Practical guidelines for performing Integrated Ecosystem Analyses:

The analyses presented below can be performed with any appropriate dataset, i.e. with time series of multiple variables representing the ecosystem. Nowadays, indirect ordination techniques are implemented in nearly all commercial statistical software packages. For identifying breakpoints in single data series, different methods exist. Here we present methods and software that are all available as freeware and thus can be directly used by the reader. Whenever possible we used R and related packages as language for statistical computing (R Development Core Team, 2007). The codes with short explanations will be described below. For users unfamiliar with R we refer to the website (http://www.R-project.org) and the manual "An introduction to R" (Venables et al., 2009). It is recommended to do the provided sample session first, before starting with any further analyses. Further, various books for beginners and advanced users exist and more will appear in near future. It is merely impossible to give an overview but a few recommendations can be found in table 2.

Table 1: Software (mainly freely available) for performing various methods used in Integrated Ecosystem Assessments.

Software	Methods	Authors	Availability	Web-Link
STARS	Detection of shifts in both the mean level of fluctuations and the variances → Sequential t-test/ F-test	S. N. Rodionov	Freeware, Add-in to Excel	http://www.beringclimate.noaa.gov/regimes/index.html
R-package (different from R CRAN)	Various complex multivariate and spatial analyses, includes Chronological Clustering	P. Casgrain, P. Legendre	Freeware, currently only available for MacOS	http://www.bio.umontreal.ca/Casgrain/en/labo/R/index.html
BRODGAR	Full statistical software. Includes various direct and indirect ordination methods, Chronological Clustering, DFA, MAFA, etc.	A.F. Zuur, E.N. Ieno	Commercial, stand-alone and with interface to R	http://www.highstat.com http://www.brodgar.com
R packages	Language and environment	R Development Core Team	Freeware	http://www.R-project.org
MASS		Venables, W.N., Ripley, B.N.	Freeware, package in R	http://cran.r-project.org/web/packages/MASS/index.html
vegan	Direct and indirect ordination methods, diversity analysis for community analyses	J. Oksanen, Kindt, R., Legendre, P., O'Hara, B., Simpson, G.L., Henry, M., Stevens, H., and Wagner, H.	Freeware, package in R	http://cran.r-project.org/web/packages/vegan/index.html
strucchange	Testing, monitoring and dating structural changes in (linear) regression models	Zeileis, A., Leisch,, F., Hornik, K., Kleiber, C.	Freeware, package in R	http://cran.r-project.org/web/packages/strucchange/index.html
rioja	Includes constrained clustering (→ Chronological	S. Juggins	Freeware, package in R	http://cran.r-project.org/web/packages/rioja/index.html

	Clustering)			

Table 2. A short list of books introducing R and its use for statistical analyses (with one example for time series analysis)

Authors (Publication Year)	Title (publisher)
Zuur, A.F., Ieano, E.N., Meesters, E.H.W.G. (2009)	A Beginner's Guide to R (Springer Science + Business Media)
Crawley, M.J. (2005)	Statistics: An introduction using R (John Wiley & Sons, Chichester)
Crawley, M.J. (2007)	The R book (John Wiley and Sons, Chichester)
Dalgaard, P. (2002)	Introductory statistics with R (Springer, New York)
Shumway, R.H., and Stoffer, D.S. (2006)	Time series analysis and its applications with R examples (Springer, New York)

Data preparation

For the described methods, data need to be organised in a $n * m$ data matrix, which means with *n* columns representing variables and *m* rows representing observations over time. Frequently observations represent annual estimates for the specific variable (if necessary averaged over space and time). If seasonal specific values should be considered they need to be treated as separate variables or in separate analytical runs.

In extensive time series missing values are often encountered. These data gaps should be few and preferably should not occur in consecutive years. Data exploration techniques, single variable analyses (e.g. regime shift analysis of single time series (STARS)) and data illustration (e.g. annual anomalies, traffic light plot) should be based on the "raw" data ("*Originaldata*" and "*Originaldata2*" - see the R code described below). In contrast, for PCAs and consequently for the analysis of sudden changes in multivariate datasets missing values need to be replaced and some variables need to be previously transformed. When performing a PCA many statistical programs exchange missing values with variable averages. However, if data show a long-term temporal trend, this might be obscured in the ordination plot. Generally, we recommend exchanging data gaps with the average of the four nearest data points. This might be not possible if gaps exist at the beginning or the end of the time series or consecutive observations are missing. Therefore we didn't write a code for exchanging missing values but advise the readers to do it themselves, e.g. in their original data spreadsheet, considering their best knowledge about temporal trends and variability in the data. Following the exchange of missing values and with the previous knowledge about the distribution form of the data, transformations of some variables might be necessary. To improve linearity between variables and reduce the relationship between the variance and the mean a logarithmic ($log_{10}(x+1)$, $ln(x+1)$) or root transformation is recommended for left-skewed variables, which is often the case for biological observations. The complemented and partly transformed dataset ("*Data.transf*" - see the R code described below) can then be used for the various multivariate analyses presented below.

To be able to use R for the analyses, the three data sheets (here called "*datasheet.txt*", "*datasheet2.txt*", and ("*datasheet_trans.txt*") with column and row labels need to be stored as separate tab-delimited text files. To be able to upload the data as described below you previously need to set the working directory to the folder containing your data (see "file" in the R toolbar). Other options are possible to import data (e.g. using a specific package for a direct import of data from Excel spreadsheets) but these are not described here.

Analytical steps

The following methods are not comprehensive and do not comprise all necessary or possible techniques for performing IEAs. In any case a thorough data exploration needs to take place beforehand to decide which transformations are necessary and which are the most appropriate tools to analyse the time series. We will not go into detail here but rather present examples how the time series can be graphically illustrated and how to accomplish some of the statistical analyses.

1. Temporal Anomalies

The temporal dynamics of variables can be presented in various ways. Here we show the R-code for a bar plot (Figure 1) using deviations from the mean ($y' = x_{ij} - \bar{x}_i$).

```
# import data
Originaldata <- read.table("CBSrawData.txt", header = TRUE)
# check the data
str(Originaldata)

# data preparation:
# create a dataframe for the anomaly values
AnomData <- Originaldata
# save the dimensions (number of rows[1] and columns[2]) in a vector
n <- dim(AnomData)
# loop to convert actual values into anomaly values (incl. missing
# values in the average- and anomaly-calculation); the loop starts with the
# second column to exclude "Year" (no variable) and ends with the last
# column (defined by the second element of the vector n (n[2]) )
for (i in 2:n[2]) {
      AnomData[,i] <- AnomData[,i] - mean(AnomData[,i], na.rm = TRUE)}

# plot the data (here for the copepod species Acartia, in spring.
# $Acartia_Spr is the respective column title):
# to set the limit of the y-axis in the barplot (ylim)
y.low <- min(AnomData$Acartia_Spr, na.rm = TRUE) - 2
y.upp <- max(AnomData$Acartia_Spr, na.rm = TRUE) + 2
barplot(AnomData$Acartia_Spr, col = "black", space = 0.7,
      ylim = c(y.low, y.upp), axis.lty = "solid",
      names.arg = AnomData$Year, axisnames = TRUE,
      ylab = "Abundance anomalies (N/m³)",
      main = "Acartia in Spring")
box(lty = "solid", col = "black")
# creates a horizontal line at y = 0
abline(a = NULL, b = NULL, h = 0, v = NULL, reg = NULL,
       coef = NULL, untf = FALSE)
```

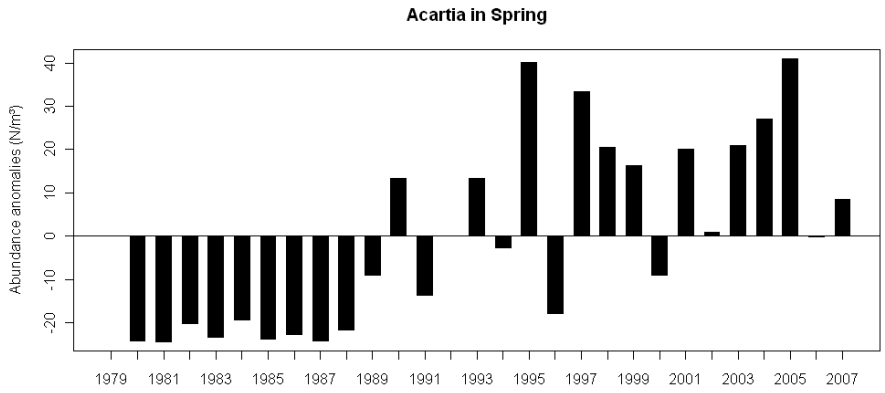

Figure 1: A bar plot showing anomalies in spring abundances of Acartia in the period 1980 – 2007.

2. Traffic light plot

This plotting tool is especially suitable to visualise trends and changes considering all variables in the integrated assessment. In the following example, each variable is divided into quintiles and to each quintile a specific colour is assigned (from green – low values, to red – high values). Variables are sorted in ascending order according to their standardised variable averages over the first five data points of the time series. An alternative is a sorting procedure according to the results of the subsequently performed PCA: Then variables are sorted with respect to their loadings on the first principal axis (Figure 2). However, the number of quantiles as well as the sorting procedure is not a convention and can be changed according to the personal needs.
*For this plot data the original data file needs to be rearranged, in a way that the column with the years (ascending order) represents the **last** column. The respective datasheet refers to "CBSrawData2.txt", or "Originaldata2" respectively.*

```
# import data
Originaldata2 <- read.table("CBSrawData2.txt", header = TRUE)
# check the data
str(Originaldata2)

# data preparation:
# save the dimensions (number of rows[1] and columns[2] ) in a vector
n <- dim(Originaldata2)
# to get names for the plot axes except the last column (year)
# the y-axis (without the last column (years) )
Variablenames <-names(Originaldata2[1:n[2]-1])
# the x-axis
Years <- as.vector(Originaldata2[,n[2]])

# generates a matrix containing the quintile values for each variable
# (exluding the years by setting the column number and loop end to n[2] -1)
Quintiles <- matrix(nrow = 5, ncol = n[2]-1, byrow = FALSE)
for (i in 1:(n[2]-1)) {
      Quintiles[, i] <- quantile(Originaldata2[,i],
            probs = c(0.2, 0.4, 0.6, 0.8, 1.0), na.rm = TRUE)}

# generation of a dataframe where original values are converted
# into code 1 to 5 (depending on the quintile) using two loops:
# first create new dataframe based on the original dataframe
CodedData <- Originaldata2
# then convert the actual values into codes using a loop:
# the first (outer) loop repeats process for every variable,
# the second (inner) loop repeats process for every element within
# a variable
for(j in seq(along = CodedData[1:(n[2]-1)])) {
for(i in seq(along = CodedData[[j]])) {
```

```
      if(is.na(CodedData[[j]][i]) == TRUE) {
            is.na(CodedData[[j]][i])<- TRUE} else
      if(CodedData[[j]][i] < Quintiles[1,j]) {
            CodedData[[j]][i]<-1} else
      if((CodedData[[j]][i] >= Quintiles[1,j]) &&
            (CodedData[[j]][i] < Quintiles[2,j])) {
            CodedData[[j]][i]<-2} else
      if((CodedData[[j]][i] >= Quintiles[2,j]) &&
            (CodedData[[j]][i] < Quintiles[3,j])) {
            CodedData[[j]][i]<-3} else
      if((CodedData[[j]][i] >= Quintiles[3,j]) &&
            (CodedData[[j]][i] < Quintiles[4,j])) {
            CodedData[[j]][i]<-4} else
      if(CodedData[[j]][i] >= Quintiles[4,j]) {
            CodedData[[j]][i]<-5} }}
# check if each column has the same counts of code values 1-5
CodedData

# plot preparation:
# to order variables in the plot in accordance to the standardised variable
# averages over the first five data points (excluding the years with
# n[2]-1):  first create vector
Avg5year <- vector(length = n[2]-1)
# then fill the vector with the standardised variable averages over the
# first five data points (mean of first 5 years - mean of full time series/
# standard deviation of full time series)
for (i in 1:(n[2]-1)) {
      Avg5year[i] <- (mean(Originaldata2[c(1:5), i], na.rm = TRUE) -
            mean(Originaldata2[, i], na.rm = TRUE)) /
            sd(Originaldata2[, i], na.rm = TRUE) }
# finally, order the variable and create an index vector
OrdVar <- order(Avg5year)

# to sort coded dataframe in accordance to the 5-year standardised
# average, use the vector OrdVar as an index
CodedData.sorted <- CodedData[,c(1:n[2]-1)][OrdVar]

# convert dataframe into a matrix (necessary for an image plot)
CodedData.mat <- as.matrix(CodedData.sorted)

# image plot:
x <- c(1:n[1])
y <- c(1:n[2]-1)
op <- par(mar = c(4, 5, 3, 6), oma = c(0.5,2,0,0), xpd = TRUE)
image(x, y, z = CodedData.mat, zlim = c(1,5),
      col = c("red", "gold", "yellow", "yellowgreen", "green"),
      axes = FALSE, xlab = "Years", ylab = "")
axis(1, at = seq(1, n[1], by = 1), tick = FALSE, labels = Years,
      cex.axis = 0.7, las = 3)
axis(2, at = seq(1, n[2]-1, by = 1), tick = FALSE,
      labels = Variablenames[OrdVar],
      cex.axis = 0.5, las = 1, padj = 1)
box()
title(main = "Traffic Light Plot", font.main = 4)
mtext("Variables (sorted by the 5-year standardised average)",
      outer = TRUE, side = 2)
# if you have a lower version than R2.10, remove the border argument
legend(x = max(x)+1, y = max(y), legend = c("< 0.2 quintile",
      "< 0.4 quintile", "< 0.6 quintile", "< 0.8 quintile",
      "> 0.8 quintile", "missing value"),
      fill = c("green", "yellowgreen", "yellow", "gold", "red", "white"),
      border =  c("green", "yellowgreen", "yellow", "gold", "red", "grey"),
      cex = 0.6, bty = "n")
par(op)
```

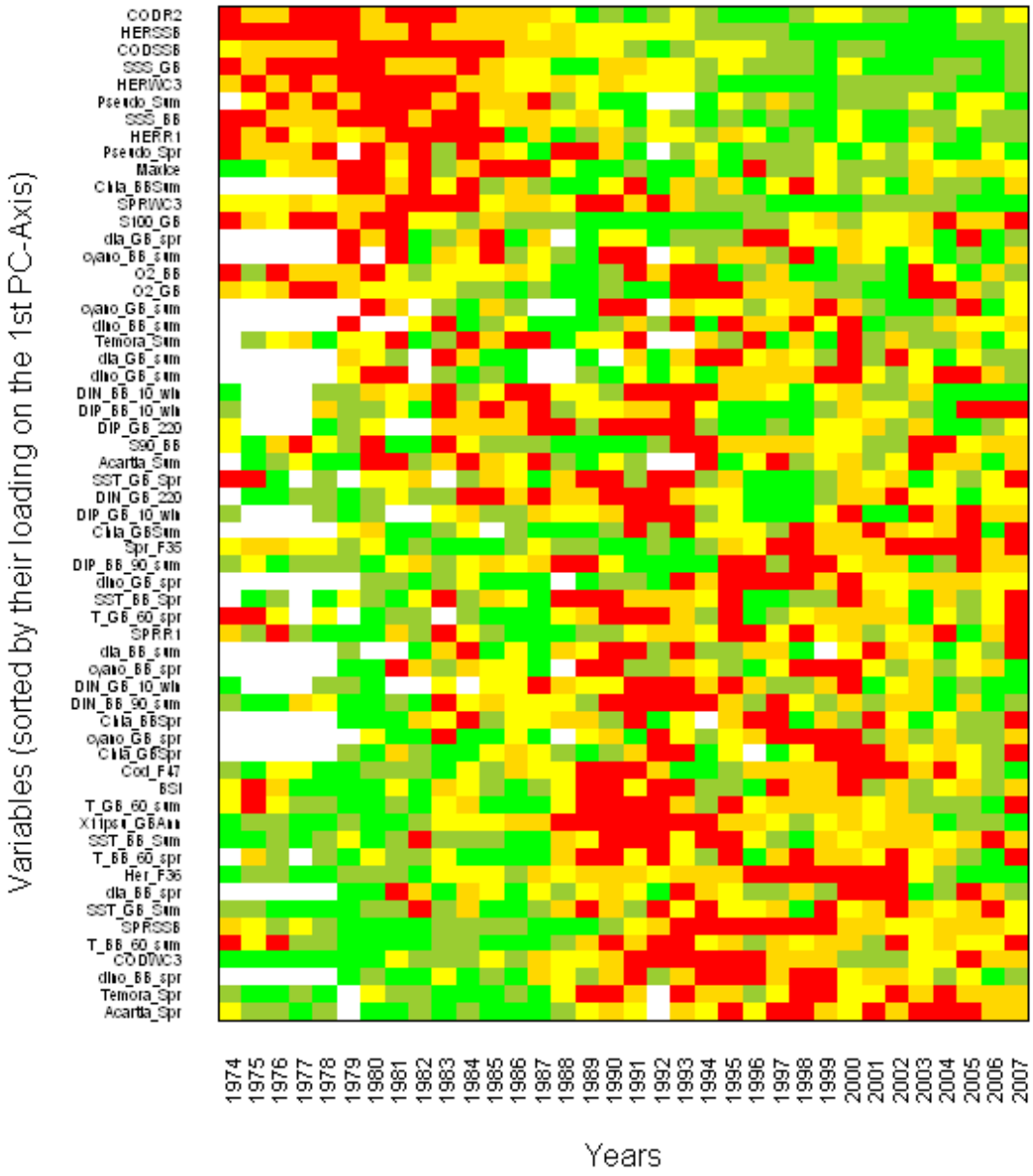

Figure 2: Traffic-light plot of the temporal development of Central Baltic Sea time-series. Variables are transformed to quintiles, colour coded (green = low values; red = high values), and sorted in numerically descending order according to their loadings on the first principal component.

For the following analyses and plots, we recommend to replace missing values by four-year averages. Further, some variables need to be previously transformed, e.g. by logarithmic transformations. The respective datasheet refers to "Trans_PCA.txt", or "Data.transf" respectively.

3. Principal Component Analysis (PCA) based on the correlation matrix ("normalised PCA")
This analysis can be performed for the full dataset, including abiotic forcing and biological response variables, as well as for data subsets, e.g. separately for biotic and abiotic variables. The code describes how to perform the analysis, get the output in terms of Eigenvalues, Scores and Loadings, and a couple of plotting tools how to present the results graphically.

```
# import data
Data.transf <- read.table("Trans_PCA.txt", header = TRUE)
# check the data
str(Data.transf)
# save the dimensions (number of rows[1] and columns[2]) in a vector
n <- dim(Data.transf)

# load package
library(vegan)

# PC analysis:
# exclude the years in PCA by using not the first column; to use the
# correlation matrix set the scale = TRUE)
PCA <- rda(Data.transf[, c(2:n[2])], scale = TRUE)
# to see the structure of the PCA result
str(PCA)

# output:
# to obtain the Eigenvalues
PCA.eigen <- PCA$CA$eig
PCA.eigen
# the summary gives the Eigenvalues, explained variance, cumulative
# variance, loadings (species scores) and scores (site scores, which are
# scaled by a constant given in the output, here 6.375328)
PCA.sum <- summary(PCA)
PCA.sum


# plots:

# create first a character vector containing the years for the score labels
years.char <- paste(Data.transf$Year, sep = "")

# biplot(with loadings and scores); to have more control of the plot,
# plotting items will be added separately. Variable names and scores
# will be not displayed by the setting: type = c("points","none"),
# the choices argument determines the PC axes (1:2 chooses PC1/PC2),
# the argument scaling = 3 scales species and sites symmetrically by the
# square root of the eigenvalues
biplot(PCA, choices = 1:2, scaling = 3,
      display = c("sites", "species"), type= c("points","none"),
      col = c("darkgreen"))
# add the variable names using same scaling value
text(PCA, display = "species", col = "darkgreen", cex = 0.5,
      scaling = 3)
# add now the scores with the year label using the same scaling value
text(PCA, display = "sites", labels = years.char, cex = 0.7, scaling = 3)
```

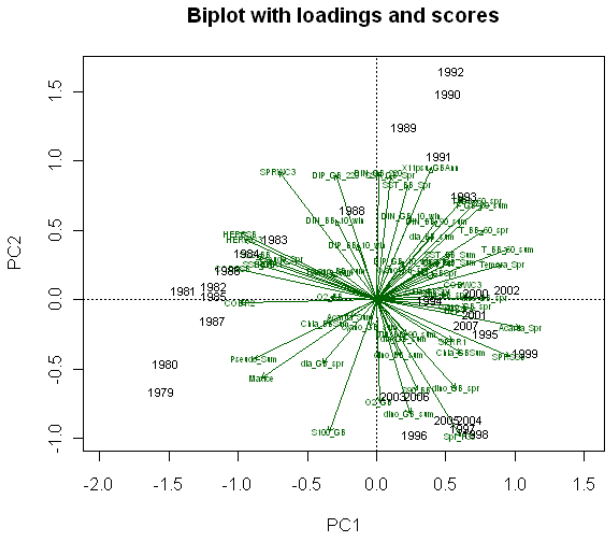

Figure 3: A correlation biplot of a standardised Principal Component Analysis (PCA) displaying the loadings and scores, labelled here with the year, on the first 2 PC axes.

```
# biplot with loadings and time trajectory in the back
biplot(PCA, choices = 1:2, scaling = 3,
      display = c("sites", "species"), type= c("points","none"),
      col = c("darkgreen"), main = "Biplot with time trajectory")
# add now the scores with the year label using the same scaling value
text(PCA, display = "sites", labels = years.char, cex = 0.7,
      col = "grey", scaling = 3)
# for the lines the coordinates of the scaled PC1/2 scores have to be
obtained
scores.sc <- scores(PCA, scaling = 3)
scores.sc
# to connect the sites in an ascending order of the year
lines(x = scores.sc$sites[, 1][order(Data.transf$Year)],
      y = scores.sc$sites[, 2][order(Data.transf$Year)],
      lty = 1, col = "grey")
# add the variable names using same scaling value
text(PCA, display = "species", col = "darkgreen", cex = 0.5,
      scaling = 3)
```

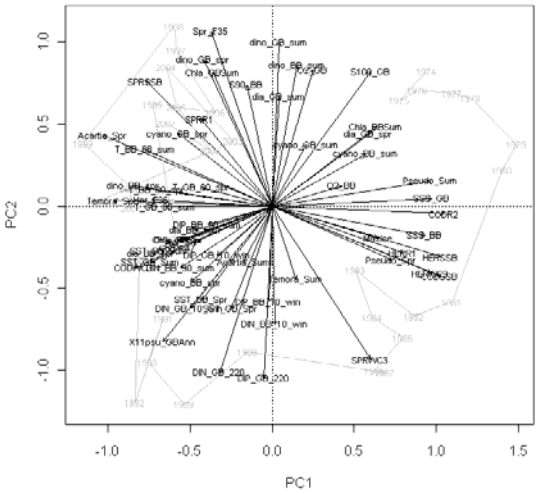

Figure 4: Results of the standardized principal component analysis using all 59 variables assembled for the central Baltic Sea showing the variable loadings on the first factorial plane (for orientation: time trajectory of Figure 5, right panel in light grey).

```
# Time trajectory plot with the unscaled scores; in the plot, scores and
# loadings are not displayed (by setting: type = c("none","none") )
biplot(PCA, choices = 1:2, scaling = 0,
      display = c("sites", "species"), type= "n",
      main = "Time trajectory plot")
text(PCA, display = "sites", labels = years.char, cex = 0.7,
      scaling = 0)
# for the lines the coordinates of the unscaled PC1/2 scores have to be
# obtained
scores.unsc <- scores(PCA, scaling = 0)
# connect the sites in an ascending order of the year
lines(x = scores.unsc$sites[, 1][order(Data.transf$Year)],
      y = scores.unsc$sites[, 2][order(Data.transf$Year)],
      lty = 1, col = "blue")


# Time series plot of PC1 and PC2 scores using rda results:
# get first unscaled PC1/PC2 site scores
scores.unsc <- scores(PCA, scaling = 0)
# get the min/max value for the limit of the y-axis
y.low <- min(scores.unsc$sites)
y.upp <- max(scores.unsc$sites)
plot(Data.transf$Year, scores.unsc$sites[, 1], pch = 16,
      ylab = "PC scores", xlab = "Years", axes = FALSE,
      xlim = c(1979, 2007), ylim = c(y.low, y.upp), cex = 0.7, cex.lab =
            0.9,
      main = "Time-trajectory of PC scores", cex.main = 0.9)
axis(2, cex.axis = 0.7, las = 1)
axis(1, at = seq(1980, 2005, by = 5), label = seq(1980, 2005, by = 5),
      cex.axis = 0.7)
box(col = "black")
abline(h = 0, lty = "dotted")
lines(Data.transf$Year, scores.unsc$sites[, 1], col = "black",
      lwd = 0.8)
points(Data.transf$Year, scores.unsc$sites[, 2],
      pch = 1, cex = 0.7, col = "black", type = "b", lty = 3)
legend("bottomright", legend = c("PC1 scores", "PC2 scores"),
      text.col = c("black", "black"), cex = 0.7,
      col = c("black", "black"),
      pch = c(16, 1), lty = c(1, 3),
      pt.cex = 0.7, bty = "n")
```

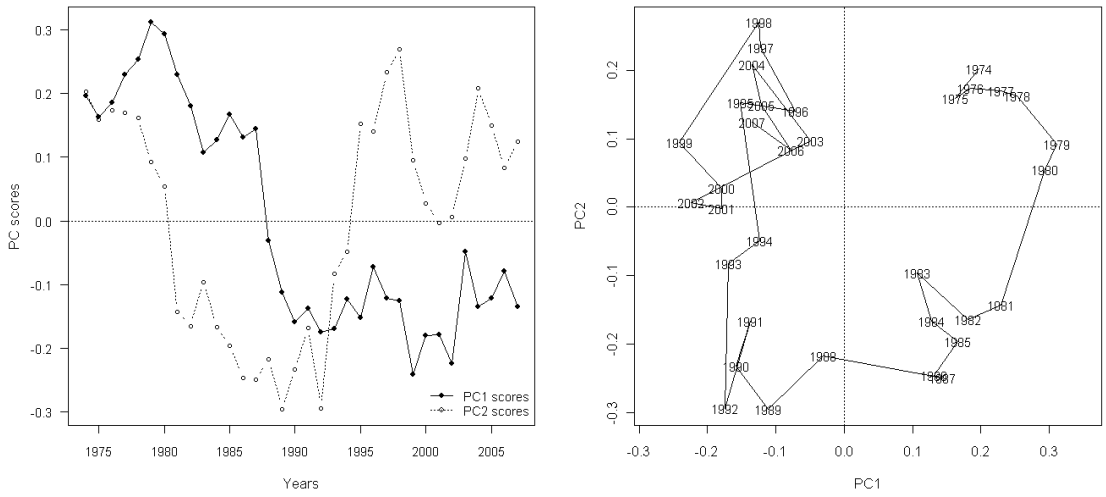

Figure 5: Results of the standardized principal component analysis using all 59 variables assembled for the Central Baltic Sea showing PC1 (black circles) and PC2 scores (white circles) against time, and the time trajectory on the first factorial plane. PC1 = 27.0 %, PC2 = 14.1 % explained variance (from left to right).

4. „Constrained Clustering“: Identifying breakpoints in multivariate data

Chronological Clustering is not yet available as function in R. So far it is implemented in the freeware “R-package”, which is however only available for MacOS, and in the commercial software “BRODGAR” (table 1). A similar, though not fully complementary because hierarchical method, is programmed in the package “rioja” for R (Functions for the analysis of quaternary science data). As agglomeration method it is recommended to avoid single linkage algorithms like “conslink” but rather applying the other option “coniss” using incremental sum of squares (Grimm, 1987). Originally this algorithm was developed for merging only stratigraphically adjacent clusters. This approach is similar as the one used for chronological clustering, although the latter method is preferred because other parameters can be defined and the statistical significance of breakpoints can be assessed.
In order to use constrained clustering in combination with PCA, data should be treated accordingly, i.e. in terms of missing values, variable transformations, standardisation of variables as well as for the respective distance measure. In normalised PCAs, data are z-transformed and the Euclidean distance between observations is preserved in the plot. Therefore, a Euclidean distance matrix represents the input data for the analysis. Finally, a broken stick model is used to determine the number of significant groups in the analysis: As long as the output value for a specific number of groups is larger than the broken stick value the groups are termed significantly different.

```
# import data
Data.stand <- read.table("Trans_PCA.txt", header = TRUE)
# check the data
str(Data.stand)

# save the dimensions (number of rows[1] and columns[2]) in a vector
n <- dim(Data.stand)

# load packages
library(vegan)
library(rioja)

# if data are not previously transformed (as in this case), the following
# code using the vegan package will standardise the data (necessary for the
# calculation of the Euclidean distance matrix from standardised data:
# decostand(x, method=standardize, MARGIN=2, na.rm=T)

# calculation of a Euclidean distance matrix from standardised data
# (stats package), excluding the first column with the years
Data.dist <- dist(Data.stand[, c(2:n[2])], "euclidean",
      diag = TRUE, upper = TRUE)


# Constrained hierarchical Clustering as described in "rioja" package:
# Coniss = constrained incremental sum of squares clustering
CC <- chclust(Data.dist, method="coniss")

# plot outcome, use the par function to display both plots together:
par(mfrow = c(1, 2))
# cluster-plot with labels
years.char <- paste(Data.stand$Year, sep = "")
plot.chclust(CC,labels= years.char, cex = 0.7,
      main = "Coniss  - Cluster plot")
# broken stick plot
bstick.chclust(CC)
title("Coniss - Broken stick plot")
```

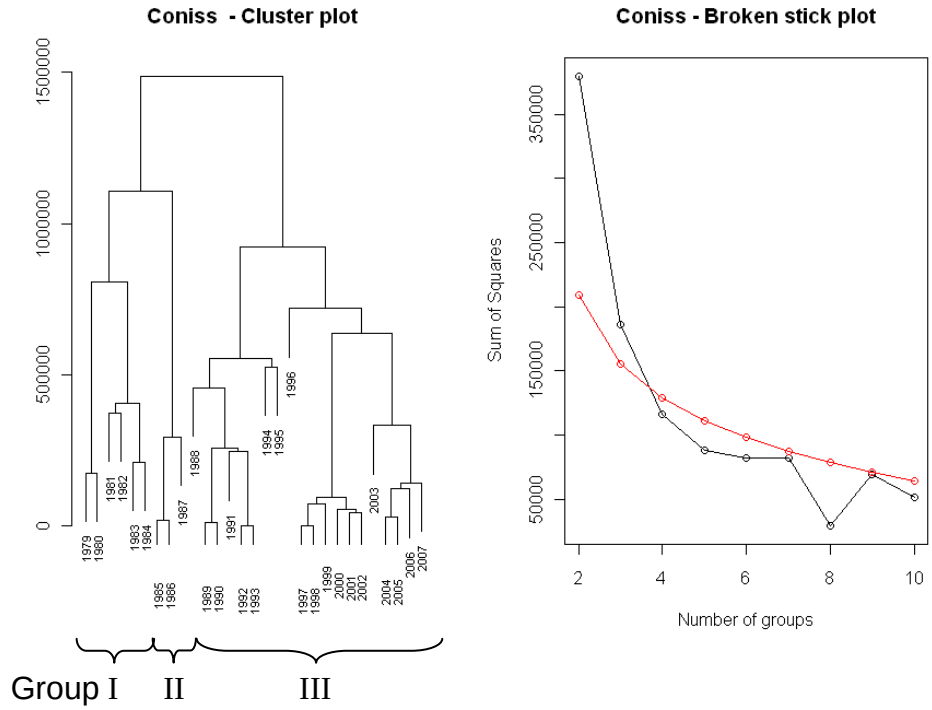

Figure 6: A: Cluster plot from the Constrained Clustering analysis using the "coniss" option, where row numbers were replaced by the year labels. B: A plot if a broken stick model to determine the number of significant groups in the cluster analysis (here: three groups are significant because only for two and three groups the value of the black line is above the critical value represented by the red line).

5. Identification of change-points in single variables and a combination of variables:
A series of methods exist, e.g. two-phase regression models (Solow, 1987) or methods using least squares estimation (Lavielle and Moulines, 2000). In R we refer to the package "strucchange", where tests for structural change in linear regression models from the generalized fluctuation test as well as from the F test framework are implemented. We also used a freeware called "STARS", where the code is written in Visual Basic for Application (Excel) and can be downloaded from http://www.beringclimate.noaa.gov/regimes/.
STARS is based on a sequential t-test that can signal a possibility of a regime shift in real time (Rodionov, 2004).
For the time series analysis described here, STARS can be used in three different ways:

1. each single time series can be analysed for sudden changes separately;
2. all variables can be analysed at once. In the "Summary" spreadsheet Regime Shift Indices (RSI) are summed up for each point in time (bar plots of cumulative RSIs);
3. STARS can be applied to ordination scores of the first and, in case of a low amount of explained variance, additionally of the following PC-axes (resulting from the previously performed PC analysis).

The following parameters need to be defined before starting the analysis (for details see Rodionov, 2004):

- Significance level α (e.g. = 0.05),
- cut-off length (e.g. = 8 years),
- Huber's weight parameter (e.g. = 2).

It is recommended to use different parameter values e.g. for the cut-off length, and see how the output behaves. Depending on the overall length of the time series and the assumed length of the amplitude, other parameters than the ones mentioned above may be appropriate.

Christian Möllmann
Rabea Diekmann