534,0.0452,
+                           0,0,0,0.9804))
> eigen(A)

$values
[1] 1.025441326 0.980400000 0.834222976 0.004835698

$vectors
           [,1] [,2]       [,3]         [,4]
[1,] 0.06634512    0 -0.0659050  0.678780909
[2,] 0.56718211    0  0.8379894 -0.732135578
[3,] 0.57945357    0 -0.5175160  0.056807091
[4,] 0.58149491    1  0.1600233 -0.002631995
```

```
> T <- A
> T[1,] <- 0
> N <- solve(diag(4)-T) ; N

         [,1]     [,2]     [,3]     [,4]
[1,]  1.00000  0.00000  0.00000  0.00000
[2,] 10.99550 11.24859  0.00000  0.00000
[3,] 17.36628 17.76602 21.45923  0.00000
[4,] 40.04878 40.97062 49.48761 51.02041
```

Chapter 7.1.3. System of equations

Solve the following system of linear equations for the unknown xi:

3x1 + 4x2 + 5x3 = 0

6x1 + 2x2 + 7x3 = 5

7x1 + x2 = 6

Check the results

```
> A <- matrix(nrow=3,data=c(3,6,7,4,2,1,5,7,0))
> B <- c(0,5,6)
> x <- solve(A,B)
> A %*% x - B

             [,1]
[1,] 2.498002e-16
[2,] 0.000000e+00
[3,] 0.000000e+00
```

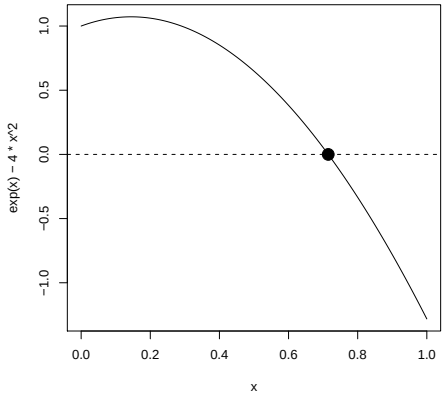

Figure 5: The root of a simple function

Chapter 8 - roots of functions

Chapter 8.3.1 simple root of equations

- Find the root of the equation $e^x = 4x^2$ in the interval [0,1]. First draw the function curve.
- Solve the equation $1000 = y * (3 + x) * (1 + y)^4$ for y and with x varying over the range from 1 to 100. Plot the root as a function of x.

```
> root<-uniroot(f=function(x) exp(x)-4*x^2,interval=c(0,1))

> curve(exp(x)-4*x^2,0,1)
> abline(h=0,lty=2)
> points(root$root,0,pch=16,cex=2)


> res<-vector()
> for (x in 1:100)
+  res[x]<-uniroot (f=function(y) y*(3+x)*(1+y)^4-1000,c(-1000,1000))$root

> plot(1:100,res)
```

Chapter 8.3.2. pCO2 rises increase acidity

- Estimate the pH at equilibrium with alkalinity 2300 $molkg^{-1}$ and the current pCO2 of 360 ppm.
- Use package seacarb to estimate the dissociation constants and Henry's constants at temperature 20°C, salinity 0, and pressure 0.
- Estimate pH as a function of pco2, varying between 200 and 1250

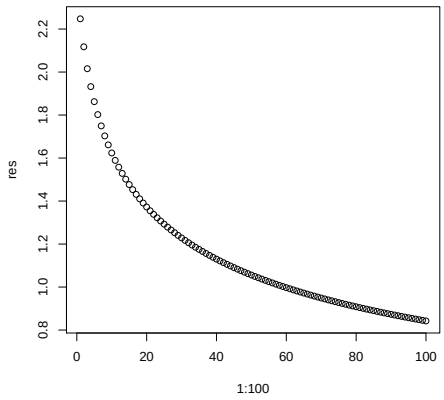

Figure 6: Roots of an equation y=f(x) for a sequence of x-values

- What is the value of pH at pCO2 = 1250

(see lecture notes for formulae)

```
> require(seacarb)
> k1 <- K1(S=0,T=20,P=0)
> k2 <- K2(S=0,T=20,P=0)
> kh <- Kh(S=0,T=20,P=0)
> nonlinfun <- function(pH,pco2=360,alk=2300e-6)
+ {
+  H    <- 10^(-pH)
+  CO2  <- pco2*kh
+  HCO3 <- k1*CO2/H
+  CO3  <- k2*HCO3/H
+  return( HCO3+2*CO3-H*1.e6 - alk)
+ }
> uniroot(nonlinfun,interval=c(2,12),pco2=360,alk=2300,tol=1e-30)

$root
[1] 8.317286

$f.root
[1] 2.728484e-12
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "total hydrogen ion concentration"

$iter
[1] 16
```

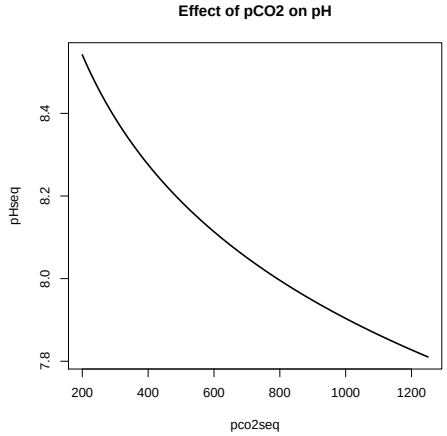

Figure 7: pH as a function of pCO2

```
$estim.prec
[1] 3.552714e-15

> pHseq   <- vector()
> pco2seq <-200:1250
> for (i in 1:length(pco2seq))
+     pHseq[i]<-uniroot(nonlinfun,interval=c(2,12),
+                       pco2=pco2seq[i],alk=2300,tol=1e-30)$root
> # max drop of pH
> pHseq[length(pHseq)]

[1] 7.81045

> plot(pco2seq,pHseq,type="l",lwd=2,main="Effect of pCO2 on pH")
```

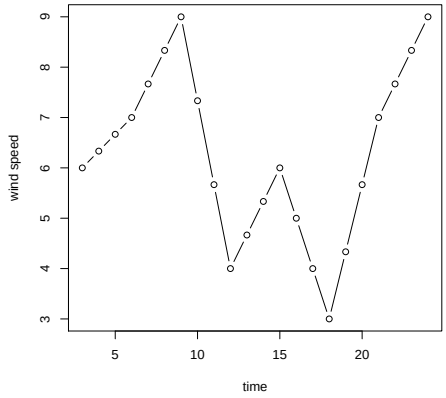

Figure 8: The interpolated wind data

Chapter 9 - interpolating, smoothing, curve fitting

Chapter 9.1 interpolating wind data Wind velocities are: 5,6,7,9,4,6,3,7,9 at time 0, 3, Ě24 o'clock respectively.

- Interpolate the three-hourly measurements to hourly measurements.
- Make a plot of the interpolated values

```
> t3    <- seq(3,24,by=3)
> wind3 <- c(6,7,9,4,6,3,7,9)
> plot(approx(t3,wind3,xout=3:24),type="b" ,xlab="time",ylab="wind speed")
```

Chapter 9.2 Fitting primary production data

Primary production (pp) at different light intensities are given; Fit the resulting production estimates (pp), as a function of light intensity (ll) with the 3-parameter Eilers-Peeters equation. The primary production is calculated as:

$$pp = p\max \cdot \frac{2 \cdot (1+\beta) \cdot {}^{I}/_{Iopt}}{({}^{I}/_{Iopt})^2 + 2 \cdot \beta \cdot {}^{I}/_{Iopt} + 1}$$

where I is light and pmax, ? and Iopt are parameters.

Add the best-fit line to the graph. (note: use coef to retrieve the best parameter values).

```
> ll <- c(0.,1,10,20,40,80,120,160,300,480,700)
> pp <- c(0.,1,3,4,6,8,10,11,10,9,8)
> plot(ll,pp,xlab= expression("light, ţEinst"~ m^{-2}~s^{-1}),
+      ylab="production",pch=15,cex=1.5)
> fit<-nls(pp ~pmax*2*(1+b)*(ll/iopt)/
+                          ((ll/iopt)^2+2*b*ll/iopt+1),
+      start=c(pmax=max(pp),b=0.005,iopt=ll[which.max(pp)]))
> summary(fit)
```

```
Formula: pp ~ pmax * 2 * (1 + b) * (ll/iopt)/((ll/iopt)^2 + 2 * b * ll/iopt +
    1)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
pmax  10.4351     0.3171  32.909 7.93e-10 ***
b      1.5998     0.4353   3.676  0.00626 **
iopt 209.6325    15.8052  13.264 9.96e-07 ***
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 0.5445 on 8 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 1.356e-06

> pars <- as.list(coef(fit))
> with(pars,
+  curve(pmax*2*(1+b)*(x/iopt)/((x/iopt)^2+2*b*x/iopt+1),
+        add=TRUE,lwd=2)   )
> title(expression (frac(pmax%*%2%*%(1+beta)%*%I/Iopt,
+                  (I/Iopt)^2+2%*%beta%*%I/Iopt+1)),cex.main=0.8)
```

```
Formula: pp ~ pmax * 2 * (1 + b) * (ll/iopt)/((ll/iopt)^2 + 2 * b * ll/iopt +
    1)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
pmax  10.4351     0.3171  32.909 7.93e-10 ***
b      1.5998     0.4353   3.676  0.00626 **
iopt 209.6325    15.8052  13.264 9.96e-07 ***
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 0.5445 on 8 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 1.356e-06
```

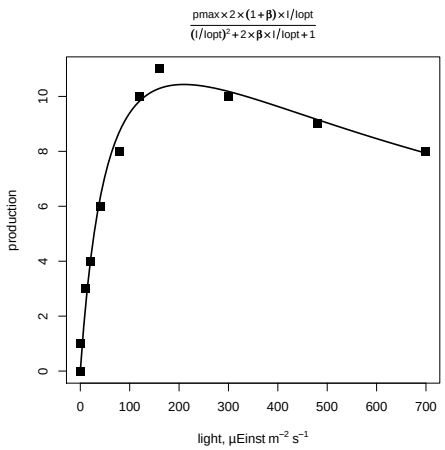

Figure 9: Primary production data with fit

Chapter 10 differential equations

Chapter 10.1 Lotka-Volterra model

- Solves the following system of ODEs

$$\frac{dx}{dt} = a \cdot x \cdot (1 - \frac{x}{K}) - b \cdot x \cdot y$$
$$\frac{dy}{dt} = g \cdot b \cdot x \cdot y - e \cdot y$$

  for initial values x=300,y=10 and parameter values: a=0.05, K=500, b=0.0002, g=0.8, e=0.03

- Make three plots, one for x and one for y as a function of time, and one plot expressing y as a function of x. Arrange these plots in 2 rows and 2 columns.

- run the model with other initial values (x=200, y=50); add the (x,y) trajectories to the phase-plane plot

```
> require(deSolve)
> model <- function (time,VAR,pars)
+ {
+  with (as.list(c(VAR,pars)), {
+   #  the rate of change of the state variables
+   dx     <- a*x*(1-x/K)-b*x*y
+   dy     <- g*b*x*y    - e*y
+
+  return(list(c(dx,dy)))
+                       })
+ }
> pars  <- c(a=0.05,b=0.0002,K=500,g=0.8,e=0.03)
> VAR   <- c(x=300,y=10)
> times <- seq(0,1000,1)
> out   <- as.data.frame(lsoda(VAR,times,model,pars))
> plot(out$x,out$y,type="l")
> VAR   <- c(x=200,y=50)
> out2  <- as.data.frame(lsoda(VAR,times,model,pars))
> lines(out2$x,out2$y,lty=2)
```

Chapter 10.2 Lorenz Butterfly

Solve the Lorenz equations:

$$\frac{dx}{dt} = -\frac{8}{3} \cdot x + y \cdot z$$
$$\frac{dy}{dt} = -10 \cdot (y - z)$$
$$\frac{dz}{dt} = -x \cdot y + 28y - z$$

Use as initial conditions x=y=z=1; create output for a time sequence ranging from 0 to 100, and with a time step of 0.005.

```
> require(scatterplot3d)
> model<-function(t,state,parameters)
```

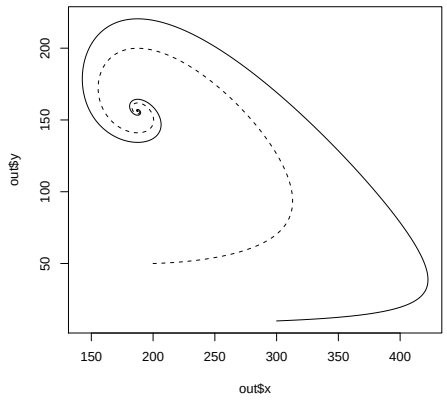

Figure 10: Result of the lotka-volterra model

```
+   {
+   with(as.list(c(state)),{
+
+     dx     <- -8/3*x+y*z
+     dy     <- -10*(y-z)
+     dz     <- -x*y+28*y-z
+
+     list(c(dx,dy,dz))            })
+
+  }  # end of model
> state <-c(x=1, y=1, z=1)
> times <-seq(0,100,0.005)
> out   <-as.data.frame(lsoda(state,times,model,0))
> scatterplot3d(out$x,out$y,out$z,type="l",
+         main="Lorenz butterfly",ylab="",grid=FALSE,box=FALSE)
```

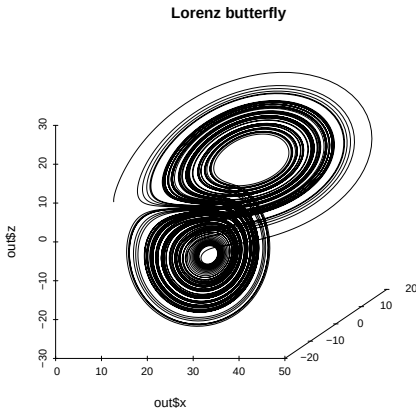

Figure 11: Results of the Lorenz model

References

Soetaert K (2008). *using R for scientific computing*. NIOO-CEME, Yerseke.

Affiliation:

Karline Soetaert
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands E-mail: k.soetaert@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/users/ksoetaert