Common Patterns

Kelly Gallacher and Claire Miller and Marian Scott

2016-04-28

Contents

1 Set-up

1.1 Loading the package

The stpca package can be loaded as follows:

```
library(stpca)
```

2 Introduction

Monitoring networks consist of multiple layers of spatial and temporal correlation inducing redundancies in the information provided. For river water quality monitoring in particular, flow connected sites may likely provide similar information. This tutorial describes a novel approach to PCA to investigate reducing dimensionality for spatiotemporal flow connected network data.

One way to re-design the monitoring network is to identify dominant spatial and temporal patterns to help regulatory agencies better focus monitoring in priority areas. These might include increasing monitoring in spatial regions where observations are unstable over time, or reducing monitoring in areas where several monitoring sites exhibit the same temporal pattern.

This tutorial explains how to use the R functions developed as part of Kelly Gallacher's PhD (**???**) and the EPSRC SECURE network project 'Statistical software to identify spatiotemporal patterns and coherence over river networks' for identifying common patterns in river network data, a joint project between The University of Glasgow, The Environment Agency and The Scottish Environment Protection Agency (**???**). The specific reference for this work is the paper 'Flow directed PCA for Monitoring networks'.

This tutorial will use demonstration data files to illustrate:

- the data formats required
- spatial and temporal weights that can be incorporated into the data analysis
- missing data imputation, where required
- performing principal components analysis for river network data
- displaying and visualising the results
- interpreting the results.

There are three steps to the analyses described in this tutorial

1. **Pre-processing the data**:
    - `createWeightS()` can be used to create a matrix of spatial weights
    - `creageWeightT()` can be used to create a matrix of temporal weights
    - `completeData()` can be used to replace missing values in the dataframe with values estimated from the rest of the data
2. **Identifying common spatial/temporal patterns**
    - `stpca()` can be used to perform prinicpal components analysis, adjusted for spatial and/or temporal correlation
3. **Visualizing the results**
    - `plot_stpca()` provides several options for plotting the results from `stpca()`

If you do not wish to adjust for spatial and/or temporal correlation, and if your dataframe has no missing values, then you can skip Step 1 and go straight to Step 2.

This tutorial will proceed by first giving some background to the statistical methods used in the stpca package, followed by a description of the data formats required. Next, a detailed walkthrough of the functions in Steps 1-3 is given, with some suggestions as to how the results might be interpreted. Finally, a Troubleshooting section describes solutions to potential problems that might occur when using these functions.

Sections 2.2 - 2.4 contain technical details describing (weighted) principal components analysis, the calculation of spatial weights (using an SSN object), and K-fold cross validation. This has been included as a reference for the interested reader and explains some terminology used throughout the tutorial but you can still implement the analyses described in this tutorial without reading this in detail.

2.1 Background

One approach to identify dominant spatial and temporal patterns is to use principal components analysis (PCA). PCA is a dimensionality reduction technique where the aim is to replace the original variables (which are correlated with one another) with a smaller number of uncorrelated *new* variables, called principal components (PC's), or scores. These *new* variables are weighted linear combinations of the original variables, which explain the directions of maximum variation in the data set.

For example, for river network data, a nutrient could be monitored regularly over a number of sites and time points. The level of the nutrient between sites at subsequent time points will be related, this results in spatial and temporal correlation over the network. This correlation could be driven by catchment or environmental factors resulting in spatial and/or temporal trend. Within a river network nutrient levels between sites could be related because of local catchments factors i.e. euclidean distance between sites, or because of upstream nutrient levels i.e. river distance between sites and direction of flow.

This tutorial explains how to perform principal components analysis for such data and hence investigate the presence of dominant spatial and temporal patterns in river water quality data. Specifically, weight matrices reflecting spatial and temporal autocorrelation are constructed and incorporated in the analysis to investigate spatial and temporal patterns that remain in the data after adjusting for known structure in the data. In particular, this tutorial illustrates how a matrix of spatial weights can be constructed to reflect the direction of water flow and strength of relationship between monitoring sites connected by river flow.

2.1.1 Principal Components Analysis

Prinicpal components analysis (PCA) is performed on a data matrix where rows are observations and columns are variables. For example, each row could represent a different monitoring site and each column could represent different water quality parameters measured (at the same time point) for each site. A variation of this is when you have measurements of a single water quality parameter at several monitoring istes and over several time points. PCA can be applied to such spatiotemporal data and the user can decide whether columns (variables) should be monitoring sites or time points. Richman (1986) refers to these different approaches to PCA as *"modes"* and each mode (where a mode is any combination of two from: site, time, water quality parameter) can be used to investigate different aspects of the data.

Specifically, in this R package, two *"modes"* of PCA are adapted: T-mode and S-mode. These two modes have observations of a single water quality parameter measured at several sites and over several time points. The particular mode used depends on the orientation of the dataframe X which could have columns containing monitoring sites (S-mode) or time points (T-mode).

2.1.2 T mode spatial pattern 파악 위함

The variables are the different time points and hence the columns of the data matrix X for the PCA are separate time points. Rows are therefore monitoring sites.

T-mode PCA aims to find spatial patterns in the data and to identify at what time points these spatial patterns occur. The presence of more than one dominant spatial pattern suggests a change in the spatial pattern over time. Alternatively, the identification of a single dominant spatial pattern would suggest that the spatial distribution of the variable of interest has remained stable over time. A stable spatial pattern over time indicates areas where monitoring could be reduced.

2.1.3 S mode

temporal pattern 파악 위함

The variables are the different sites and hence the columns of the data matrix X for the PCA are separate sites. The rows are therfore time points.

S-mode PCA aims to estimate dominant temporal patterns in the data and to provide an indication of which sites exhibit similar temporal patterns. If common temporal patterns can be identified then this suggests there are groups of monitoring sites behaving in a similar way over time. It might also indicate redundancy in the monitoring network and provide evidence to reduce the number of monitoring sites.

2.2 PCA and back transforming

PCA is performed in this package using singular value decomposition (SVD) of the column mean centered $n \times p$ data matrix $\mathbf{X}$ such that:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

where $\mathbf{U}$ and V are called the left and right singular vectors of $\mathbf{X}$, and $\mathbf{D}$ is a diagonal matrix containing the singular values (these are the square roots of the eigenvalues if PCA is performed using eigendecomposition of the covariance/correlation matrix of $\mathbf{X}$).

Each prinicpal component (PC, also known as scores), is a weighted linear combination of the original variables and the weights used to calculate the PC's are called loadings. In SVD the loadings are found in the columns of $\mathbf{V}$ while the PC's/scores can be calculated as either $\mathbf{UD}$ or $\mathbf{XV}$. In SVD, k PC's will be calculated and k is $\min(n, p)$.

$\mathbf{X}$ can also be written as

$$\mathbf{X} = \mathbf{X}\mathbf{V}_{1:k}\mathbf{V}_{1:k}^\top = \mathbf{X}\mathbf{V}_{k+1:p}\mathbf{V}_{k+11:p}^\top$$

where $\mathbf{V}_{1:k}$ means the first k columns of $\mathbf{V}$ and $\mathbf{V}_{k+1:p}$ are the last $p - k$ columns of $\mathbf{V}$.

PCA can be adjusted for spatial and temporal correlation using appropriate row and column weights. A $p \times p$ column weight matrix $\mathbf{\Omega}$ and $n \times n$ row weight matrix $\mathbf{\Phi}$ can be constructed so that PCA is applied to

$$\tilde{\mathbf{X}} = \mathbf{\Phi}\mathbf{X}\mathbf{\Omega} = \tilde{\mathbf{U}}\tilde{\mathbf{D}}\tilde{\mathbf{V}}^\top$$

.

However, the loadings and PC's are now related to $\tilde{\mathbf{X}}$ rather than $\mathbf{X}$. Loadings and PC's can be calculated for $\mathbf{X}$ using a suitable back transformation, sometimes called the "general solution".

First consider

$$\mathbf{\Phi}\mathbf{X}\mathbf{\Omega} = \mathbf{\Phi}\mathbf{X}\mathbf{\Omega}\tilde{\mathbf{V}}_{1:k}\tilde{\mathbf{V}}_{1:k}^\top + \mathbf{\Phi}\mathbf{X}\mathbf{\Omega}\tilde{\mathbf{V}}_{k+1:p}\tilde{\mathbf{V}}_{k+1:p}^\top.$$

Pre-multiplying all of the terms in this equation by $\mathbf{\Phi}^{-1}$ and post-multiplying by $\mathbf{\Omega}^{-1}$ gives

$$\mathbf{X} = \mathbf{X}\boldsymbol{\Omega}\tilde{\mathbf{V}}_{1:k}\tilde{\mathbf{V}}_{1:k}^{\top}\boldsymbol{\Omega}^{-1} + \mathbf{X}\boldsymbol{\Omega}\tilde{\mathbf{V}}_{k+1:p}\tilde{\mathbf{V}}_{k+1:p}^{\top}\boldsymbol{\Omega}^{-1}.$$

The PC's of $\mathbf{X}$ are therefore $\mathbf{X}\boldsymbol{\Omega}\tilde{\mathbf{V}}$ and the loadings are $\boldsymbol{\Omega}_{-1\top}\tilde{\mathbf{V}}$.

2.3 Spatial and temporal weights

Spatial and temporal weights can be calculated that describe correlation in the data. The `stpca()` function adjusts standard PCA using the inverse square root of the spatial and temporal weights matrices to allow the user to identify interesting spatial and temporal features previously masked by spatial or temporal correlation.

2.3.1 Spatial weights for stream networks

Erin E Peterson and ver Hoef (2014) shows how an object containing information about the structure of a river network can be created, for use in R, using ArcGIS software. This object is called a SpatialStreamNetwork (SSN) object and once produced, this SSN object can be imported into R and the SSN package can be used to fit (geo)statistical models to data collected on river networks (see the vignette "Reducing Networks" for a brief description of geostatistical models). Some of the information stored within the SSN object can be used to calculate spatial weights, reflecting spatial correlation related to flow direction and strength of connectedness in a river monitoring network.

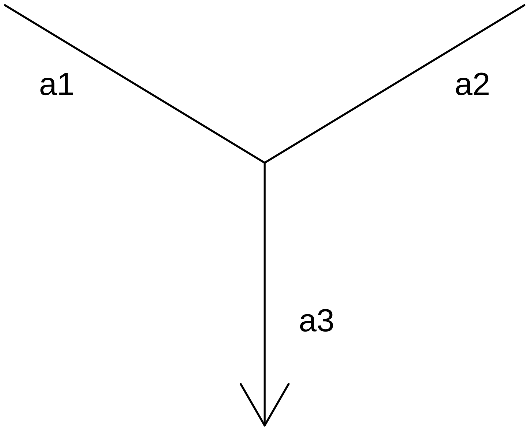
For example, in geostatistics it is usually assumed that obsevations close to each other in space will be more strongly correlated than observations far apart in space. For observations recorded on river networks however this is not always true since an observation recorded on a small tributary could have a very different value (and only be weakly correlated) with a second observation recorded downstream at a location on the main stem of the river. Although these two observations are connected by the flow of the river, the strength of connection can be described as weak.

E. E. Peterson and ver Hoef (2010) describe the calculation of spatial weights reflecting the correlation related to flow direction and strength of connectedness in a river network. The information required for this is contained within an SSN object. The weights can be based on flow, or proxy values for flow such as upstream area (km^2) (the sum of area of land draining to a stream segment and area of land draining to all upstream stream segments).

For weighted PCA, the asymmetric matrix of spatial weights can be calculated using the following steps. The calculations follow the steps described in detail E. E. Peterson and ver Hoef (2010) with the exception that the matrix is not forced to symmetry in the final step.

The rest of this section will describe how the weights are calculated. This is intended to be a brief description and you should consult E. E. Peterson and ver Hoef (2010) or Erin E Peterson and ver Hoef (2014) for full details. Weights are calculated for segments and a monitoring site is allocated the weight of the segment on which it is located.

Here is an example of a very simple river network with three stream segments. For any size of river network, it is assumed that moving upstream, an individual segment can only split into two branches and the point where two branches join is the confluence point (i.e. you cannot have more than two segments joining at a confluence point). There is no limit on the number of splits that can occur. The area of land (km^2) that drains to each segment is a_1, a_2, and a_3. The total land that drains to segement 1 is *a1* but the total land that drains to segment 3 is $a_1 + a_2 + a_3$.



To calculate spatial weights, first calculate the proportional influence (PI) of each pair of segments joining at a confluence. In this example, $\text{PI}_1 = \frac{a_1}{a_1+a_2}$ and $\text{PI}_2 = \frac{a_2}{a_1+a_2}$. $\text{PI}_3 = 1$ since this segment contains the outlet of the river (shown by the arrowhead). PI can only take values between 0 and 1.

Once the PI has been calculated for all stream segments, the *additive function value* (AFV) can be calculated. The AFV is calculated for each segment individually by calculating the product of the PI values lying between the segment containing the outlet and each individual segment. This means that if your river network consists of 20 segments, you will calculate 20 AFV's. In this example, $\text{AFV}_{1,3} = \text{PI}_1 \times \text{PI}_3$, $\text{AFV}_{2,3} = \text{PI}_2 \times \text{PI}_3$, and $\text{AFV}_3 = 1$.

Now that the AFV has been calculated for each segment, weights reflecting spatial correlation between two flow connected monitoring sites can be calculated. Weights are calculated as

$$\sqrt{\frac{\text{AFV}_u}{\text{AFV}_d}}$$

where AFV_u represents the upstream site in the flow connected pair of sites and AFV_d is the downstream site in the pair. Note that

$$\sqrt{\frac{\text{AFV}_d}{\text{AFV}_u}} = 0$$

2.4 K-fold cross validation

K-fold cross validation is used in the `completeData()` function to impute missing values and you should refer to the help documentation for the `estim_ncpPCA()` function in the missMDA package in R for full details. A brief description of K-fold cross validation is given here.

K-fold cross validation is used in statistical modelling to choose the 'best' value from a range of possible values. In the `completeData()` function, the `estim_ncpPCA()` function from the missMDA package is called to estimate the number of principal components (k) that should be used to impute missing values in the spatiotemporal dataframe.

In general, K-fold cross validation works by randomly removing a proportion of the data and fitting a model to the remaining data. The model is then used to estimate the withheld data and the mean squared error is calculated, where error is the difference between the estimated value and the true value. This process is repeated, with a proportion of the data randomly removed a specifed number of times and the sum is calculated. This procedure is performed in the `completeData()` function for several values of k, and the k with the smallest mean squared error is then used to impute missing values.

Once the missing values have been replaced with suitable estimated values, PCA can be performed.

3 Data format

Three types of data are required to perform all of the analyses described in this tutorial.

- A dataframe containing observations of a single water quality parameter recorded for several monitoring sites at several, regularly spaced, time intervals. These time intervals could represent daily/weekly/monthly/annual observations/averages, for example. The dataframe could be in the form of a .csv file or any other suitable format so that once it is read into R, it is of `"data.frame"` class. Columns should represent monitoring sites and column names should uniquely identify each monitoring site. Rows should be time points in chronological order and row names can be used to identify the dates represented by rows. Missing values should be represented as `NA`.
- An object of `"SpatialStreamNetwork"` class (SSN object) is required if you wish to use the `createWeightS()` function to create spatial weights reflecting flow direction and strength of connectivity in the river network. You can supply your own matrix of spatial weights to use in the `stpca()` function and this would not require an SSN object, but `createWeightS()` is designed to calculate spatial weights using information contained within an SSN object.
- A shapefile with .shp extension is also required to produce some of the plots in `plot_stpca()`.

A demonstration data file will be used for this tutorial, which has 30 monitoring sites over a river network (made up of 80 stream segments), which have each had concentrations of a water quality determinad (say) simulated for 25 time points. Full details describing the demonstration data generation are provided in the vignette "Creating demo data".

An example of the suitable format for dataframe containing spatiotemporal observationsand is given below.

```r
# load the demo dataframe containing spatiotemporal observations and
# inspect the first few rows and columns
data(demoY)
head(demoY)[,1:7]
```

```
##          1         2        3         4        5        6         7
## 1 83.66534  90.54943 87.56660  94.74550 87.72304 84.89257  90.84081
## 2 84.26214  95.17072 71.76019  93.64118 91.07545 75.92171  95.23313
## 3 82.70443 100.67188 82.30221  99.37059 69.37519 68.17076 101.01647
## 4 77.43921 100.51986 79.44762 103.12816 69.32245 74.34863 104.73041
## 5 77.12826 102.73167 69.98795 107.31950 70.81690 80.65534 103.61155
## 6 64.41192 111.97123 66.04143 111.69628 73.59843 67.04863 106.72822
```

Each column in the dataframe is a site (monitoring location from 1 . . . 30), and each row contains a value of the response at a specific time point (from 1 . . . 25). (*Note: only the first 7 columns are displayed*). Time points are regularly spaced and in chronological order so 1 . . . 25 could represent 25 monthly averages for the water quality determindand.

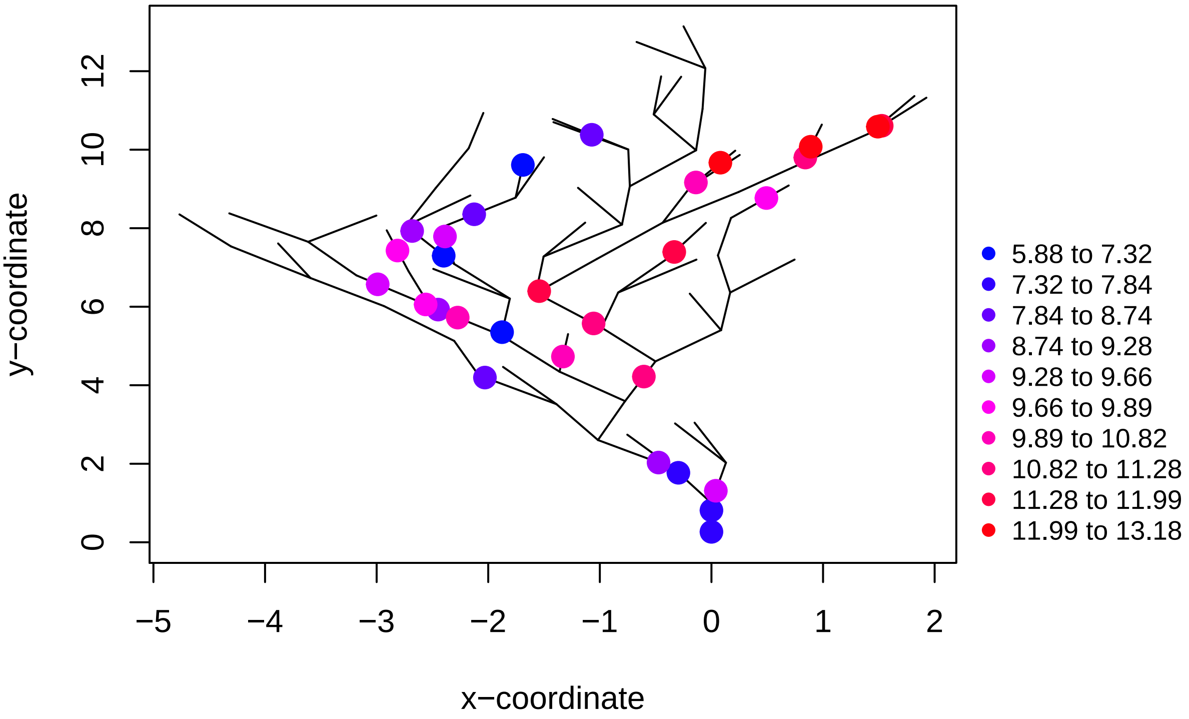
The `demoY` dataset is currently constructed in a suitable form for S-mode PCA. If T-mode PCA is of interest then the PCA is performed on a dataframe where the columns are time points and the monitoring sites are rows.

Important: The dataframe must always be entered in functions with columns representing monitoring sites and rows representing time points, as if S-mode PCA is to be performed. The functions will automatically transpose the data if T-mode PCA is required.

The river network for the demo data, demoNet, is an object of class SpatialStreamNetwork (SSN, see documentation for the SSN package in R for more details), and is displayed below. The points are coloured by the “observed” values at the first time point:

```r
data(demoNet)

plot(demoNet, "Sim_Values",
     xlab="x-coordinate", ylab="y-coordinate",
     cex=1.5)
```



Inspection of the plot shows that the sites in the top right of the plot (all coloured purple and red) have higher values than the other sites in the network.

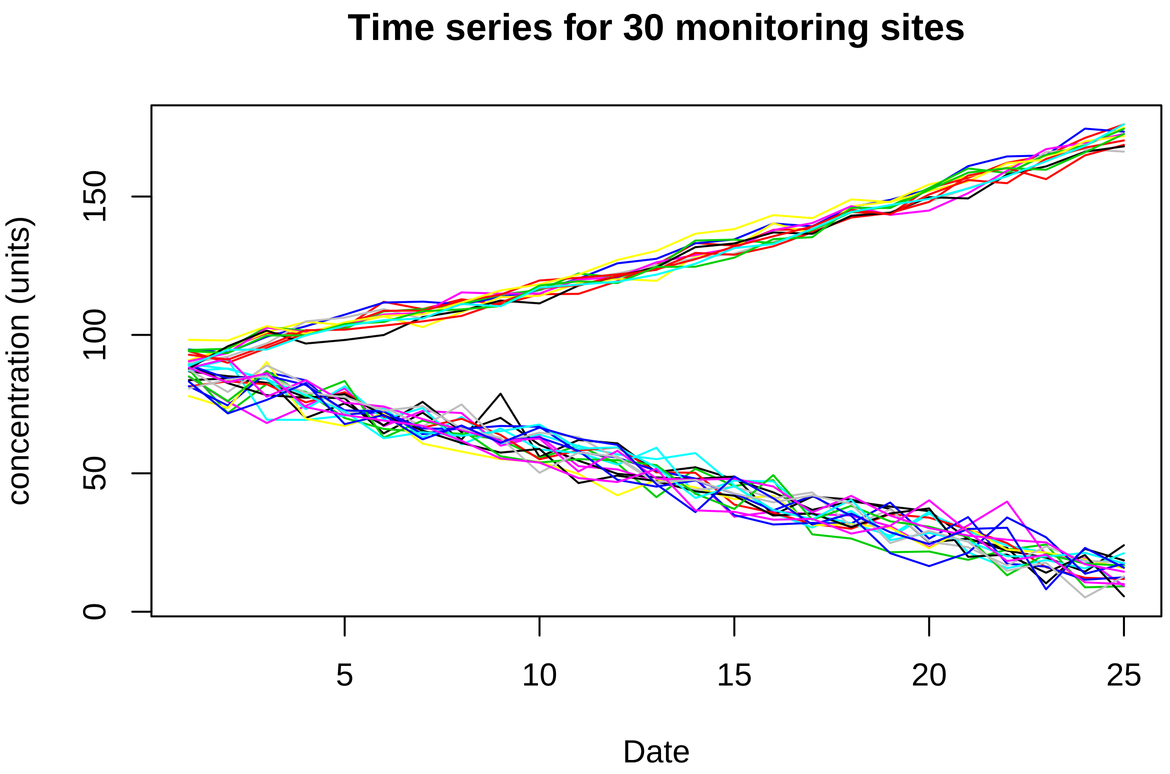
To read in your own SSN object, use the following:

```r
library(SSN)

# get the filepath for the SSN object, including the .ssn extension
demoNet.path <- system.file("demoSSN/demoNet.ssn", package="stpca")

# import the SSN object.  Please refer to the importSSN() help file
# in the SSN package for more details
demoNet <- importSSN(demoNet.path)
```

Each monitoring site in demoNet has observed values simulated for 25 time points and the 30 time series are shown below:



You can see that in this demo dataframe, some monitoring sites have a decreasing temporal pattern and some have an increasing temporal pattern. Further details about these spatiotemporal observations can be found using `?demoY`, `demoNet`, or in the “Creating demo data” vignette.

4 Pre-processing

This section will describe how the `createWeightS()`, `createWeightT()`, and `completeData()` functions can be used to prepare the data before applying principal components analysis (PCA, using the `stpca()` function).

4.1 Create Weights

Two functions are provided in the `stpca` package: `createWeightS()` and `createWeightT()` that can be used to calculate matrices of spatial or temporal weights, respectively, reflecting spatial or temporal correlation in the data. The resulting matrices can then be used as inputs for the function that performs PCA, `stpca()`. This section will describe possible spatial and temporal weights and demonstrate how these can be calculated using the appropriate functions.

4.1.1 Spatial weights

The `createWeightS()` function can be used to calculate spatial weights using hte information contained within an SSN object. The function first extracts the stream distance matrix for observed data from an SSN object and constructs an asymmetric weight matrix based on the specified additive function value. This weight matrix reflects flow direction in the river network, and strength of connectivity between monitoring sites. A matrix of spatial weights can be created using the following:

```
spatial.wt <- createWeightS(ssndata = demoNet.path,
                            afvcol = "addfunccol")
```

```
# display the first few rows and columns
head(spatial.wt)[,1:7]
```

```
##           1         2         3         4         5 6         7
## 1 1.0000000 0.0000000 0.0000000 0.0000000 0.5773503 0 0.0000000
## 2 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0 0.0000000
## 3 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 0 0.0000000
## 4 0.0000000 0.4472136 0.0000000 1.0000000 0.0000000 0 0.4472136
## 5 0.0000000 0.0000000 0.0000000 0.0000000 1.0000000 0 0.0000000
## 6 0.3015113 0.1740777 0.3892495 0.3892495 0.1740777 1 0.1740777
```

The function requires two inputs: `ssndata` is the filepath to the SSN object and `afvcol` is the name of the column of additive function values in the dataset containing observed values within the SSN object. The function returns an asymmetric $p \times p$ matrix of spatial weights, where p is the number of monitoring sites. The columns of the returned matrix represent upstream sites while the rows represent downstream sites. A non-zero value means that the upstream monitoring site is directly connected to the downstream site by the flow of water in the river network. A zero value means that two sites are not flow connected. The resulting weight matrix should only contain values between 0 and 1.

4.1.2 Temporal weights

The `createWeightT()` function creates a symmetric matrix of temporal weights based on temporal autocorrelation. At present only AR(1) structure has been implemented.

These weights describe the strength of correlation ρ (you can think of this as the strength of relationship) between the observed value at time point t (y_t,), and the observed value at the previous time point (y_{t-1},). This means that the observed value at the present time point is some multiple of the value at the previous time point, plus random independent error.

$$y_t = \rho y_{t-1} + \varepsilon_t$$

The random error $\varepsilon_t \sim \mathrm{N}(0, \sigma^2)$.

The temporal correlation matrix can be constructed by using the code below. The first few rows and columns are displayed.

```
temporal.wt <- createWeightT(n = 25, rho = 0.75)
head(temporal.wt)[,1:7]
```

```
##           [,1]      [,2]     [,3]     [,4]      [,5]      [,6]      [,7]
## [1,] 1.0000000 0.7500000 0.562500 0.421875 0.3164062 0.2373047 0.1779785
## [2,] 0.7500000 1.0000000 0.750000 0.562500 0.4218750 0.3164062 0.2373047
## [3,] 0.5625000 0.7500000 1.000000 0.750000 0.5625000 0.4218750 0.3164062
## [4,] 0.4218750 0.5625000 0.750000 1.000000 0.7500000 0.5625000 0.4218750
## [5,] 0.3164062 0.4218750 0.562500 0.750000 1.0000000 0.7500000 0.5625000
## [6,] 0.2373047 0.3164062 0.421875 0.562500 0.7500000 1.0000000 0.7500000
```

The function requires two inputs: `n` is the number of time points in the dataframe and `rho` (ρ)is the correlation parameter. The function does not estimate ρ and so the user must determine a suitable value for this, where $\rho \in (0, 1)$, prior to using this function. There are many functions avaiable in R to do this.

4.2 Impute missing values

In order to perform PCA the data have to be regularly spaced with no missing values. For datasets that have missing values a wrapper function has been provided here to impute missing values using the `missMDA` package (Husson and Josse 2015). This imputation method is also based on PCA and replaces missing values by first replacing missing values with column means, performing PCA and then reconstructing the data from a specified number of principal components. The missing values, originally replaced by column means, are then replaced by the values from reconstructing the data. The number of principal components used to reconstruct the data must be estimated and the function estimates this number and returns a complete dataset with no missing values. Diagnostic plots can also be produced if required.

For example, first of all read in a demo dataset of the same structure as `demoY` but with missing values, and inspect the first few rows and columns:

```
data(demoYmiss)
head(demoYmiss[,1:7])
```

```
##         x1        x2       x3        x4       x5       x6        x7
## 1 83.66534        NA 87.56660  94.74550 87.72304 84.89257        NA
## 2 84.26214  95.17072 71.76019  93.64118 91.07545 75.92171  95.23313
## 3 82.70443 100.67188 82.30221        NA 69.37519 68.17076 101.01647
## 4 77.43921 100.51986 79.44762 103.12816 69.32245 74.34863 104.73041
## 5 77.12826 102.73167       NA 107.31950 70.81690 80.65534        NA
## 6 64.41192        NA 66.04143 111.69628 73.59843 67.04863 106.72822
```

Missing values are represented as `NA`. You can choose to impute missing values in either T-mode or S-mode and so the function will make use of correlation between time points or monitoring sites, respectively, to provide estimates of missing values. As an example, we will look at imputing missing values in S-mode.

First, will create a complete datase with no missing values, suppressing the diagonostic plots for the moment. The missing values can be imputed using the following:

```
Smode.impute <- completeData(x = demoYmiss, pca.mode = "Smode",
                             estim.plot = FALSE,
                             mi.plot = FALSE,
                             mean.plot = FALSE)
```

This function has two inputs that must be spcified in order that the function can run: `x` is the dataframe with missing values and `pca.mode` is the PCA mode (`"Smode"` or `"Tmode"`). The arguments `mean.plot`, `estim.plot`, and `mi.plot` are used to specify whether or not diagnostic plots should be produced (we will look at these later).

Several outputs are produced (many of which are used to produce the diagnostic plots):

```
names(Smode.impute)
```

```
## [1] "ncp"       "criterion" "res.MIPCA"
```

```
names(Smode.impute$res.MIPCA)
```

```
## [1] "res.MI"        "res.imputePCA" "call"
```

The complete dataset can be found in this example in `Smode.impute$res.MIPCA$res.imputePCA`.

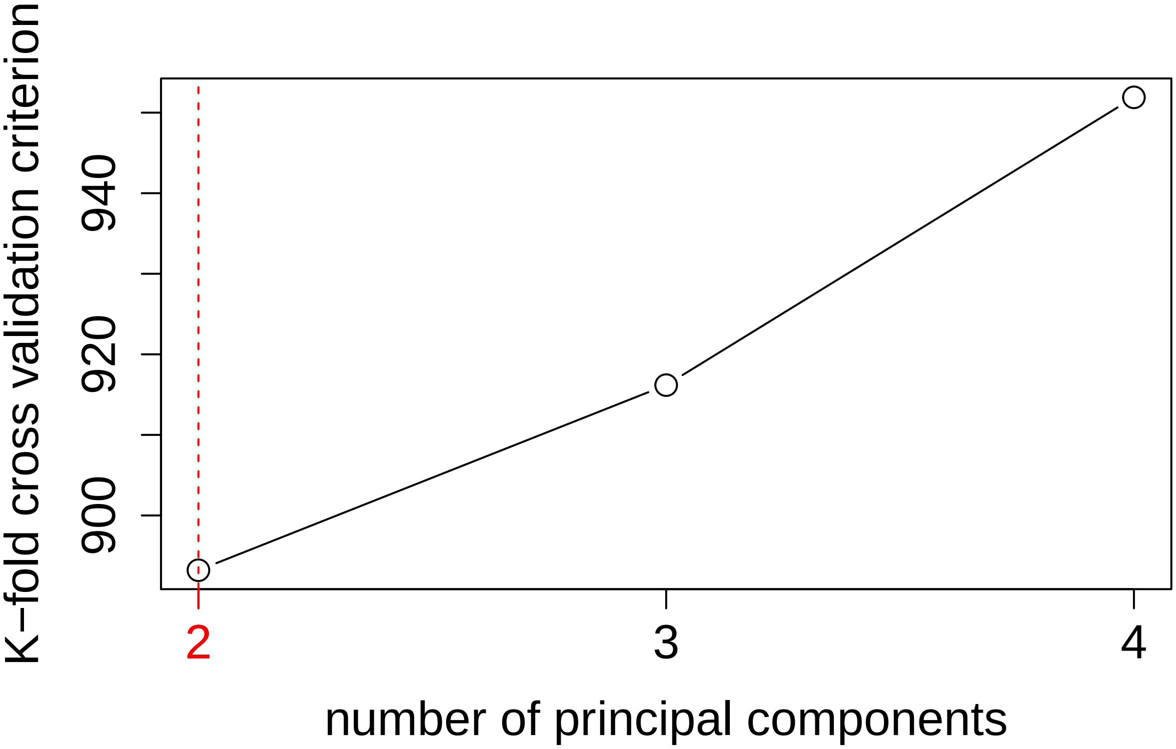
A description of the other outputs can be found in the help file for this function which you can access using `?completeData` (providing the `stpca` package has been loaded in R).

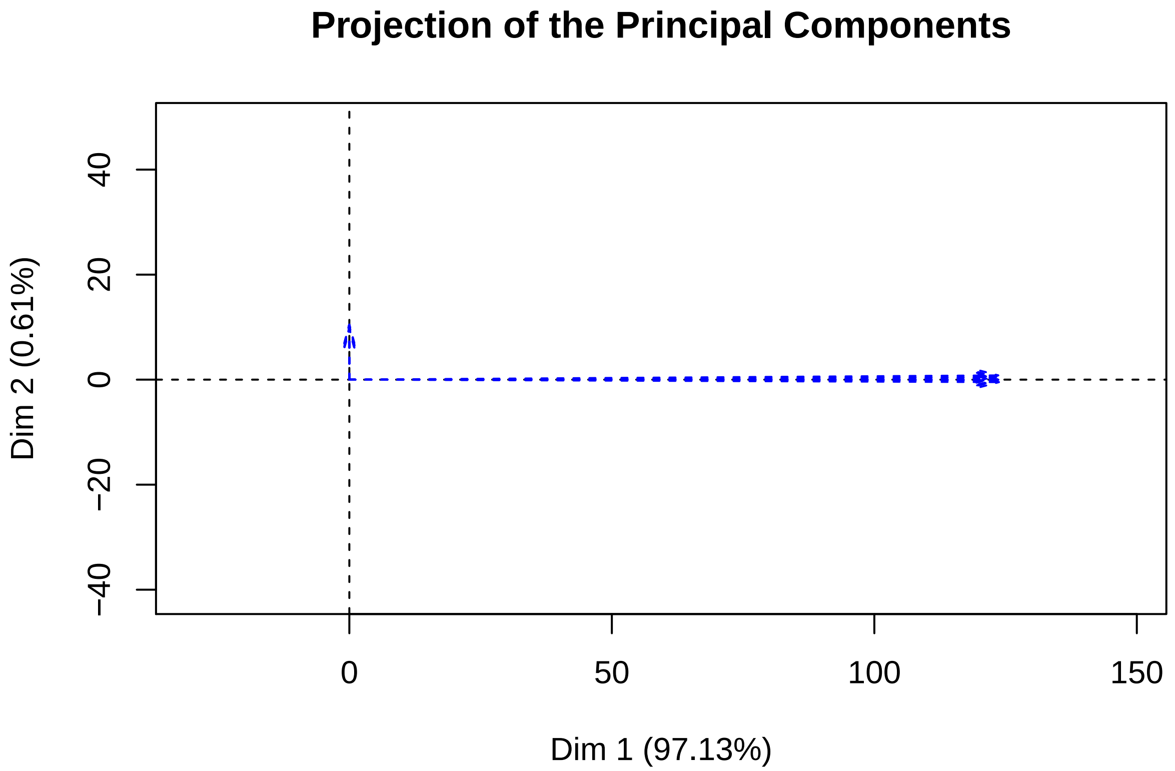
Next, we can look at some of the other options that are available in the `completeData()` function. First, specify the minimum and maximum number of principal components to use to calculate missing values, `ncp.min` and `ncp.max` respectively. In order to look at all of the possible plots `ncp` is forced in this example to be greater than 1. For each integer in the range [`min.ncp`, `max.ncp`], K-fold cross validation is used to identify the `ncp` that minimises the cross validation criterion.

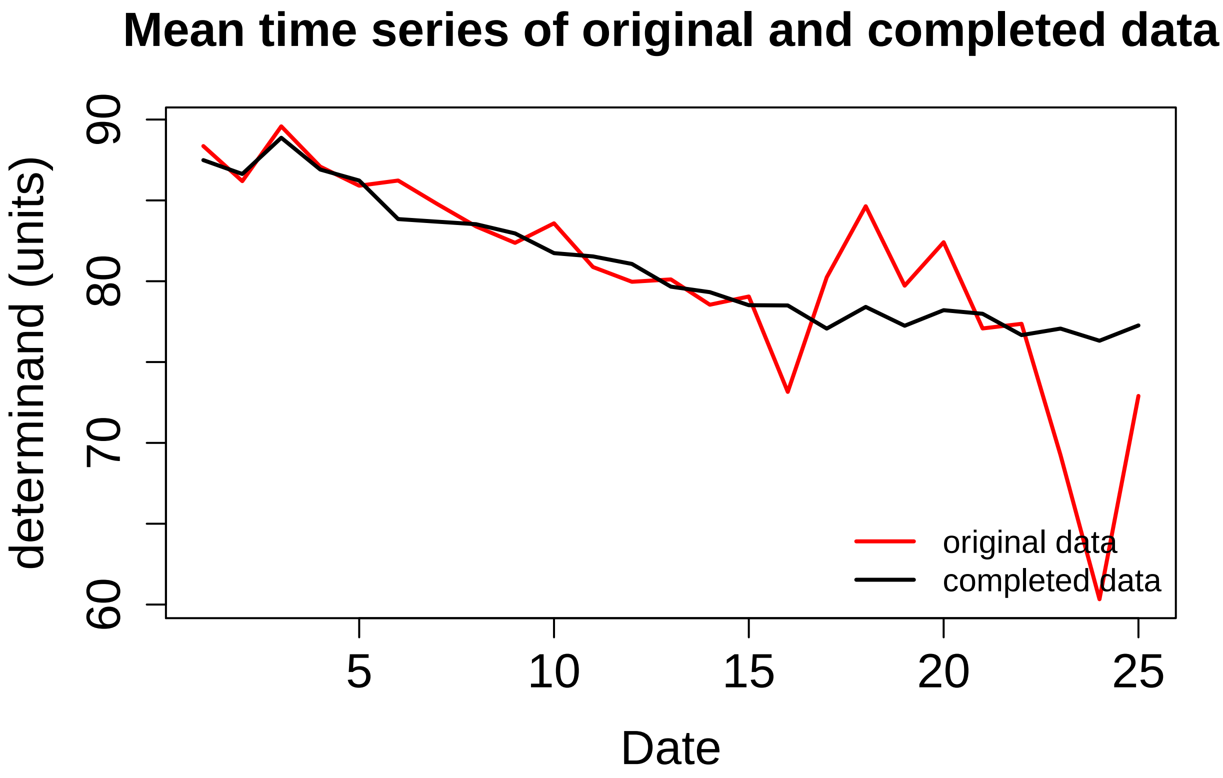
Next, `estim.plot`, `mi.plot`, and `mean.plot` are set to `TRUE` so that all possible diagnostic plots are produced. Then, `response.var` is used to supply the y-axis label for `mean.plot`.

Finally, `nbsim` and `nboot` are specified. `nbsim` is the number of simulations used for K-fold cross validation and `nboot` is the number of simulated datasets to produce to investigate variability in imputation results due to missingness (see `MIPCA` in the missMDA package for further details). `nbsim` and `nboot` are required to produce `mi.plot`.

```
Smode.impute <- completeData(x = demoYmiss, pca.mode = "Smode",
                             ncp.min = 2, ncp.max = 4,
                             estim.plot = TRUE,
                             mi.plot = TRUE,
                             mean.plot = TRUE,
                             response.var = "determinand (units)",
                             nbsim = 10, nboot = 10)
```


K−fold cross validation criterion
900
920
940
2
3
4
number of principal components


Projection of the Principal Components
Dim 2 (0.61%)
40
20
0
−20
−40
0
50
100
150
Dim 1 (97.13%)



The first plot is `estim.plot` and shows the K-fold cross validation results for each integer in the range [`ncp.min`, `ncp.max`]. The number of PC's (`Smode.impute$ncp`) that gives the smalles K-fold cross validation criterion is highlighted in red on the x-axis and by a red vertical dashed line, and this is the number of PC's used to calculate missing values.

The second plot is `mi.plot` and gives some indication of the uncertainty in the completed dataset due to estimating missing values. This plot can only be produced if `ncp`>1 and a warning message will be produced if `mi.plot` is set to `TRUE` but `ncp` is estimated to be less than 2. If the blue arrows are closely grouped around the horizontal and vertical lines then this means there is little uncertainty associated with missing values. If the blue arrows are widely spread around the dashed lines then you should proceed with caution when using the completed dataset. It is recommended that in this case, any subsequent analyses should be performed on several of the `nbsim` simulated data sets to assess the variability of any inferences made from analysing the completed dataset.

The third plot is `mean.plot` and shows the average temporal pattern for all monitoring sites based on the original incomplete dataset (red line) and based on the dataset completed using imputation (black line). This plot can help the user assess how closely the completed data resembles the original data with missing values. It is recommended that the user spends some time investigating any large differences between the two lines. One possible reason is that at some time points, there are very few monitoring sites with observed values,and those values that are observed are much higher or lower than would be expected at other monitoring sites.

It is recommended that `nbsim` an `nboot` are set quite low at first to get an initial impression of the results but that the function is run with much higher values (such as the default values where `nbsim = nboot = 100`) to produce a complete data set for further analyis.

A further argument `scale = FALSE` is also available. Scaling the data means dividing each value in a column by the column standard deviation, and is usually used with PCA to prevent the analysis from being dominated by a few variables with the highest variances, which can happen when the variables are recorded on scales of different magnitude. Setting scale to FALSE means that you intend to perform PCA on the covariance matrix of the data, while TRUE means that you are going to perform PCA on the correlation matrix. Since the observations in the demo data are all on the same scale it makes sense to try PCA on the covariance matrix, i.e. setting scale = FALSE. If you think the analysis is being dominated by a small number of variables (monitoring sites in S-mode, or time points in T-mode) with the highest variance, then try setting scale = TRUE. If scale = FALSE here, you should also use scale = FALSE when using stpca().

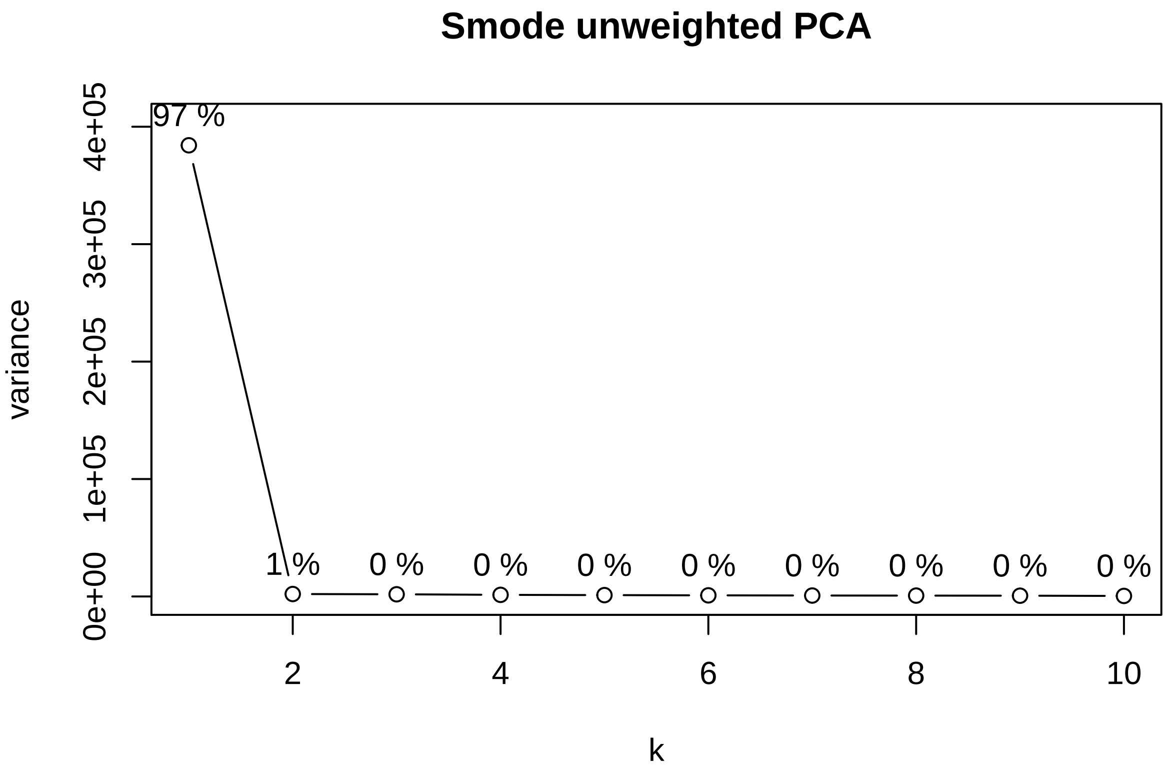
5 Data Analysis

The stpca() function performs PCA on the spatiotemporal data in T-mode or S-mode using singular value decomposition of the data matrix (rather than eigen decomposition of the covariance or correlation matrix). PCA can be adjusted using spatial and/or temporal weights if desired. If spatial or temporal weights are used then the returned loadings and principal components are backtransformed using the transformation described in the main reference for this tutorial document.

5.1 S-mode

Firstly, we will fit an S-mode PCA with no weights

```
Smode.pca.uw <- stpca(x = demoY,
                      pca.mode = "Smode",
                      pca.wt = c("unweighted"))
```



The code above only needs 3 inputs - x is the dataframe on which to perform PCA, pca.mode is the mode of

PCA: either "Smode" or "Tmode", and pca.wt is the type of weighting to use. Specifying "unweighted" performs a standard S-mode PCA.

This code will produce a scree plot for the first 10 PC's by default. Once PCA is carried out it is up to the user to decide how many PC's to interpret and the scree plot can be used for this (although there are several other possibilities in the PCA literature). Look for an 'elbow' in the plot and the PC's before the bend are the ones most of interest. In this example of unweighted S-mode PCA, only one PC appears before the bend and so we will only attempt to interpret the first PC.

We can see what proportion of the total variance in the data is explained by the first PC:

```
Smode.pca.uw$unweighted$var.prop[1]
```

```
## [1] 0.9690309
```

Or the total amount of variance explained by the first three PC's:

```
Smode.pca.uw$unweighted$cumvar[1:3]
```

```
## [1] 0.9690309 0.9742944 0.9789910
```

This means that the first three PC's explain 97.9% of the variance in the data.

Now let's fit S-mode PCA using the spatial weights and temporal weights created previously. First, we need to check that the column names in demoY are the same as the column (and row) names of the matrix of spatial weights, spatial.wt:

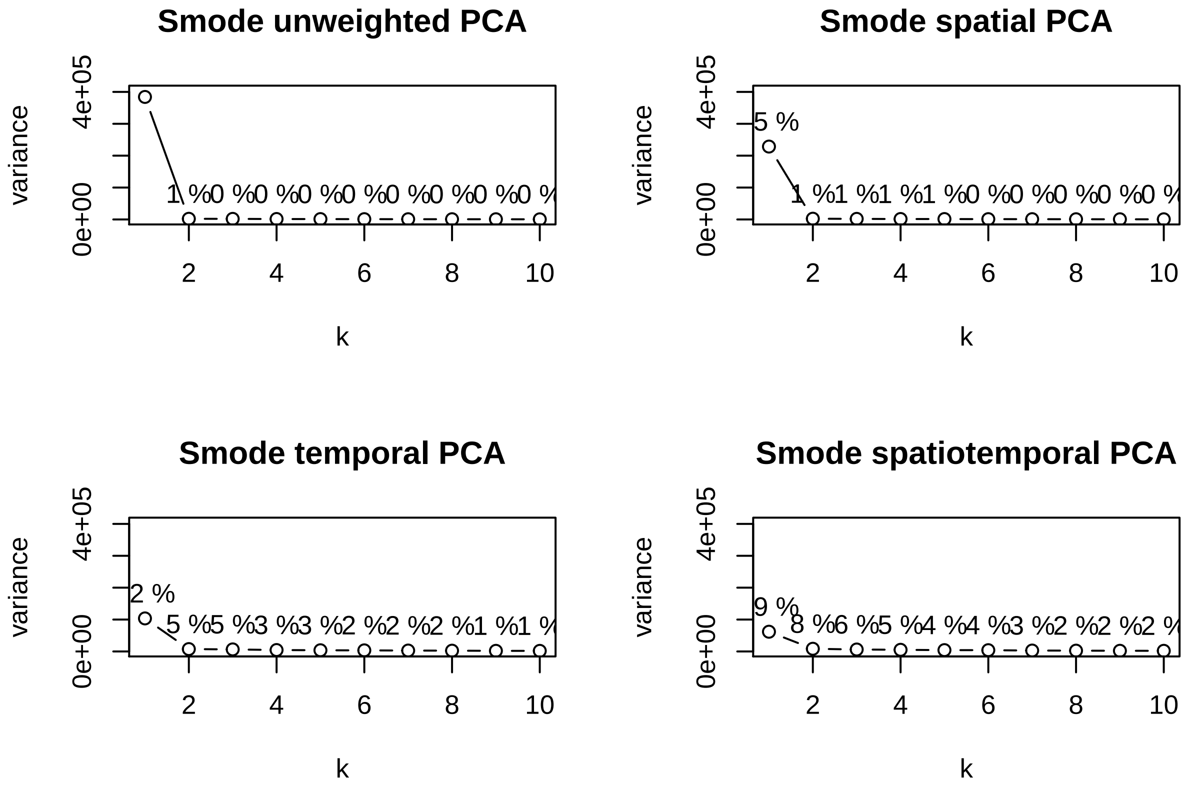
```
mean(colnames(spatial.wt) == colnames(demoY))
```

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

If the result is 1 then the order of the monitoring sites in the dataframe are in the same order as the monitoring sites in the spatial weight matrix and you can proceed. If the result is not 1 then you need to fix this before using stpca() with spatial weights. The column names of the dataset containg spatiotemporal observations (demoY in this example) **MUST** be of the same form and in the same order as the columns (and rows) of the matrix of spatial weights. This is because the stpca() function checks for this to ensure spatial relationships between monitoring sites are correctly represented and if they are not the same then an error message will be produced and the function will fail to run.

Now, we can fit S-mode PCA with spatial and temporal weights as:

```
Smode.pca.all <- stpca(x = demoY,
                       pca.mode = "Smode",
                       spatial.wt = spatial.wt,
                       temporal.wt = temporal.wt,
                       pca.wt = c("unweighted", "spatial",
                                  "temporal", "spatiotemporal"))
```



The arguments `spatial.wt` and `temporal.wt` are used to specify the names of the matrices containing spatial and temporal weights. Spatial and temporal weights can be calculated using the `createWeightS()` and `createWeightT()` functions described earlier but you are free to include your own matrices of spatial and temporal weights, providing they are of the correct size. `pca.wt` has had three other weighting schemes added so the function will perform PCA under four different weighting schemes, and produce results for each of these.

5.2 T-mode

In order to fit T-mode PCA, the dataframe must have monitoring sites in columns and time points in rows. This seems strange at first since T-mode PCA is performed on data where the columns are time points and the rows are monitoring sites but in fact the `stpca()` function will automatically transpose the data to the correct orientation if `pca.mode = "Tmode"`.

Important: If you have imputed missing values using the completeData() function and set pca.mode = "Tmode" then you will need to transpose the completed dataframe so that columns are monitoring sites and rows are time points.

For example, the completed dataset with no missing values produced using `completeData()` can be transposed so that columns are monitoring sites and rows are time points:

```
# first impute missing values
Tmode.impute <- completeData(x = demoYmiss, pca.mode = "Tmode",
                             mean.plot = FALSE, estim.plot = FALSE,
                             mi.plot = FALSE,
                             nbsim = 10, nboot = 10)
```

```
# transpose the completed data so that columns are monitoring sites and
# rows are time points
Tmode.completedata <- t(Tmode.impute$res.MIPCA$res.imputePCA)

# check column names are in same format/order as "spatial.wt"
mean(colnames(Tmode.completedata) == colnames(spatial.wt))
```

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

```
colnames(Tmode.completedata)[1:5]
```

```
## [1] "x1" "x2" "x3" "x4" "x5"
```

```
colnames(spatial.wt)[1:5]
```

```
## [1] "1" "2" "3" "4" "5"
```

```
# need to remove "x" from column names
colnames(Tmode.completedata) <- 1:30
```

We can fit a T-mode PCA. Firstly, unweighted. . . .

```
Tmode.pca.uw <- stpca(x = Tmode.completedata,
                      pca.mode = "Tmode",
                      pca.wt = c("unweighted"),
                      scree = FALSE)
```

and then weighted. . . .

```
Tmode.pca.all <- stpca(x = Tmode.completedata,
                       pca.mode = "Tmode",
                       spatial.wt = spatial.wt,
                       temporal.wt = temporal.wt,
                       pca.wt = c("unweighted", "spatial",
                                  "temporal", "spatiotemporal"),
                       scree = FALSE)
```

5.3 Other options

Other options you might want to consider when using `stpca()` are `center` and `scale`. These are both logical arguments, taking a value of either `TRUE` or `FALSE`. The default settings are `center = TRUE` and `scale = FALSE`. This means that the dataframe (`demoY` in the example here) is centered by subtracting the column mean from each value in the dataframe and this is equivalent to PCA on the covariance matrix. Setting scale = TRUE means that each value in the dataframe will be divided by the column standard deviation and this is equivalent to PCA on the correlation matrix.

Centering the data is standard practice in PCA but the option to center the data or not has been included in the function in case data have already been centered. It is up to the user to decide whether or not to scale the data. In general, if the columns of the data frame are all measured in different units then scaling the data will mean that the analysis is not dominated by the variable(s) with the largest values.

6 Visualising results

The plots.stpca() function plots results from stpca(). Five types of plots can be produced and the interpretation of these plots will depend on whether T-mode or S-mode PCA has been performed We will look at each of the plots, first for S-mode and then for T-mode PCA. For S-mode we will only look at the results for unweighted (standard) PCA in order to introduce the plot.stpca() function, while for T-mode we will look at all of the plots for the four possible weighting schemes so that comparisons of the plots can be discussed.

Note that for plots = "map" and plots = "glyph" it is assumed that you have a shapefile (.shp) that can be used to plot the river network.

6.1 S-mode

6.1.1 S-mode: map

The map plot is suitable for plotting loadings (in S-mode) for a single PC. First, read in the shapefile for the river network. This can be any appropriate .shp file. If the user has an SSN object (as discussed earlier) then the edges.shp file from the SSN object can used to represent the lines of the river network. The edges.shp file will be in a folder whose name has extension .ssn.

You will also need coordinate information for the location of the monitoring sites.

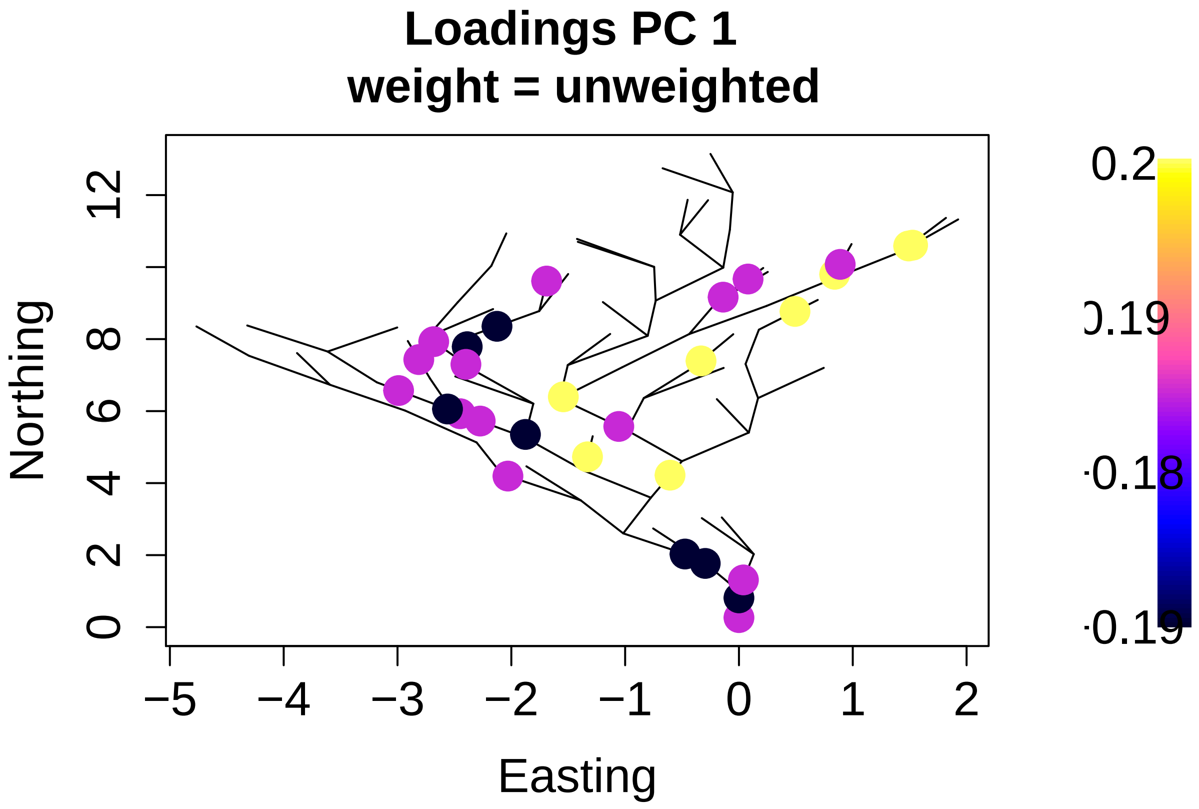
```
# load maptools library, required for reading in a .shp file
library(maptools)
# set the filepath for the .shp file
river.path <- paste(demoNet.path, "edges.shp", sep="/")
# read in the shapefile for the river network
river.dat <- readShapeLines(river.path)

# the coordinate information for the monitoring sites is stored here in the SSN
# object for the demo dataset.  First, import the SSN object
library(SSN)
demo <- importSSN(demoNet.path)

# pull out the "Obs"" dataframe from the SSN object to get the coordinates for
# the monitoring sites
coords <- getSSNdata.frame(demo, "Obs")[, c("NEAR_X", "NEAR_Y")]
```

Now produce the plot:

```
# produce the map plot
plot_stpca(x = Smode.pca.uw, plots = "map",
           river = river.dat, coords = coords)
```



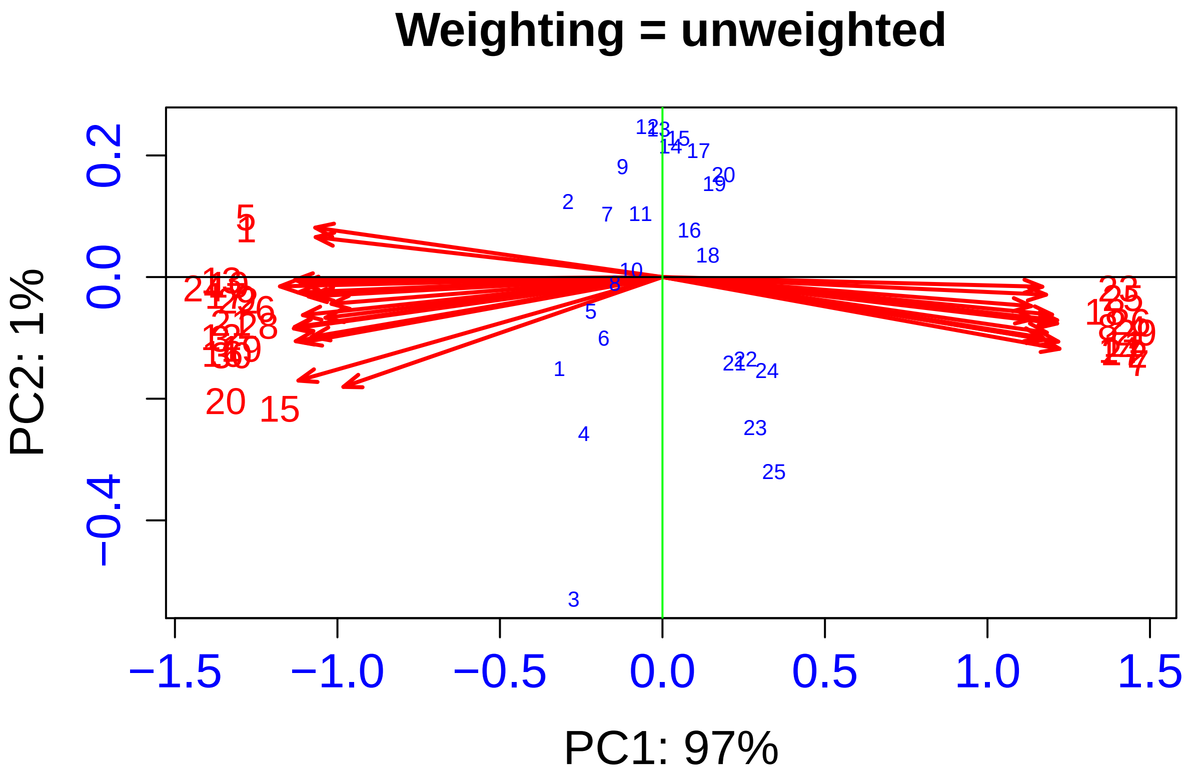
This code requires four inputs: `x` is the name of the object containing the results from `stpca()`, `plots` specifies which plot to produce, `river` specifies the name of the .shp file for the river network, and `coords` specifies the dataframe containing coordinates for the monitoring sites. Note that the rows in `coords` must be in the same order as the columns in `demoY` otherwise the results will be plotted at the wrong location.

Two other arguments might be specified when `plots = "map"`. `pc.map = 1` by default, but this can be changed to plot the results for a different PC. A second argument, `probs` can also be changed to alter the appearance of the map. By default this is set to `probs = c(0, 0.25, 0.75, 1)`, and the colours in the map indicate the 25% of smallest loadings (black) and 25% highest loadings (yellow), while purple indicates the middle 50%. `probs` is passed to the `quantile()` function in R and so any vector of values can be included in `probs` as long as all values are in the range [0,1].

6.1.2 S-mode: biplot

The biplot is commonly used to display the results of pairs of PC’s and can be produced as:

```
plot_stpca(x = Smode.pca.uw, plots = "biplot",
           biplot.factor = 100)
```



This requires three inputs: `x` is the name of the object in which the results from `stpca()` are stored, `plots` is used to specify the type of plot to be produced, and `biplot.factor` is a scalar used to improve the viewing of the biplot where smaller values will increase the length of the arrows in the plot.

A fourth argument can be included: `pc.biplot` specifies which two PC's should be plotted. The biplot can be produced for any pair of PC's but the default is `c(1, 2)` meaning that the biplot will be produced for the first two PC's. The axis labels show the percentage of total variance explained by these PC's.

Interpreting the biplot can be difficult and much material is available online to help with this but some general guidlines are:

- The red arrows are the loadings and the blue points are the principal components, or scores. The x and y axis correspond to the principal components.
- Arrows of similar length, pointing in the same direction represent variables with equal importance in the PC and are positively correlated. Arrows pointing in opposite directions represent variables that are negatively correlated.

- If a score (blue point) is in the same quadrant as an arrow head, then the observed value is high for that variable.

For the simulated observations in `demoY`, we can say, from the biplot:

- In S-mode PCA, the inspection of the arrows can reveal groups of monitoring sites with similar temporal patterns.
- PC1 captures 2 groups of monitoring sites in the data, and the monitoring sites in each group are found by looking at the arrow head labels on either side of the vertical green line. Almost all of the variance in the data is related to these two groups.

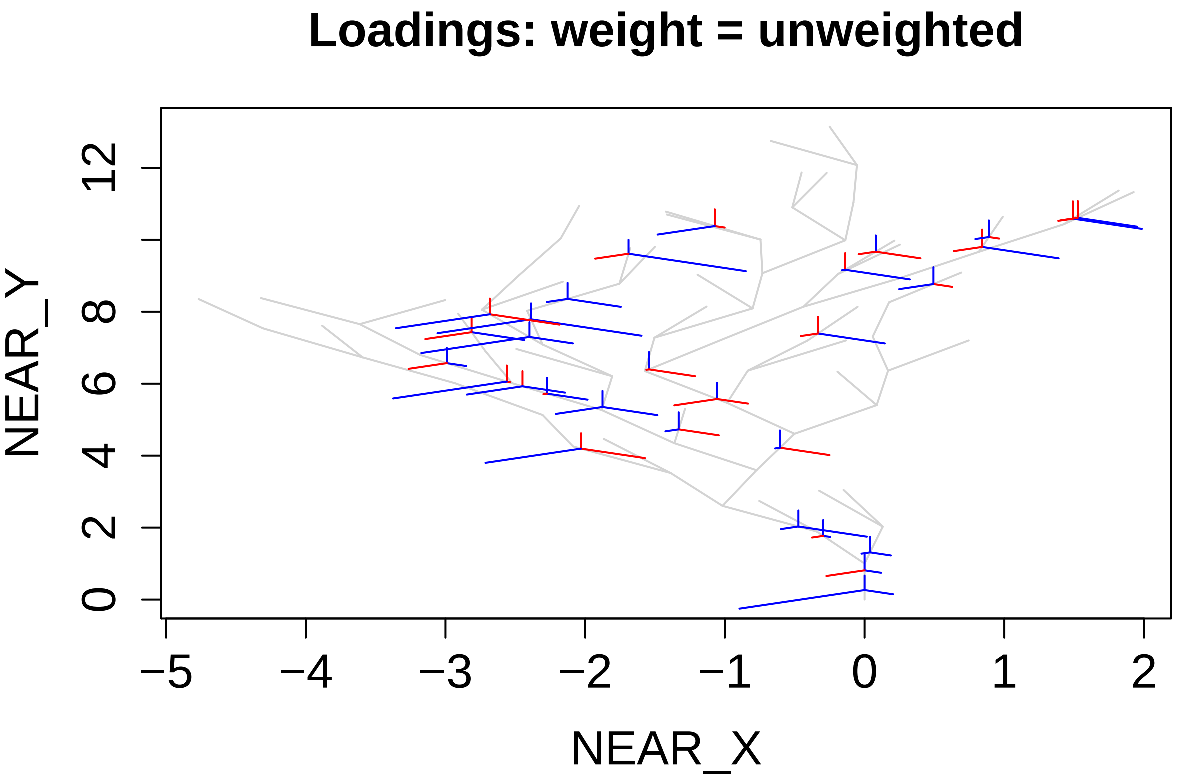
- PC2 shows that monitoring sites x1 and x5 are different to the rest of the monitoring sites, but this only accounts for 1% of the variance in the data.

6.1.3 S-mode: glyph plot

A glyph plot shows the loadings (S-mode) or principal components (T-mode) for three or more PC's at each monitoring site location. It is assumed for this plot that you have access to a shapefile of the river network.

```
plot_stpca(x = Smode.pca.uw, plots = "glyph", river = river.dat, coords = coords,
    r1 = 10)
```

```
## For plots = glyph: did you check that coordinates for monitoring site ID's are in
## the same order as data used for stpca()?
## If not then glyphs might not correspond to the correct monitoring site.
```



Four inputs are required to produce the glyph plot: `x` is the name of the object in which the results from `stpca()` are stored, `plots = "glyph"` specifies which plot is to be produced, `river` is the name of the object containing the river network, and `r1` is a scaling factor used to improve the viewing of the plot where large values will decrease the size of the glyphs.

A fifth argument: `pc.glyph` can be used to specify which PC's should be displayed. The default is `c(1, 2, 3)`. The vector specified by `pc.glyph` should be of length three or more. The plot will be produced if `pc.glyph` contains fewer than three values, but a warning message will be produced.

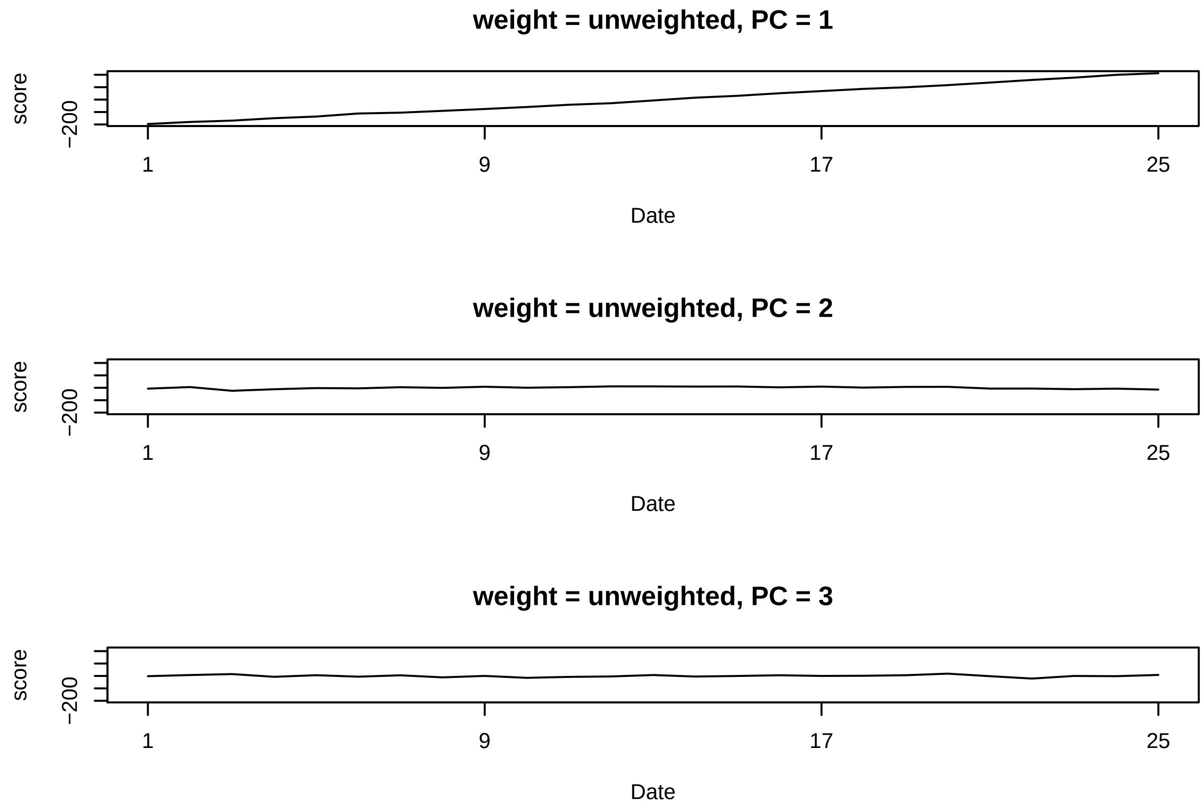
The S-mode glyph plot produced here shows positive loadings in blue and negative loadings in red. The loading for the first PC is shown in the twelve o'clock position, moving clockwise through the other PC's specified using `pc.glyph`. The length of the line indicates the magnitude of the loadings.

A message will appear in the console window when you produced this plot reminding you check that the order of the rows in the coordinates dataframe is the same as the order of the columns in the `demoY` dataframe. This is NOT a warning to say they are in a different order but prompts the user to check that results are being displayed at the correct locations.

6.1.4 S-mode: time series plot

A time series plot shows the principal components plotted over time and gives an indication of the temporal pattern captured by each principal component. The time series plot can be produced by the following:

```r
plot_stpca(x = Smode.pca.uw, plots = "ts")
```



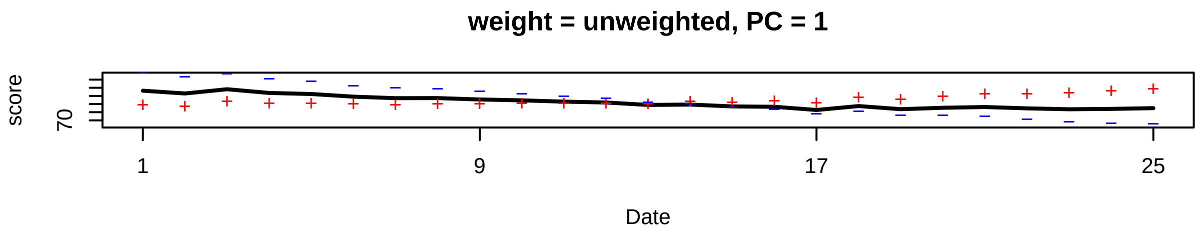
This requires two inputs: `x` is the name of the object in which the PCA results are stored, and `plots = "ts"` specified which plot to produce. A third argument: `pc.ts` can be included which specifies the PC's to be plotted, the default is `c(1, 2, 3)`. You can use `"ts"` to display as many PC's as you wish but a large number will be difficult to view on the screen. This plot can only be produced for S-mode PCA since the time series of the PC's displays a temporal pattern, whereas in T-mode, a time series of the loadings has no meaningful interpretation.

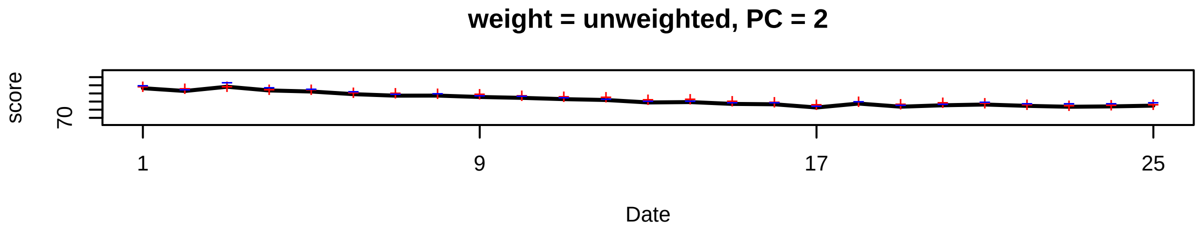
The time series plot produced here shows that the first principal component captures an almost linear temporal pattern. It is important to remember however that since loadings can be positive or negative, the time series plot alone is not enough to determine whether the prinicpal component captures an upward or downward trend. The biplot or a map of the loadings is useful here to determine the direction of the trend. For example, the biplot shows that the loading for PC1 is negative for monitoring site x15, meaning that the temporal pattern at x15 is the negative of the pattern seen in the time series plot and so this monitoring site in fact has a downward trend over time.

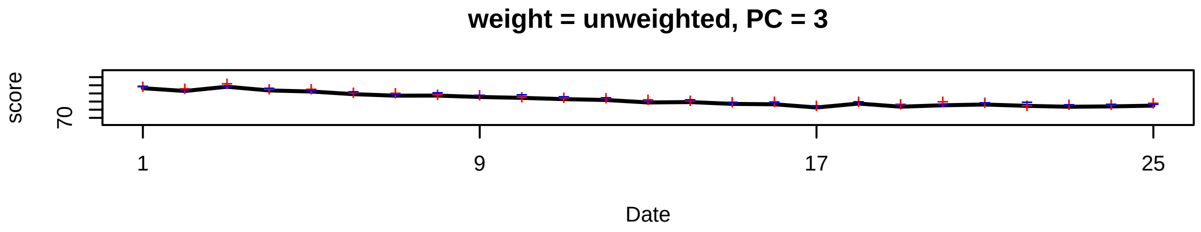
6.1.5 S-mode: mean +/- scores plot

A final interesting plot to consider is how the temporal pattern identified from the S-mode PCA scores relates to the mean time series from all monitoring sites. Such plots can be produced using the following:

```r
plot_stpca(x = Smode.pca.uw, plots = "meanPM")
```







Two inputs are required here: `x` is the name of the object in which the PCA results are stored, and `plots = "meanPM"` specifies which plot is to be produced. Two further inputs are avaialable: `pc.meanPM = c(1, 2, 3)` can be used to specify which PC's should be plotted, and `meanPM.factor = 0.05` is a scaling factor used to improve the contrast between the mean time series and the other points on the plot.

As with `plots = "ts"`, the vector `pc.meanPM` can be of any length but large numbers of plots will not be easy to view. `meanPM.factor` can be adjusted until you are happy with how the plot looks as the red and blue points help illustrate the variance around the mean time series (black line) captured the principal component. The red symbols are *mean time series + (meanPM.factor* scores)* while the blue symbols are *mean time series - (meanPM.factor* scores)*. The size of the gap between the coloured points and the black line has no interpretation, rather it is the pattern of the red and blue points around the black line that is important.

On this plot, the first PC captures a shift in mean where some monitoring sites will exhibit a temporal pattern lower than the mean early in the time period, shifting to higher than the mean later in the time period. Other monitoring sites will display the opposite pattern with values higher than the mean early in the time period, shifting to lower than the mean at the end of the time period.

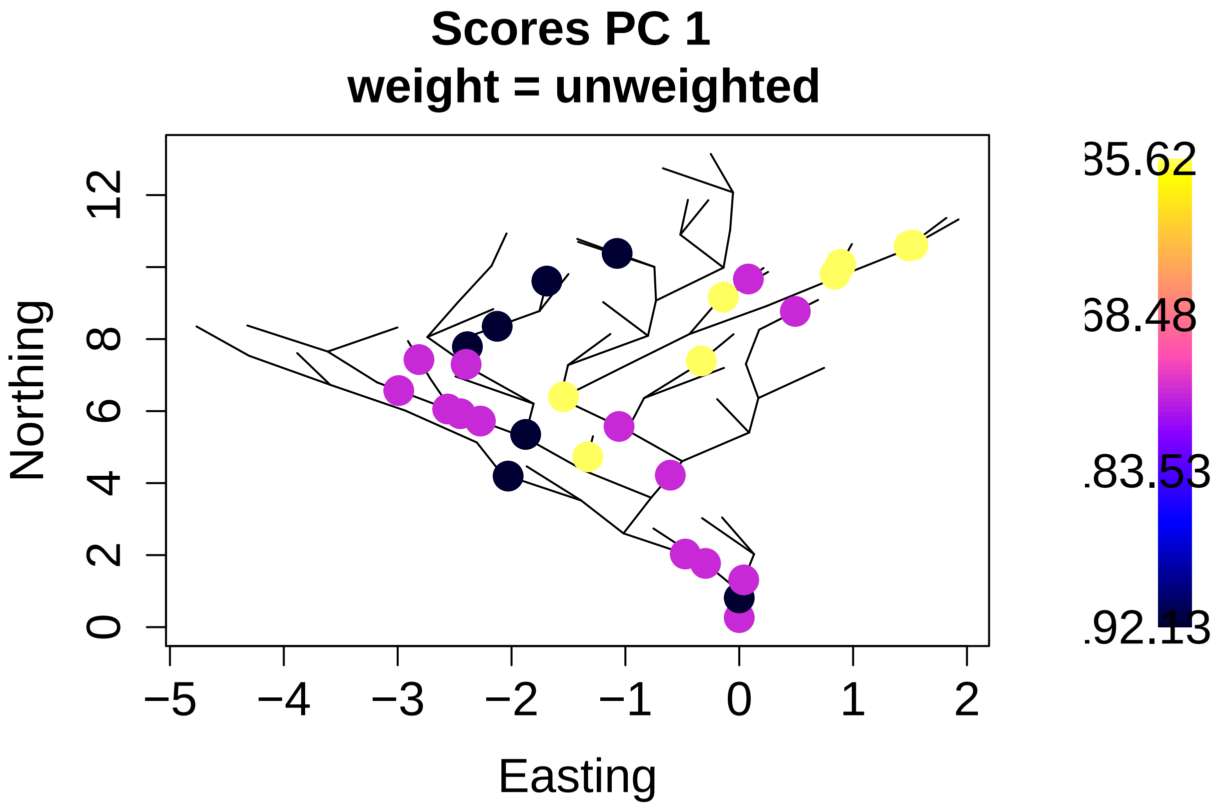
6.2 T-mode

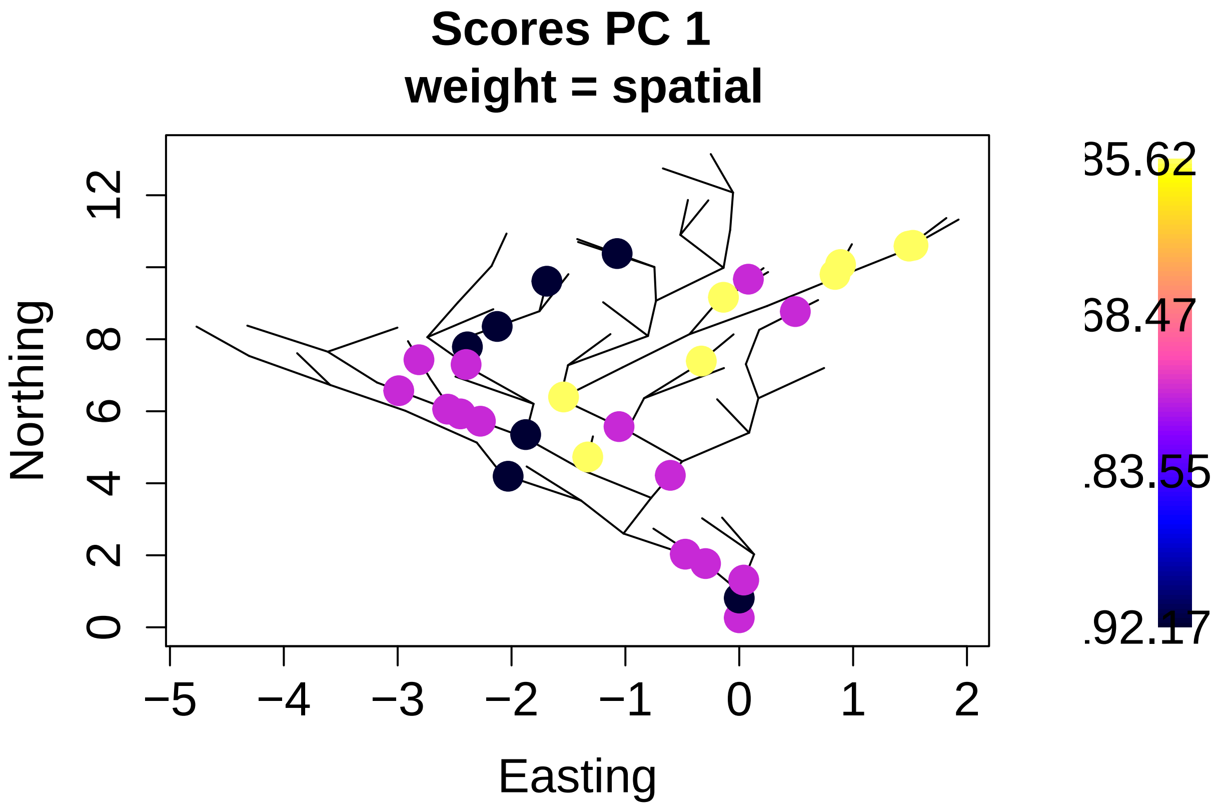
For T-mode PCA, we will produce the plots for weighted as well as unweighted PCA. The functions are all called in the same way but remember that for T-mode PCA, the loadings are related to time points while the principal components/scores are related to monitoring sites (this is the opposite way round from S-mode PCA).

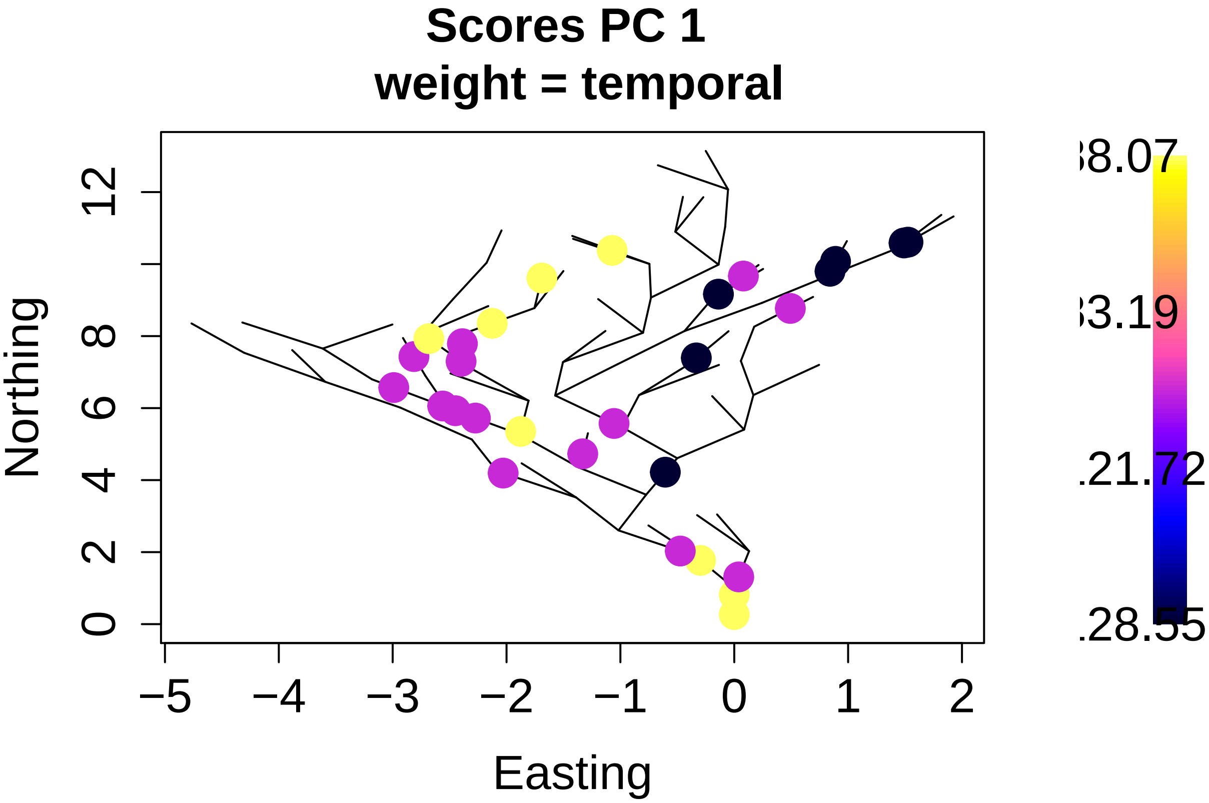
6.2.1 T-mode: map

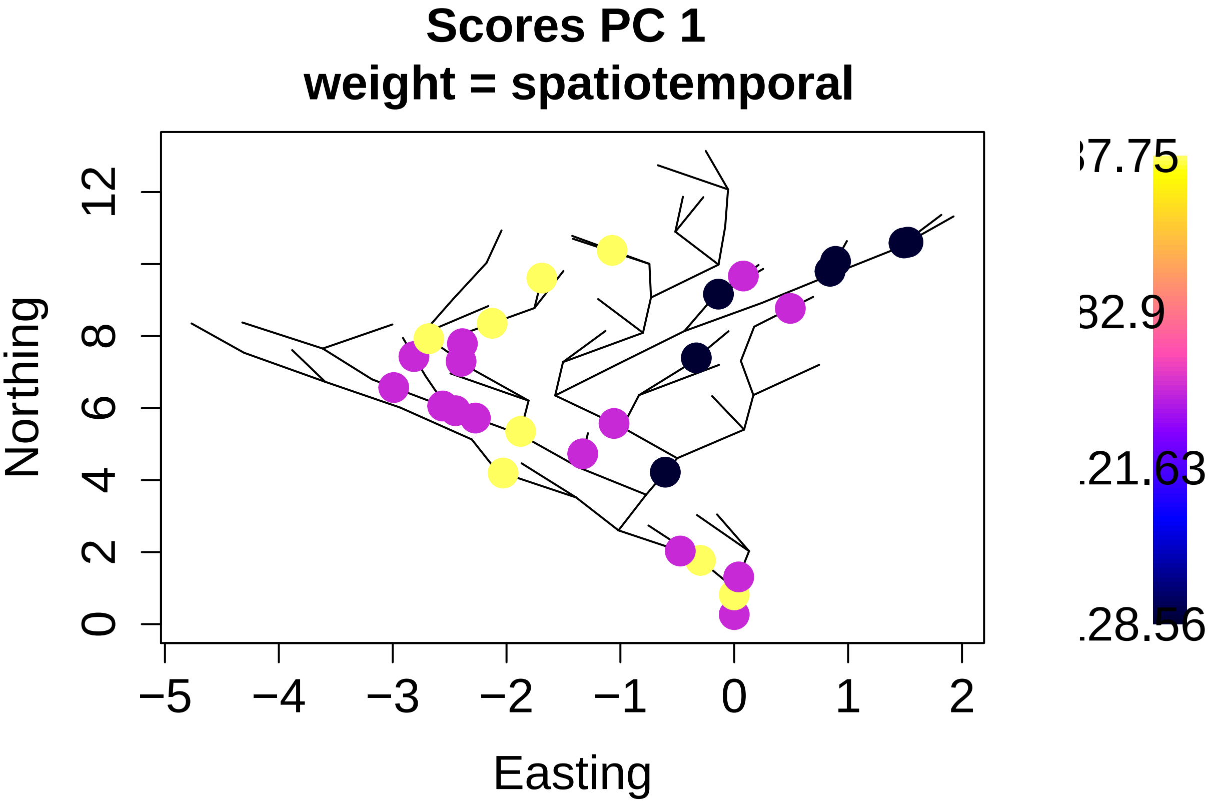
The map plot is suitable for plotting a single principal components scores (in T-mode) . As with S-mode, a river network shapefile is required, along with a matrix of two columns containing location coordinates for monitoring sites. The monitoring sites locations should be in the same order as monitoring sites in the dataframe used as input for `stpca()` to ensure that results are plotted at the correct location.

```r
plot_stpca(x = Tmode.pca.all, plots = "map", river = river.dat, coords = coords)
```


Scores PC 1
weight = unweighted
Northing
Easting
0
2
4
6
8
12
−5
−4
−3
−2
−1
0
1
2
35.62
68.48
.83.53
.92.13


Scores PC 1
weight = spatial
Northing
Easting
0
2
4
6
8
12
−5
−4
−3
−2
−1
0
1
2
35.62
68.47
.83.55
.92.17


Scores PC 1
weight = temporal
Northing
0
2
4
6
8
12
−5
−4
−3
−2
−1
0
1
2
Easting
8.07
3.19
.21.72
.28.55

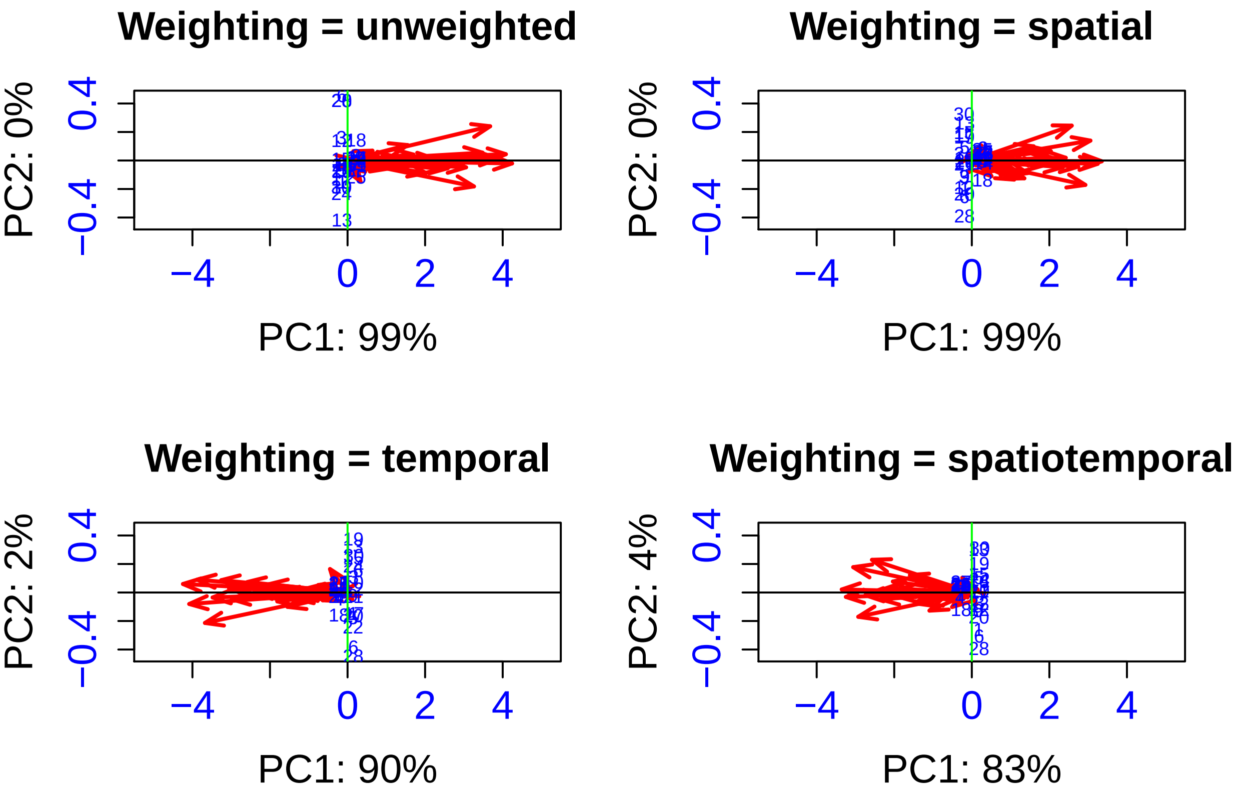


One important thing to consider when comparing plots for different weighting schemes is that including different weighting schemes can lead to loadings changing from positive to negative values and so although colours might change between different maps, this might be due to a change in sign rather than a change in magnitude.

6.2.2 T-mode: biplot

First produce the biplot for T-mode PCA:

```
plot_stpca(Tmode.pca.all, plots="biplot", biplot.factor=100)
```



In T-mode, the red arrows on the biplot are loadings related to time points and the blue points are the principal components/scores which are now related to monitoring sites. The principal components here are now a linear combination of observed values at several time points.

The biplot can be used here to look for time points with similar spatial patterns. If the spatial pattern of observed values changed over time (this is often found in climatology examples where weather patterns move in space) then you would expect to find two or more dominant PC's, and the spatial pattern represented by each PC was found at several time points, that can be identified as groups of arrows.

For example, the biplot on the top left (unweighted T-mode PCA) shows that the first PC accounts for 99% of the variability in the data. This suggests that the spatial pattern of observed values remained stable over the full time period. The arrows are closely grouped together and so we can say that the spatial pattern represented by PC1 is the average spatial pattern over all time points. The blue points show monitoring sites on either side of the vertical green line. The monitoring sites on the same side of the green line as the arrow heads are monitoring sites where the observed values were high at all time points, while the monitoring sites on the opposite side of the line from the arrow heads indicates monitoring sites with lower values at all time points.

Looking at the other biplots, we can see that adjusting for temporal correlation has had a greater effect on the results than adjusting for spatial correlation, since the percentage of variance explained by the first PC reduces more when PCA has been adjusted for temporal correlation. One explanation for this is that the temporal weights used are removing some of the structure in the data (by removing correlation between columns/time points) and removing correlation makes the columns independent. In PCA, if all n columns in the dataframe are independent then each PC will explain $100 * \frac{1}{n}\%$ of the total variance in the data. This means that if all columns are completely correlated then one PC will explain 100% of the variance in the data.

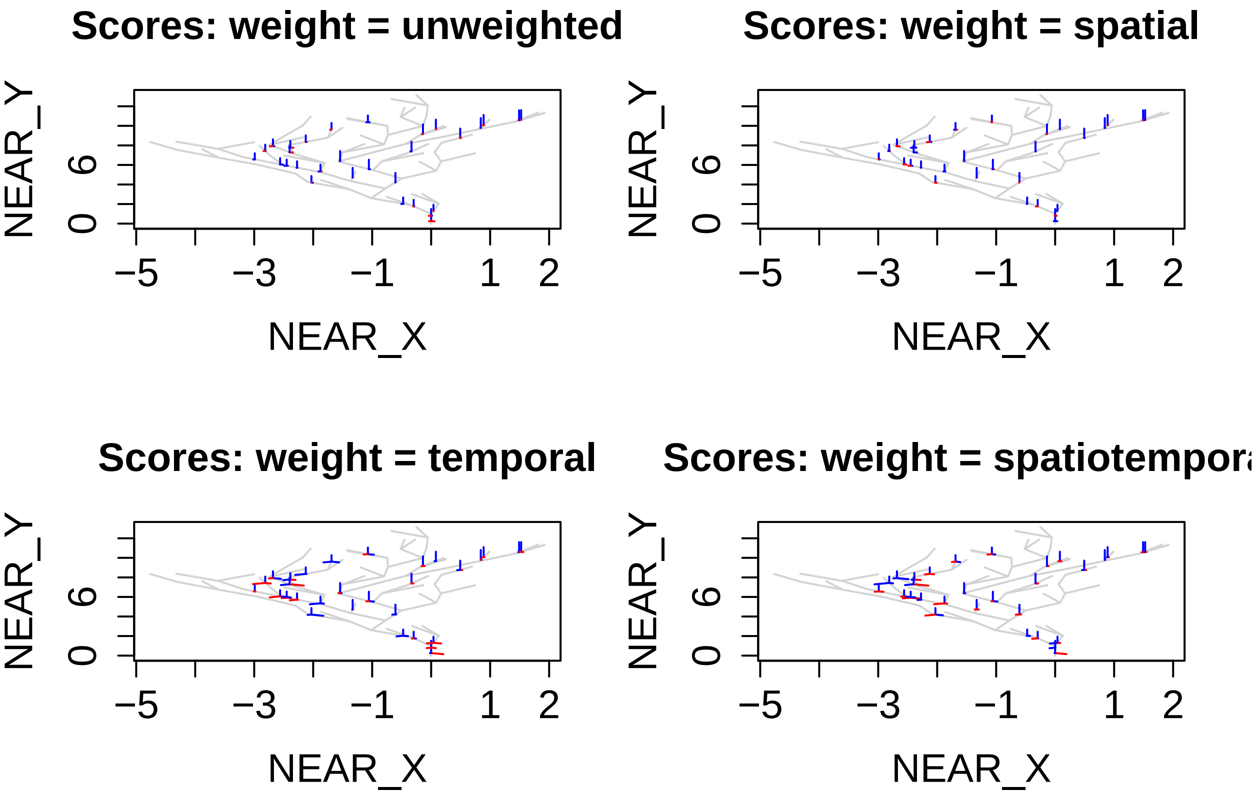
Notice that for weights = “temporal”" and weights = “spatiotemporal”, the signs of the loadings have changed for PC1 compared to weights = “unweighted” or weights = “spatial”, but the loadings are still of similar.

6.2.3 T-mode: glyph plot

The glyph plot for T-mode PCA will plot the principal component scores as glyphs, rather than the loadings which were plotted for S-mode PCA.

```
plot_stpca(x = Tmode.pca.all, plots = "glyph",
           river = river.dat, coords = coords)
```

```
## For plots = glyph: did you check that coordinates for monitoring site ID's are in
## the same order as data used for stpca()?
## If not then glyphs might not correspond to the correct monitoring site.
```



The glyph plots for T-mode PCA applied to the demo data show that when temporal correlation is accounted for, the second and third PC's capture spatial patterns in the west of the river network that were not visible in the unweighted PCA.

6.2.4 T-mode: time series and meanPM plot

The time series plot (`plot = "ts"`) and mean time series plus/minus scores plot (`plot = "meanPM"`) should not be used to display the results of T-mode PCA. In S-mode PCA, the time series plot shows temporal patterns captured by the principal components, as well as variation around the mean captured by each principal component, but in T-mode PCA these plots would display the loadings (rather than scores in S-mode). This does not have any clear interpretation and if you try to use either of these plots for T-mode PCA a warning message will appear and no plots will be produced:

```r
plot_stpca(x = Tmode.pca.uw, plots = "ts")
```

```
## For plots = ts: ts plot should only be used when pca.mode = Smode
```

```r
plot_stpca(x = Tmode.pca.uw, plots = "meanPM")
```

```
## For plots = meanPM: meanPM plot should only be used when pca.mode = Smode
```

7 Troubleshooting

7.1 `completeData()`

Sometimes the following error message appears and the missing values cannot be imputed:

```
Error in eigen(crossprod(X, X), symmetric = TRUE) :    infinite or missing values in 'x'
```

This happens if one or more columns in your dataframe consist only of NA's. This can occur if you aggregate irregularly sampled data to monthly averages, for example. If this happens, you could

1. delete the empty columns, although this is only sensible if all of the empty columns are grouped together at the beginning or end of the time period.
2. replace at least one value in each column with a sensible value such as the row mean. Do not replace many of the missing values in this way or you can introduce bias into the imputation method. Replace one or two missing values in each column and then let the `completeData()` function estimate the rest.
3. change the `pca.mode`. For example, if you were trying to apply `completeData()` in `"Tmode"` but find you have empty columns, change `pca.mode` to `"Smode"` (assuming this does not also result in empty columns). This means that you are imputing missing values based on correlation between monitoring sites rather than correlation between time points. It is up to the user to decide if this approach is suitable, depending on the aim of the analysis being performed.

7.2 `stpca()`

7.2.1 Function is slow

This function can take a few minutes to run. This is due to calculating the matrix square root of the weights matrices and the time taken for these calculations increases as the size of the dataframe increases. The matrix square root means that matrix $\mathbf{A} = \mathbf{aa} = \mathbf{a}^2$, or in R: `a %*% a`, as opposed to the elementwise square root where the square root of each entry in the matrix is calculated.

7.2.2 Note appears when `pca.wt = "temporal"` or `pca.wt = "spatiotemporal"`

Sometimes the following Note appears:

```
Note: method with signature 'symmetricMatrix#missing' chosen for function 'Schur',  target
signature 'dsyMatrix#missing'.  "dsyMatrix#ANY" would also be valid
```

There is nothing to worry about if this appears - it is not an error or warning and is only giving some information about options that were used when calculating the matrix square root of "temporal.wt".

References

Husson, Francois, and Julie Josse. 2015. *MissMDA: Handling Missing Values with Multivariate Data Analysis.* https://CRAN.R-project.org/package=missMDA.

Peterson, E. E., and J. M. ver Hoef. 2010. “A Mixed-Model Moving Average Approach to Geostatistical Modeling in Stream Networks.” *Ecology* 91: 644–51.

Peterson, Erin E, and Jay M ver Hoef. 2014. “STARS: An ArcGIS Toolset Used to Calculate the Spatial Information Needed to Fit Spatial Statistical Models to Stream Network Data.” *J Stat Softw* 56 (2): 1–17.

Richman, M. B. 1986. “Rotation of Principal Components.” *Journal of Climatology* 6 (3): 293–335.