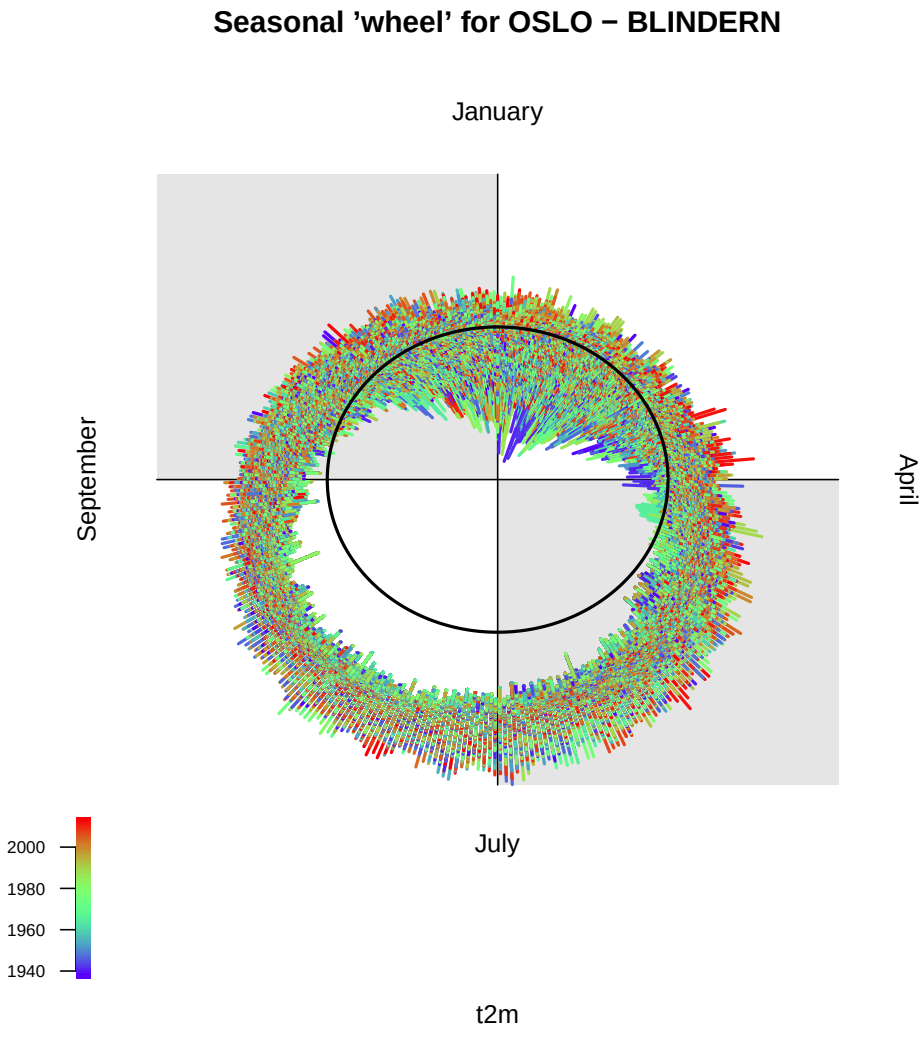
Norwegian Meteorological Institute

MET report

no. 11/15
Climate

'esd' - The Empirical-Statistical Downscaling tool & its visualisation capabilities.

Rasmus E. Benestad, Abdelkader Mezghani, & Kajsa M. Parding



Norwegian Meteorological Institute

MET report

Title	Date
`'esd'` - The Empirical-Statistical Downscaling tool & its visualisation capabilities.	June 19, 2015
Section	**Report no.**
Model and Climate Analysis	11/15
Author(s)	**Classification**
Rasmus E. Benestad, Abdelkader Mezghani & Kajsa M. Parding	● Free ○ Restricted
Client(s)	**Client's reference**
Climate community	

Abstract

We present a tool, the R-package `'esd'`, made freely available by the Norwegian Meteorological Institute (MET Norway) for use by the climate community. It was primarily built for empirical-statistical downscaling of climate information and has been extended to search, process, dissect, and analyse meteorological and climatological data (local as well as global climate data sets). It consists of i) retrieving and manipulating large samples of meteorological, climate and model data from various sources, ii) searching, dissecting and displaying the information in the data, (iii) and computing a range of analyses. The acronym `'esd'` can be associated with both 'Easy & Simple Data' processing and 'Empirical-Statistical downscaling'. It provides simple and intuitive ways for reading data from weather station, gridded data sets, as well as trajectory data such as cyclone paths. The philosophy behind its design has been to reduce the time spent on coding, reformatting data, testing the code, or looking up the manuals for correct syntax. The functions are designed to be intuitive and easy to remember in addition to being efficient.

Keywords

Data retrieval, analysis, visualisation; empirical-statistical downscaling models, diagnostics, evaluation, R package, climate data.

Disciplinary signature
Jan-Erik Haugen

Responsible signature
Øystein Hov

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no
www.yr.no

Contents

1 Introduction **1**

1.1 The need for climate information . . . 1
1.2 Motivation . . . 1
1.3 History . . . 2
1.4 The new tool: `'esd'` . . . 3

2 Documentation **5**

2.1 Retrieving data: I/O . . . 5
2.1.1 Function 'retrieve' . . . 6
2.1.2 Function 'station' . . . 6
2.1.3 Quick search - Function 'select.station' . . . 7
2.2 Data structures . . . 9
2.3 Function 'summary' . . . 10
2.4 Data visualisation . . . 14
2.4.1 Function 'plot' . . . 14
2.4.2 Function 'map' . . . 14
2.4.3 Function 'vec' . . . 14
2.4.4 Info-graphics . . . 18

3 Data handling & processing **21**

3.1 Formulas & functions . . . 21
3.1.1 Small handy functions . . . 21
3.1.2 Wet-means, frequencies, counts and spells . . . 21
3.2 re-gridding . . . 25
3.2.1 How the re-gridding works . . . 25
3.3 Nearest data point . . . 26
3.4 Subsetting . . . 26
3.5 Combining and synchronising . . . 28
3.6 Anomalies . . . 28
3.7 Aggregate: monthly, seasonal and annual statistics . . . 29
3.8 Spatial averaging of field objects - aggregate.area . . . 29
3.9 Transformations and conversions: as. . . . 33

4 Analysis & Diagnostics **35**

4.1 Empirical Orthogonal Functions . . . 35
4.2 Principal Component Analysis . . . 35
4.3 Canonical Correlation Analysis . . . 36
4.4 Other types of analysis . . . 36
4.5 Predict & project . . . 36
4.6 Trajectory objects . . . 40

5 Downscaling **45**
5.1 Defining predictands . . . 45
5.2 Defining predictors . . . 45
5.2.1 Tuning . . . 46
5.3 Options for downscaling . . . 47
5.3.1 Downscaling for single station and a single model . . . 47
5.3.2 Downscaling a group of stations - PCA . . . 47
5.3.3 Downscaling an ensemble of climate models . . . 47
5.4 Downscaling of trajectory objects . . . 48
5.5 Downscaling probabilities and number of events . . . 53
5.6 Weather generators - synthesising daily time series . . . 53
5.6.1 Stochastic precipitation . . . 54
5.6.2 Stochastic temperature . . . 54

6 Evaluation, assessment & validation **55**
6.1 Central limit theorem . . . 55
6.2 Huth's dilemma . . . 55
6.3 Non-stationarity check . . . 55
6.3.1 iid.test . . . 55
6.4 Diagnose . . . 56
6.4.1 Station . . . 57
6.4.2 Combined fields . . . 57
6.4.3 Common EOFs . . . 57
6.5 Downscaled results . . . 57
6.5.1 Cross-validation . . . 57
6.5.2 Residuals . . . 58
6.5.3 Downscaled ensemble results . . . 58

7 Trouble shooting **59**
7.1 General . . . 59
7.2 Functions `'annual'` and `'aggregate'` . . . 59
7.3 DSensemble . . . 60
7.3.1 Poor fit . . . 60
7.3.2 Other error messages . . . 60
7.4 Validate . . . 61

8 Summary **61**

9 Acknowledgement **62**

1 Introduction

1.1 The need for climate information

A call for climate services and other initiatives and programmes to provide climate information to stake holders and the general public has been made over the recent decades (e.g. the WMO global framework for climate services (GFCS), the CORDEX program, and the Joint Programming Initiative (JPI-Climate)). The urgency of their mission has been emphasised by recent extreme weather events and the publication of a number of climate research reports (e.g. *Hov et al.* (2013); *Field et al.* (2012); *Stocker et al.* (2013)).

The development of efficient and versatile tools for climate analysis are essential to this effort. For instance, there is always a need for extracting relevant climate information, reading data, and testing and visualising the results. The present 'esd' tool has been developed to meet these requirements.

The `'esd'` tool is being used in different projects such as FP7 SPECS and CLIPC[1], COST-VALUE[2], INDICE[3], CixPAG[4], MIST-II [5], CLIMATRANS[6], and EU-CIRCLE [7] to establish networks and standards for assessment and validation of model results needed for climate services.

1.2 Motivation

Open, efficient, flexible, and intuitive tools making use of state-of-the-art statistical know-how have long been needed for the purpose of climate analysis. Often the lack of resources (man power) dedicated to such work limits the possibilities of offering tailor-made user-relevant information. There is a wide range of requirements, from climate data handling and analysis, and validation/diagnostics to bias adjustment of model results and downscaling (including prediction and projection). Even if the main purpose may be just to carry out empirical-statistical downscaling (ESD; *Benestad et al.* (2008)), many other tasks are necessary to prepare the analysis and understand the results. The need to analyse multi-model ensembles rather than single model (*Benestad*, 2011) also makes additional demands on the tools. In other words, `'esd'` is a tools for 'Big data' climate analysis.

[1] CLIPC will provide access to climate information of direct relevance to a wide variety of users, from scientists to policy makers and private sector decision makers.

[2] The COST Action VALUE (2012-2015) aims at providing a European network to develop downscaling methods, validate them, and improve the collaboration between the research communities and stakeholders.

[3] INDICE is a collaboration between Norway and India that studies the hydrological consequences of climate change on the Hindu-Kush region in Himalaya

[4] CiXPAG project aims at investigating the complex interactions between climate extremes, air pollution and agricultural ecosystems.

[5] This project aims at providing reliable projections to be used by STATKRAFT as a leading company in hydro-power internationally and Europe's largest generator of renewable energy.

[6] CLIMATRANS project aims at investigating the complex interactions between climate extremes, air pollution and transportation in Indian mega-cities.

[7] EU-CIRCLE project aims at developing a Climate Infrastructure Resilience Platform (CIRP) as an end-to-end modelling environment.

An important aspect of any climate analysis tool is data input-output (I/O) and handling data. Although model results are meant to be provided in a standard format (e.g. the Coupled Model Inter-comparison Project[8] - CMIP - following the 'Climate and Forecast' (CF) convention), experience shows that there are often differences that can cause problems. For instance, the CMIP5 ensemble of models involves 4 different types of calendars.

Data files are often organised in accordance with some kind of Common Information Model[9] (CIM) or a Data Reference Syntax[10] (DRS), but there is also a benefit in standard structures in the working computer memory when using programming environments such as R. A well-defined and standardised data structure makes it possible to create a generic climate analysis tool.

Searching for available and complete observational data sets is always a difficult and time consuming task without prior knowledge about the quality of the retrieved data sets. In the context of empirical-statistical downscaling, observations are needed both for validation of model results and for calibrating models. Furthermore, there may be a need to aggregate annual mean values rather than daily or monthly, and to extract subsets, for instance a given calendar month, a limited interval, or a smaller spatial region.

The purpose of the `'esd'` package is to offer an R-package that covers as many as possible of these requirements, from acquiring data and converting to a standardised format, to performing various statistical analyses, including statistical downscaling, all in a simple and user-friendly way.

1.3 History

Most of the previous work on ESD carried out at MET Norway has been based on the R-package `'clim.pact'` (*Benestad et al.*, 2008) from the early 2000s. This tool had evolved into a large set of methods and functions over time, but without a 'strict discipline'. Hence, the functions would use a range of different set-ups and lacked a streamlined finish. Therefore, it was not very user friendly. The idea is that any climate data analysis should be quick, easy, and efficient, with a bare minimum effort going into coding and reformatting of data sets. It should be intuitive and easy to remember how to perform the analysis. This way, more of the time can be devoted to do the analysis itself and writing reports, rather than rewriting code, debugging, testing, and verifying the process. While much of the methods needed in climate analysis were included in `'clim.pact'`, it was evident that the most efficient solution was to create a new R-package from scratch with a more purposeful and well-defined data structure. Much of the methods, however, are based on functions in `'clim.pact'`, although the benefit of experience from developing that package has been utilised from the start of the creation of `'esd'`.

[8] http://cmip-pcmdi.llnl.gov/

[9] `https://www.earthsystemcog.org/projects/downscalingmetadata/`

[10] `http://cmip-pcmdi.llnl.gov/cmip5/output_req.html`

Since it is creation, the 'clim.pact' package has been widely used by the climate community for research purposes and has been a valuable asset for assessment studies (*Winkler et al.*, 2011). For instance, *Spak et al.* (2007) compared statistical and dynamical downscaling of surface temperature in North America using results obtained from the 'clim.pact' tool, and *Girma et al.* (2010) used the 'clim.pact' tool to downscale rainfall in the upper Blue Nile basin for hydrological modelling.

The R-package was used by *Schöner and Cardoso* (2005) to downscale air temperature mean and precipitation sums from regional model as well as directly from ERA-40 fields, and *Tyler et al.* (2007) used it to produce temperature scenarios by downscaling output from 17 different global climate. However, *Estrada et al.* (2013) pointed out that 'clim.pact' and similar tools *do not provide the necessary tests to ensure the statistical adequacy of the model and therefore misleading results could be expected.* We will address some of the issues that *Estrada et al.* (2013) highlighted later on.

1.4 The new tool: 'esd'

The new tool 'esd', like its predecessor 'clim.pact', is based on a 'plug-and-play Lego principle', where data are seen as an object that is used as input and new results are returned as output following a standard data structure. This means that different functionalities can be combined where the output from one process is used as input to the next. In 'esd' the data structure has changed from that in 'clim.pact', and uses more attributes and less list objects. Furthermore, the data objects build on the time series object 'zoo', which takes care of much of the chronology. A consequence of using 'zoo' as a basis also is that it implies adopting S3-methods, which also makes the tool more flexible and user-friendly. Through the S3-methods, new plotting methods have been made available for station and field objects, empirical orthogonal functions (EOFs), and other types. The use of S3-methods implies the definition of new classes describing the data types. It can also make the tool work more seamlessly with other R-packages and other structures if appropriate conversion tools are made.

The development of the 'esd' software fits in with the trend of the R-language's increasing role in the climate change debate (*Pocernich*, 2003) and as an open science platform (*Pebesma et al.*, 2012). Furthermore, both R and the 'esd' R-package are valuable tools for linking high education and research (*IBARRA-BERASTEGI et al.*). The 'esd'-package has already been used as material for capacity building in connection with the bilateral INDICE project between Norway and India (http://www.nve.no/en/Projects/INDICE/), and the CHASE-PL project between Norway and Poland
(http://www.isrl.poznan.pl/chase/index.php/project/).

The wide range of functionalities included in the 'esd' tool makes it suitable for processing results from global and regional models. The tool also includes methods for plotting and generating various info-graphics: time series, maps, and visualisation of complex information.

The data processing aspects include re-gridding, sub-setting, transformations, computing empirical orthogonal functions (EOFs) and principal component analysis (PCA) for stations and gridded data on (ir)regular grids, regression, empirical-statistical downscaling, canonical correlation analysis (CCA) and multi-variate regression (MVR). The tool also offers several diagnostic methods for quality assessment of the data and downscaled results. The `'esd'` methods can be tailored to meet with specific user needs and requirements.

As the library was built for the R computing environment (*R Core Team*, 2014), it inherits from the large number of R built-in functions and procedures. It also includes an additional set of predefined objects and classes, sample and structured data and meta-data, and uses the S3-methods to ensure that information is appropriately maintained. Finally, the `'esd'` library has been built with the emphasis of traceability, compatibility, and transparency of the data, methods, procedures, and results, and is made freely available for use by the climate community and can be installed from the MET Norway Github account (`https://github.com/metno/esd`) or Figshare (`http://dx.doi.org/10.6084/m9.figshare.1160493`).

The remainder of this report describes technical issues related to the `'esd'` tool. A listing of the syntax and examples are also included in this report.

2 Documentation

A short and single line in R produces a complex figure with various information: the generic 'plot' yields a plot as seen in the left panel of Figure 4 or can make a graphical presentation of downscaled results that both shows the numbers as well as their quality. How is that possible? The trick is to define different data object types, known as 'classes' in R and define a specific data reference syntax (DRS) or common information model (CIM) that includes the meta-data in the working computer memory as well as in files stored on discs. This is all done automatically on-the-fly behind the scene, so that the user does not have to worry about these matters (it is of course possible to change the meta-data to e.g. correct for potential errors).

The command `library('esd')` *must be given at any new open R session.*

The R-package can be installed directly from Github (`https://github.com/metno/esd`) or Figshare for Mac/Linux[11] and Windows[12])

2.1 Retrieving data: I/O

There are different types of data that can be handled by the 'esd' tool: station data, gridded data, and (storm) trajectories. Station data contain meteorological observations recorded at weather (or hydrological) stations, while gridded data can comprise various analyses (e.g. E-OBS gridded version of the European and Climate Assessment data set), reanalyses (e.g. NCEP, ERA, ...) or global/regional climate model results (e.g. CMIP3/5 experiment). Trajectories are mainly used for analysis of storm tracks (e.g. IMILAST[13])

There are two main methods for retrieving data in the 'esd' tool: 'station' and 'retrieve'. It is also possible to read data using the R-built in functions and convert it to esd data format. This requires more effort from the user using the 'esd' pre-defined functions 'as.station', 'as.field'), and 'as.trajectory'.

The package comes with a set of sample data mainly included for demonstration purposes, testing, or troubleshooting. For gridded data, these are filtered data (mainly for reducing the size of the data objects) and **should not be used as predictors** *in the actual analysis.*

For instance, air temperature reanalysis data are stored as a set of 20 empirical orthogonal functions (EOFs) with global coverage, which are then transformed to a field object upon a retrieval. However, the sample station data can be used without restrictions and corresponds to observed values at a specific location (e.g. 'data(Oslo)' or 'nacd=station(src='nacd')').

[11] `http://figshare.com/articles/esd\_for\_Mac\_amp\_Linux/1160493`
[12] `http://figshare.com/articles/esd\_for\_windows/1160494`
[13] http://www.proclim.ch/imilast/index.html

2.1.1 Function 'retrieve'

The `'retrieve'` is designed to read data from NetCDF files following the standard Climate and Forecast ('CF') conventions and ordered on a longitude-latitude grid. The latter could be regular or irregular grid (e.g. the output of the Weather Research and Forecasting model (WRF) or any outputs on a rotated grid from RCMs). The function `'retrieve'` also performs quick checks of the data itself to verify that the meta data and data are consistent. For instance, in case the frequency of the data is missing, `'retrieve'` will try to detect the frequency automatically form the data itself. The function `'retrieve'` returns two types of objects depending on the type of the spatial grid. For instance, data stored on a regular grid is returned as `'field'` objects including attributes containing meta data, and data stored on irregular grid - such as rotated longitude-latitude grids - are returned as a `'station'` objects. `'retrieve'` has also been adapted to read global climate data from the CMIP3/5 experiments, most of the global reanalysis such as those provided by the European Centre for Medium-Range Weather Forecasts (ECMWF) known as ERA-40 and ERA-INTERIM, the National Center for Environmental Prediction (NOAA) known as NCEP/NCAR reanalysis, the Japan Meteorological Agency (Japanese Reanalysis JRA-25,55), and the NASA GSFC Global Modeling and Assimilation Office (GMAO) known as MERRA reanalysis. A full overview of all available reanalysis can be found at `http://reanalysis.org/atmosphere/overview-current-reanalyses`.

As for now, `'retrieve'`, does not display a list of missing attributes that are mandatory for further post-processing of the data. The user must add the missing attributes manually.

The strength of `'retrieve'` is that it can read and return formatted objects with common attributes for post-processing, visualising and making outputs comparable (e.g. re-gridding the field objects into the same grid resolution). Basically, all reanalysis, general circulation models (GCMs), and regional climate models (RCMs) can be read using the `'esd'` tool and further combined into one object, or analysed, albeit with some limitations due to the size of the returned object.

2.1.2 Function 'station'

The package `'esd'` includes the function `'station'` for obtaining historical climate data by querying various web portals, for instance, the MET Norway archive (KDVH) provided by the Norwegian Meteorological Institute[14]. Data from MET climate web service 'eKlima' needs to be adapted manually, the Global Historical Climate Network (GHCN, (*Peterson and Vose*, 1997)) provided by the American National Climatic Data Center[15], and the European Climate Assessment and Dataset (ECA&D, (*Klein Tank et al.*, 2002)) made available by the Royal Netherlands Meteorological Institute (KNMI) (http://eca.knmi.nl/). Some of the data that is

[14] http://eklima.met.no; however, this function only works within the firewall

[15] http://www1.ncdc.noaa.gov/pub/

included in the package has been pre-formatted within the ‘clim.pact’ package and adapted to meet the new ‘esd’ data format requirements.

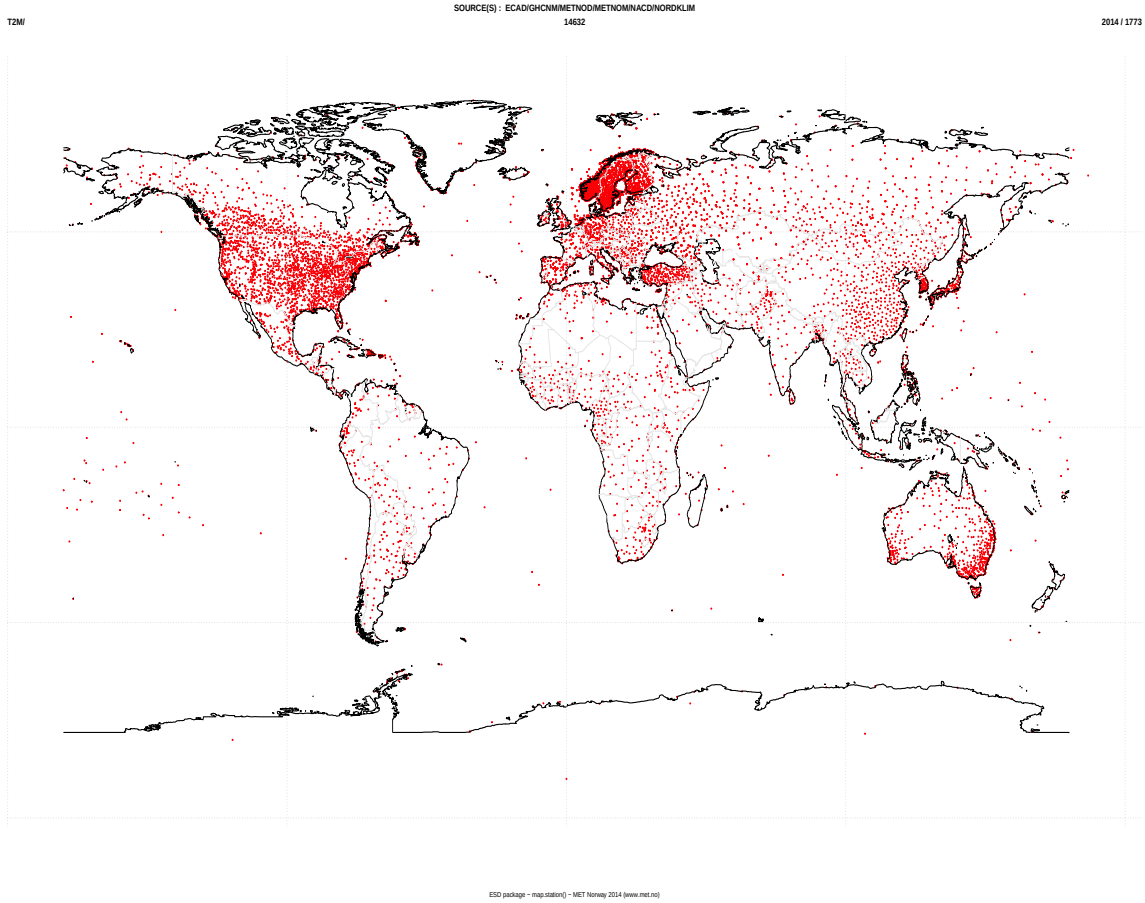


Figure 1: Map of available weather stations recording temperature that are included in the meta-data of the ‘esd’ package.

2.1.3 Quick search - Function ‘select.station’

The sample data includes also a meta-data object (‘stationmeta’) that can be loaded directly and contains meta data such as name of the ‘loc’*ation and its ‘id’*entification number, geographical coordinates such as ‘lon’*gitude, ‘lat’*itude, and ‘alt’*itude), ‘country’ name, ‘param’*eter name of recorded weather variables (e.g. temperature and precipitation), data source (‘src’) or provider for almost 100000 stations all over the world (Figures 1 and 2). These meta-data have been imported from existing meta data from the different data source providers. It has to be noted also that most of the available stations are managed by the World Meteorological Organisation (WMO) and made available by the American National Climate Data Centre (NCDC) for scientific research only. The meta data has been merged from the different sources mentioned earlier. Also, other additional data sources can be easily included into the package.

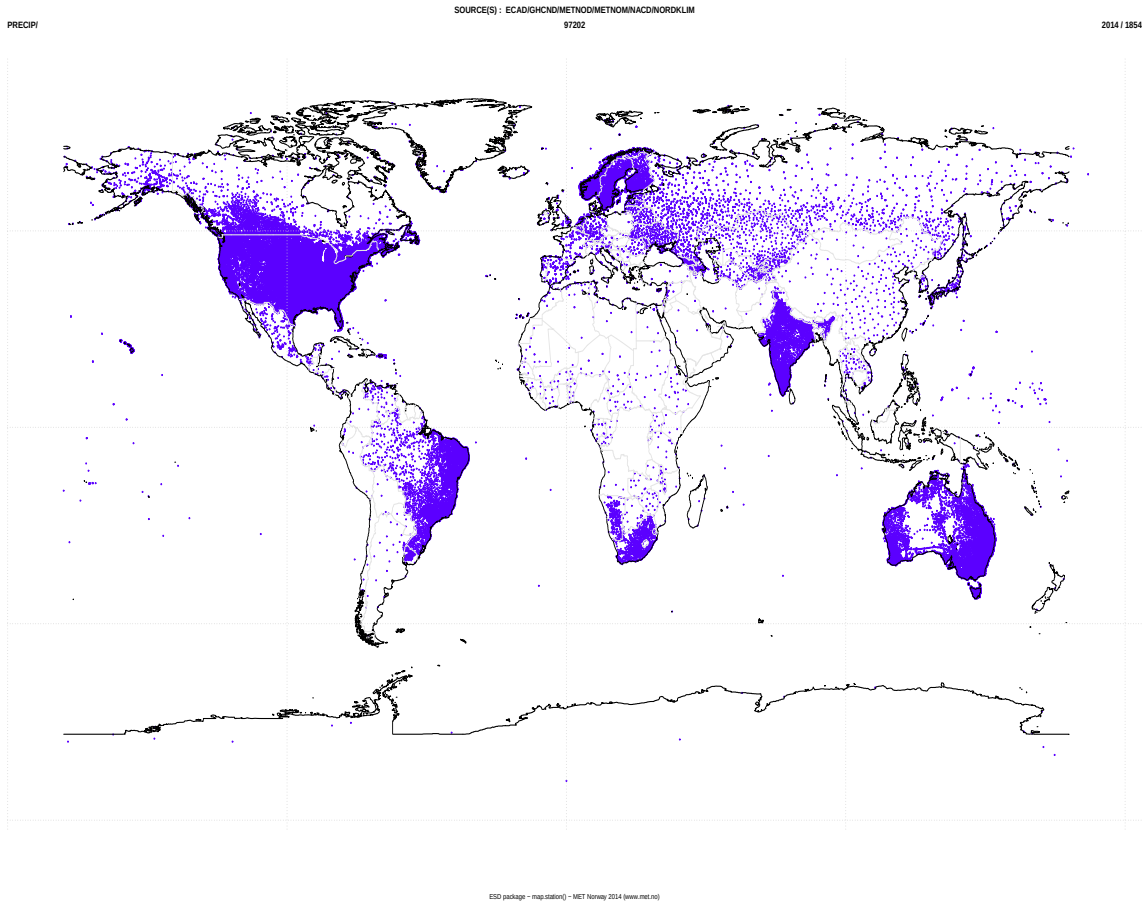


Figure 2: Map of available weather stations recording precipitation that are included in the meta-data of the `'esd'` package.

There are two ways of obtaining station data using `'station'` method. The first option, which we recommend, is to select a subset of stations from the meta data using the function `'select.station'` from a set of criteria. Among these criteria, the minimum length of the recorded data (`'nmin'`) can be specified to get, for instance, a subset of stations recording for at least a minimum number of (e.g. 100) years of data (e.g. `select.station(nmin=100)`). Thus, a subsample of the meta data is returned and can be checked for duplications of stations from the various sources. Afterwards, users can start downloading the data for the selected stations.

It has to be mentioned that the downloading process for the GHCN and ECA&D is different as the data is stored differently. For the GHCN data sets, each station is stored separately and a direct retrieval can be made. We highly recommend that users perform a prior selection of stations for the GHCN data sets before starting the downloading process. For the ECA&D data sets on the other hand, all stations are downloaded and stored locally as zip files at first call, after which data is extracted for the selection of stations. Other data sets, such as the NACD (*Frich et al.*, 1996), NARP (*Førland*), and Nordklim (*Tuomenvirta et al.*, 2001) are stored entirely in `'esd'` (only holds monthly values of a limited set).

The ‘station’ method can retrieve station data from a range of different sources, many over the web (GHCN and ECA&D). Example 2.1 demonstrates how to select, retrieve and plot temperature observations from a single station.

The ‘esd’ tool holds meta-data for various data sources, which makes it quick and easy to search for weather data recorded at stations according to the parameter, the geographical location (region, country, and coordinates (single or a range of longitude and latitude values) and altitude, and time interval.

2.2 Data structures

Most data objects handled by ‘esd’ are typically time series, and hence based on the ‘zoo’ class. The ‘zoo’ class extends the tseries class from regular to irregular time series. In the ‘esd’ tool, additional categories or classes have been defined dealing with temporal resolution distinguishing daily (‘day’), monthly (‘month’), seasonal (‘season’), and annual (‘annual’) data (as temporal classes) and spatial resolution distinguishing ‘station’ and ‘field’ classes. Data on sub-daily scales may be represented in the ‘day’ class but the ‘esd’ tool has not been tested yet for this time scale. The way R handles different classes and objects can sometimes be confusing. The first element in a list of different classes is used to identify the appropriate method to be used for that specific object. For instance, if a data object belongs to the classes (‘field’,‘zoo’), then, appropriate methods for ‘field’ objects are used rather than those for the ’zoo’ objects.

Different types of data objects are processed and handled differently, and the way ‘esd’ keeps track of the different types is through the so-called ‘S3-method’ and classes.

The ‘esd’ tool follows the same R programming functionalities and uses the first element of the class of an object to define the appropriate method to be used. The built-in R functions ‘class’ and ‘inherits’ are used to check whether an object inherits from any of the classes specified in the ‘esd’ package. Example 2.2 shows some of the classes used in ‘esd’.

The different data objects come bundled with relevant meta-data, stored as data attributes. These are shown using the ‘str’*ucture function, as displayed in Example 2.3 for a station object.

The various attributes have different functions, e.g. for handling the data, traceability, identification, and visualisation. The idea is that they are based on a standard terminology for which the terms are commonly agreed on and follow standard definitions.

A common core set of attributes will make it easier to share data and methods. The attribute ”history” contains the call(s), time stamp(s), and session info that have been used to create and process the data itself.

2.3 Function 'summary'

The S3 method '`summary`' has been extended to the classes defined in '`esd`' in order to provide more tailor-made information. Example 2.4 shows summary statistics for each calendar month of a daily station object.

Example 2.1.

```
# Select a station across India recording daily maximum temperature
# from the global historical climate network-daily
ss <- select.station(cntr='India',param='tmax',src='ghcnd')
ss.new <- subset(ss,subset=!duplicated(ss$location))
map(ss.new,cex=1.2,col="red",bg="pink",add.text=TRUE)
ss <- select.station(stid='IN022021900',cntr='india',param='tmax')
y <- station(ss)
#>[1] "Retrieving data ..."
#>[1] "1 TMAX IN022021900 NEW DELHI/S INDIA GHCND"
# Display the name of the location
loc(y)
# Highlights the location on the map
points(lon(y),lat(y),pch=19,col="blue",cex=2)
# aggregate daily values to annual values
ya <- annual(y,FUN='mean',nmin=100)
# Subset for the period 1970 to 2012
ya <- subset(ya,it=c(1970,2012))
# plot the time series including the error bar
plot(ya,ylim=c(29.5,32.5))
# Add the linear trend as
lines(trend(ya),col="red",lwd=2)
```

a) b)

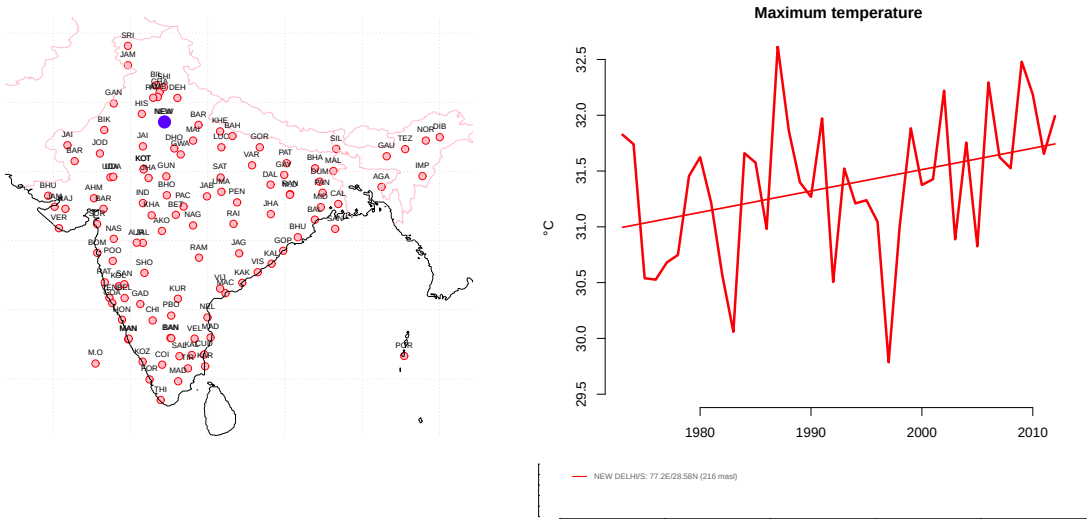


Figure 3: Map showing available stations across India from the GHCN-D dataset and b) a plot of the annual maximum temperature recorded at New Delhi weather station (blue point) including linear trend line.

Example 2.2.

```
# Example of monthly station data:
> data(Oslo)
> class(Oslo)
[1] "station" "month"   "zoo"
# Example of daily station data:
> data(ferder)
> class(ferder)
[1] "station" "day"     "zoo"
# Example of annual station data:
> class(annual(ferder))
[1] "station" "annual"  "zoo"
# Example of a field object
> t2m <- t2m.NCEP(lon=c(-30,30),lat=c(40,70))
> class(t2m)
[1] "field" "month" "zoo"
> class(EOF(t2m))
[1] "eof"   "field" "month" "zoo"
```

Example 2.3.

```
# Load the data for Ferder weather station
data(ferder)
# Display the structure of the data as
> str(ferder)
'zoo' series from 1900-01-01 to 2013-12-04
  Data: atomic [1:41611] 2.1 -1.8 -0.9 -3 -7.2 -6.5 -2.6 -2.4 -1.6 -0.3 ...
- attr(*, "location")= chr "Ferder lighthouse"
- attr(*, "station_id")= num 27500
- attr(*, "wmo_id")= num 1482
- attr(*, "longitude")= num 10.5
- attr(*, "latitude")= num 59
- attr(*, "altitude")= num 6
- attr(*, "variable")= chr "t2m"
- attr(*, "unit")= chr "deg C"
- attr(*, "country")= chr "Norway"
- attr(*, "source")= chr "MET Norway eklima"
- attr(*, "long name")= chr "daily mean temperature"
- attr(*, "URL")= chr "http://eKlima.met.no"
- attr(*, "calendar")= chr "gregorian"
- attr(*, "quality")= logi NA
- attr(*, "aspect")= chr "original"
- attr(*, "type")= chr "observation"
- attr(*, "reference")= chr "MET Norway climate archive"
- attr(*, "info")= logi NA
- attr(*, "method")= chr "mean estimated from thermometer measurements"
- attr(*, "history")=List of 3
..$ call       : chr "1 0"
..$ timestamp  : chr "Fri Dec 13 15:51:49 2013"
..$ sessioninfo:List of 3
.. ..$ R.version  : chr "R version 3.0.2 (2013-09-25)"
.. ..$ esd.version: chr "esd_0.2-1"
.. ..$ platform   : chr "x86_64-pc-linux-gnu (64-bit)"
Index:  Date[1:41611], format: "1900-01-01" "1900-01-02" "1900-01-03" "1900-01-04" ...
```

Example 2.4.

```
## Load data for Ferder
> data(ferder)
## Display the summary of statistics
> summary(ferder)
     Min. 1st Qu. Median    Mean 3rd Qu. Max. NA's
Jan -21.2  -3.100    0.0 -0.3226   2.900  9.0   13
Feb -17.1  -3.400   -0.6 -0.9718   2.000  7.7   25
Mar -12.9  -0.700    1.2  0.9619   3.025 11.7   14
Apr  -6.0   3.200    4.8  4.8250   6.400 13.8   NA
May   1.0   8.100   10.0 10.1000  12.000 20.3   NA
Jun   6.3  12.800   14.2 14.5200  16.100 24.2   NA
Jul  10.6  15.500   16.9 17.0500  18.400 24.9   NA
Aug  10.0  15.100   16.3 16.5600  17.800 24.8    2
Sep   3.6  11.900   13.4 13.2700  14.800 20.7   NA
Oct  -1.3   7.000    9.4  8.9340  11.100 15.4    8
Nov  -7.3   2.375    4.8  4.5730   7.200 16.3   24
Dec -15.6  -1.200    1.9  1.5340   4.700  9.7   29
```

Table 1: Data objects in `'esd'` are determined by a set of classes, listed in this table. This may be extended in the future to include radiosonde and radar data objects.

`'station'`	Class defining station objects. Can be single or multiple stations with daily, monthly, seasonal or annual temporal resolution.
`'spell'`	Looks similar to the `station` class, but the events are irregularly spaced and contains both duration of wet/hot as dry/cold spells. The distinction also enables `'esd'` to apply different plotting and analysis methods than those for regular stations.
`'field'`	Currently represents time series of 2D variables, but may in principle contain any number of spatial dimensions.
`'eof'`	Class defining an EOF describing the spatial patterns (EOFs), the temporal variations (PCs), and the eigenvalues.
`'pca'`	Class defining a PCA is similar to the `eof` class, but allows for irregular grid of stations.
`'cca'`	Class that defines the results of a CCA, containing a pair of patterns and the canonical correlations.
`'ds'`	Class for DS results.
`'dsensemble'`	Class for downscaled ensembles.
`'diagnose'`	Class for diagnostic results.
`'trajectory'`	Class for trajectories.
`'xval'`	Class for cross-validation.
`'xsection'`	Class for cross-sections (Hovmuller diagrams).
`'mvr'`	Class for multivariate regression (MVR) objects, which hold the matrices that maps one data set onto the data space of another.

2.4 Data visualisation

The main data visualisation is provided through `plot` and `map` methods, but there are other more specific methods for producing additional graphs.

2.4.1 Function 'plot'

The `'plot'` method in `'esd'` extends the S3 plot methods from package 'graphics' to new `'esd'` classes (Table 1). `'plot(x)'` and `'plot.station(x)'` are equivalent if `'x'` is an object of class `'station'`.

Various plotting outputs are generated depending on the class of the objects used as inputs. For instance, the `'plot.station(x)'` function produces a figure based on the default `'graphics'` and `'zoo'` plots but adapted for the station object (Figure 4a).

For some classes, plot can produce several or a combination of plots giving more description of the output. The argument `'plot.type'` is also used to distinguish between single or multiple plots in one window. The `'plot'` function also inherits all graphical parameters from `'par'` with additional parameters used by `'esd'`. An example of the function `'plot'` applied to a station object is shown in Example 2.5 and Figure 4a.

Although the plot itself gets more complicated for EOFs, the syntax remains as simple as for the station object (Example 2.6, Figure 5).

2.4.2 Function 'map'

The function `'map'` is also an S3 built method and used to produce a map of geographical and geophysical data points and gridded data. `'map'` can be seen as a spatial plotting method, while plot is mainly used for time series. Unlike `'plot'`, `'map'` is proper to `'esd'` and is an S3 method and do not extend the existing `'map'` function from package `'maps'` (http://CRAN.R-project.org/package=maps), which too works for the different `'esd'` objects. When applied to one single station, `map` plots its location (Example 2.5, Figure 4b).

2.4.3 Function 'vec'

The function `vec` plots vectors of a 2D flow field (Example 2.7, Figure 6).

Example 2.5.

```
# Load bjornholt data set
data(bjornholt)
# plot the time series
plot(bjornholt)
# map the location
map(bjornholt)
```

a) b)

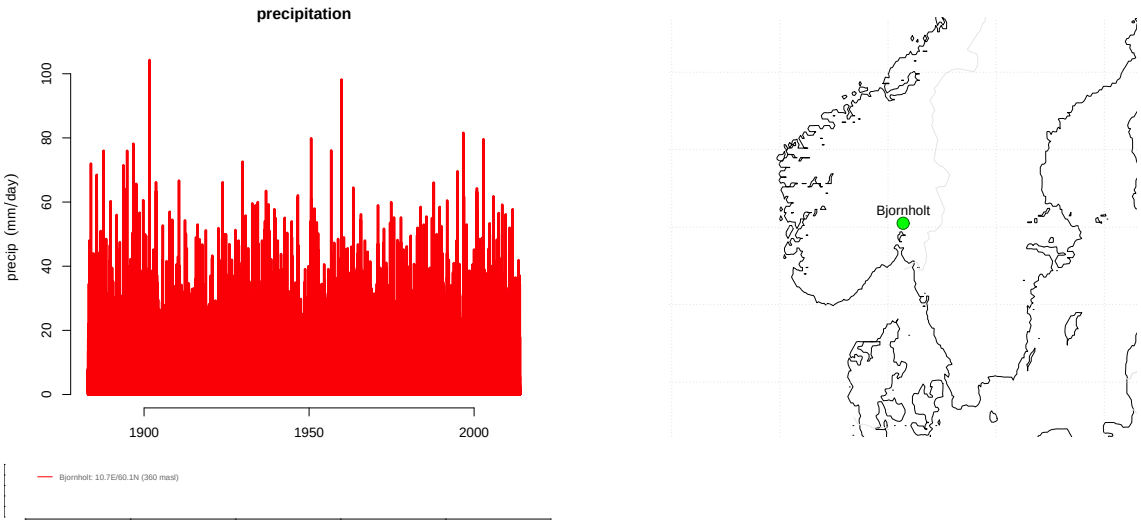


Figure 4: Example of a station (a) and a map plot (b) for a single station 'Bjornholt'.

Example 2.6.

```
  #Get NCEP 2m air temperature for the selected spatial window defined by lon and lat
t2m <- t2m.NCEP(lon=c(-30,30),lat=c(40,70))
  # Computes the EOFs
  X <- EOF(t2m)
  # Plot the result
  plot(X)
```

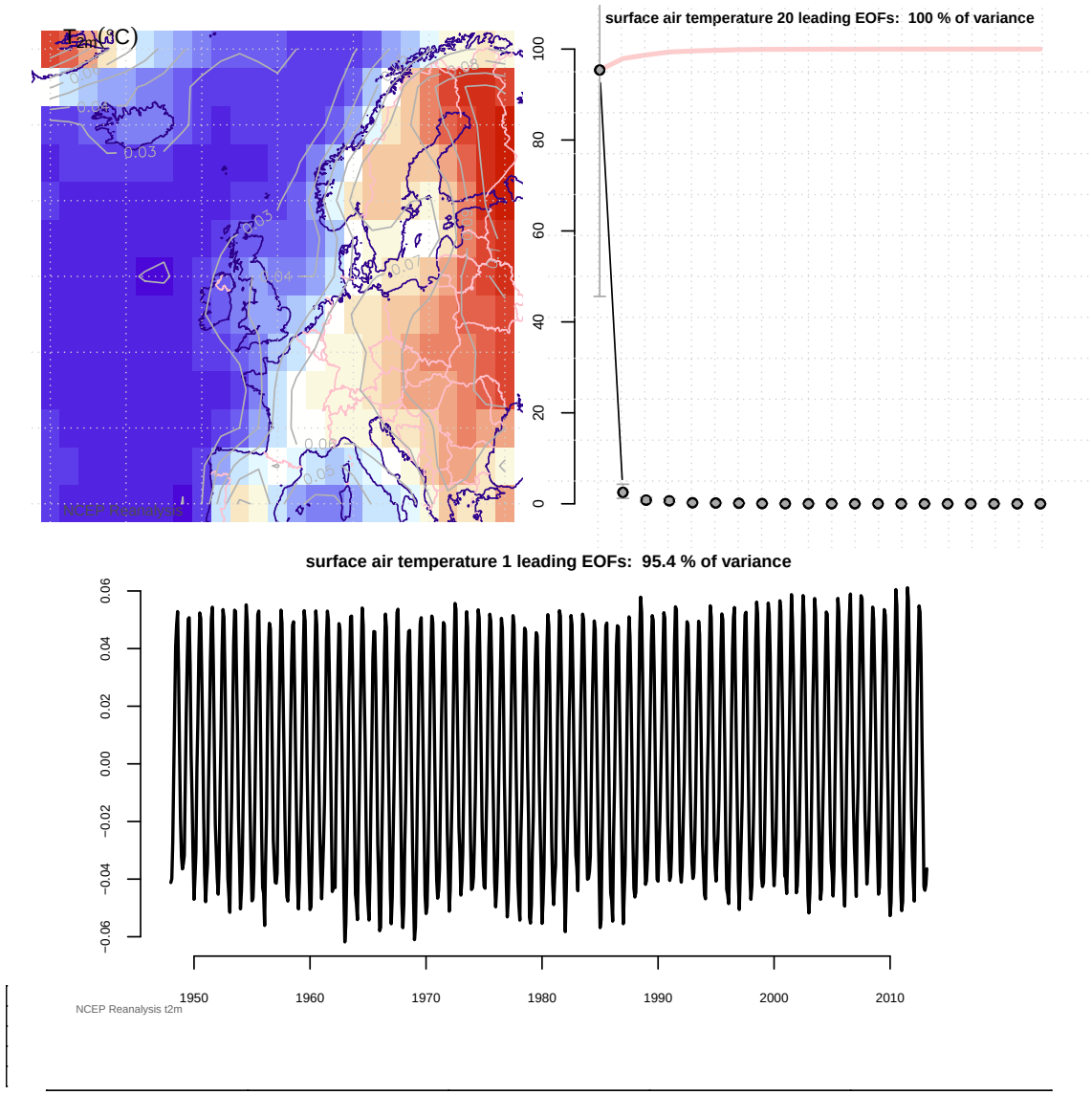


Figure 5: An example of a plot results for an EOF object.

Example 2.7.

```
# Load 10m zonal and meridional wind components
u10 <- retrieve('data/ERAINT/eraint_elnino.nc',param='u10')
v10 <- retrieve('data/ERAINT/eraint_elnino.nc',param='v10')
# Map the data
map(u10,colorbar=FALSE)
# Display the vectors
vec(u10,v10,new=FALSE,a=2,length=0.05)
```

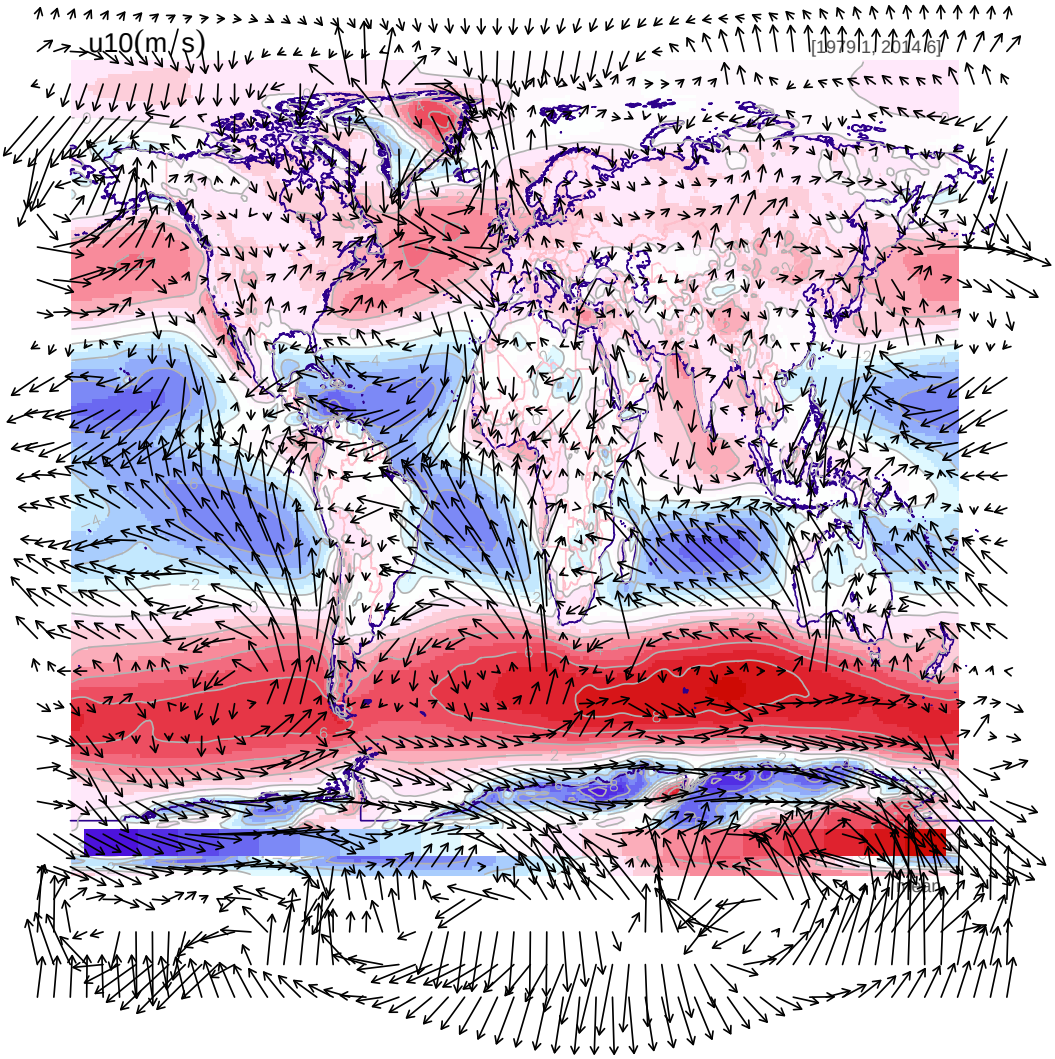


Figure 6: The output of the example with `'vec'`. In this example, there are different plotting windows for the vectors and the underlying map, but this can be adjusted in the arguments of `'vec'`.

2.4.4 Info-graphics

One of the purposes of `'esd'` is to easily produce visualisation and graphics to bring out new aspects of the information embedded in the data. The development of the info-graphics in this tool has also been inspired by *Spiegelhalter et al.* (2011), in order to distill the essence of the analysis.

The information stored in climate data can be extracted in various ways, with emphasis on different aspects, as can be seen in Example 2.8 and Figure 7. A `'cumugram'` is shown, displaying the cumulative average value of some variable starting from the first day of the year. The results for the Oslo temperature in this example shows that 2014 has been the warmest year on record since the summer. A 'rainbow structure' is consistent with a gradual increase in the temperature. The second diagram, `'wheel'`, emphasises the time of the year when the most extreme events have taken place, and `'climvar'` to the right shows how the year-to-year variance varies with season with a minimum in late summer. The `'diagram'` method can also be used to view the data by comparing day-by-day values of the present temperature with those of the previous years. The figure shows that there have been some exceptionally mild autumn temperatures in 2014. Other functions for making info-graphics include `'vis'`, which make alternative graphics output displaying different information.

Trends can be estimated by linear regression using the simple `'trend'` function. An alternative trend analysis can be done using the function `'vis.trends'` which estimates linear regressions for sliding periods of various lengths (Example 2.9, Figure 8). The results are presented visually with the strength of the trends shown as a colour scale on a grid where the x- and y-axes represent the starting point and the length of each period, respectively. Periods with statistically significant trends are marked with black outlines. The advantage of `'vis.trends'` is that it shows trends of various time scales, considering all variations of start- and end-points. The longest period is found in the upper left corner, representing the full length of the time series. The most recent period is shown in the bottom right corner. As demonstrated in Example 2.9, the strength and significance of estimated trends are sensitive to the period considered. The multiple period trend analysis is therefore a more robust alternative to single period trend fitting.

Example 2.8.

```
# Get 2m temperature data for Oslo
# (works within MET Norway firewall only)
x  <- station(stid=18700,param='t2m',src='metnod')
x <- subset(x,it=c(1837,2014))
## Or gets data from ECA&D as
## x  <- station(loc="Oslo Blindern",param='t2m',src='ecad')
## x <- subset(x,is=duplicated(loc(x))) # to remove duplicated stations
# Cumulative average
cumugram(x)
# Seasonal wheel
wheel(x)
# seasonal variations of year-to-year variance
climvar(x)
# daily seasonal cycle for all years
diagram(x)
```

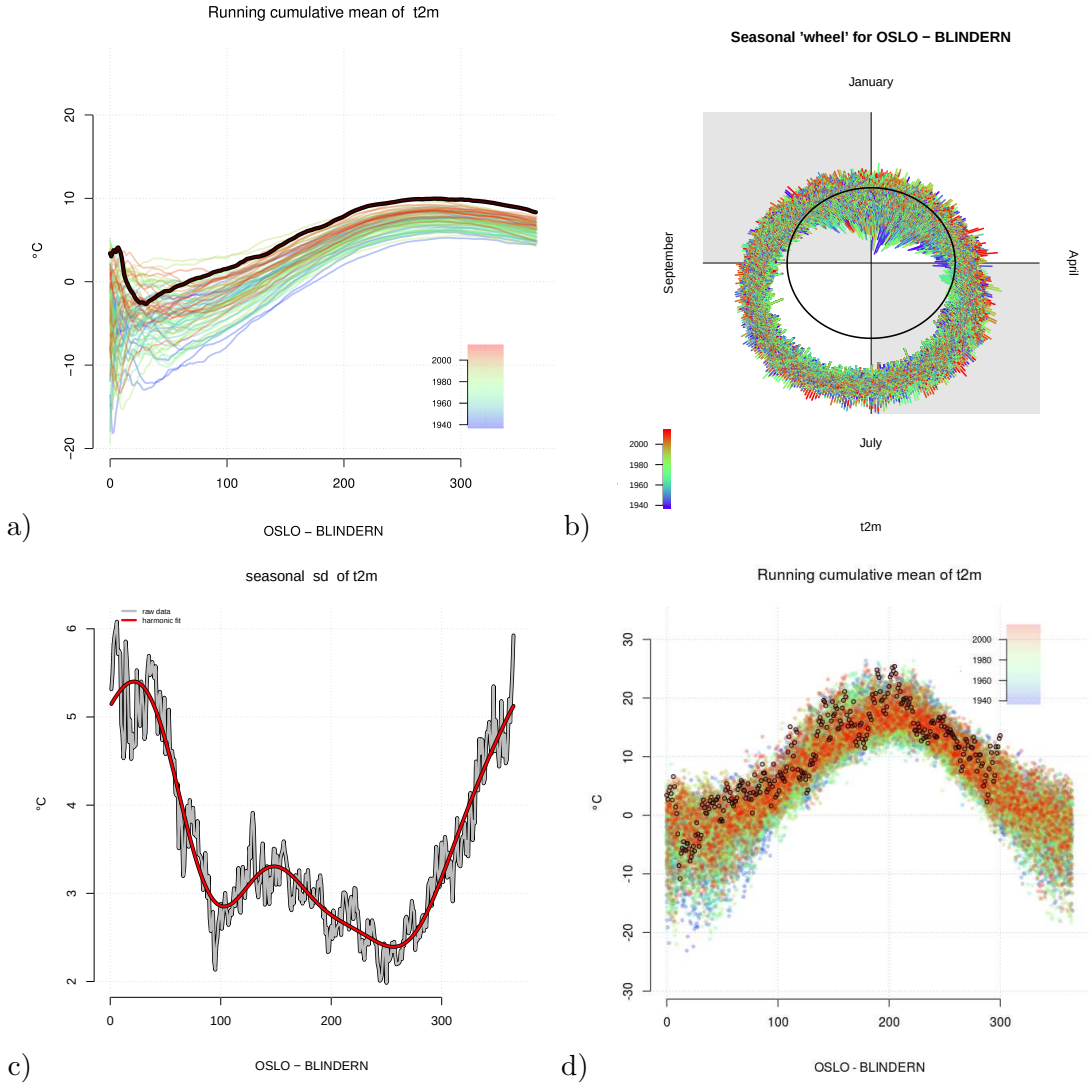


Figure 7: Examples of `'cumugram'`, `'wheel'`, `'climvar'`, and `'diagram'` plots. The `'cumugram'` shows how the mean value of some variable has evolved from the start of the year and compares this curve to previous years. The graphics produced by `'wheel'`, on the other hand, emphasises how the seasonal variations affect the variable, e.g. whether some extremes tend to be associated with a specific season. Panel c shows results produced by `'climvar'` shows the year-to-year statistics for a variable, e.g. the standard deviation of the temperature on February 1^{st}. The `'diagram'` method can be used in different context, and for a `'station'` object, it produces graphics that compare the day-to-day values with those of previous years.

Example 2.9.

```
# Get 2m temperature data for Ferder and calculate annual mean
data(ferder)
x <- annual(ferder)
# Plot the time series and add a trend line
plot(x,ylim=c(4,11))
lines(trend(x),col="red",lwd=2)
# Visualise trends for various periods 40 years or longer (minlen)
# and mark trends that are significant at the 1% level (pmax)
vis.trends(x,minlen=40,pmax=0.01)
```

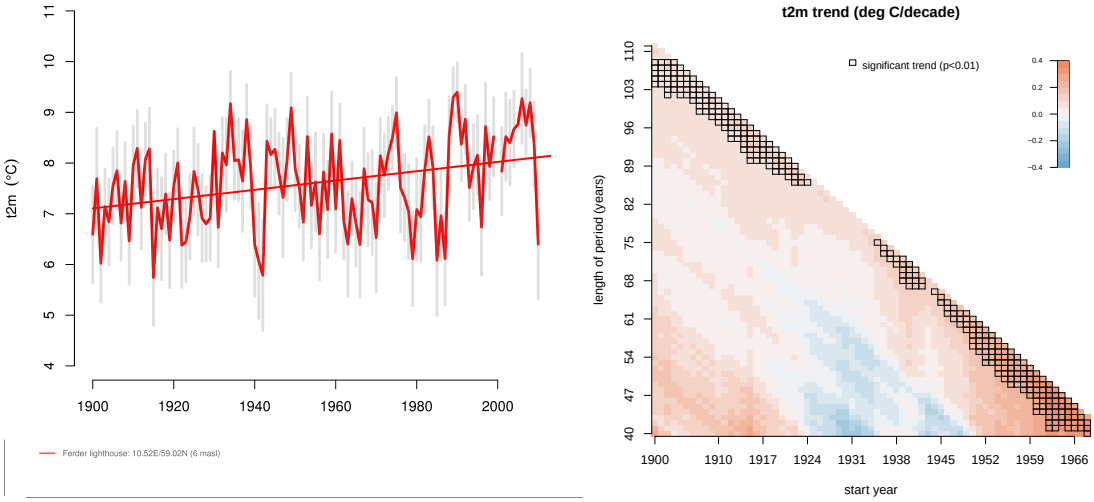


Figure 8: Visualising the trend in the temperature time series from Ferder for the full period (left panel) and simultaneously for different periods of various length (right panel). The trend analysis indicates a significant increase in temperature at Ferder since the 1900s, but also shows that the strength and significance of the trend is sensitive to the period considered because the increase is not monotone.

3 Data handling & processing

The 'esd'-package contains a set of different methods for handling and processing data and model results. All the processes, however, are designed to leave a 'stamp' that shows how the objects have been created and is saved in the 'history' attribute of an object. The history can be extracted using the function 'history()'. This is very useful as it makes the analysis more transparent, traceable, and reproducible. One important objective of 'esd' is also to make data handling simple and intuitive, as well as efficient.

All 'esd' functionalities will use the same type of arguments and logic.

For instance, the argument 'it' is used to pass a time index to the call, whereas 'is' is used as a space index - these are for instance used to extract sub-samples from the data, e.g. when one wants to work on a subset (in time and/or space) of the original data (e.g. a smaller region, a shorter interval, or a specific season/month; see below).

3.1 Formulas & functions

3.1.1 Small handy functions

The two functions 'g2dl' (Greenwich to dateline) and 'sp2np' (south-pole to north-pole) were included in 'esd' to organise and sort data in a consistent way. Many gridded data come with different choices: some are ordered from north to south, others from south to north; some start from Greenwich (0 degrees east) whereas others start from the date line (180 degrees west).

Other functions to simplify data processing include 'lon' (longitude), 'lat' (latitude), 'alt' (altitude), 'cntr' (country),'loc' (location name), 'varid' (variable name), and 'stid' (station number). Some of these only apply to 'station' objects.

3.1.2 Wet-means, frequencies, counts and spells

A number of functions have been introduced to make analysis of precipitation simpler, such as the estimation of wet-day mean (a threshold is set to 1mm/day as default) or wet-day frequency. These functions include 'wetmean', 'wetfreq', 'count', 'nv', and 'spell', and can be used in association with 'aggregate'. Some of these functions are wrappers for the function 'exceedance' which discards all data with a value lower than a given threshold value. Table 2 gives an overview and some examples are provided in Example 3.1 and 3.2.

Table 2: A list of specialised functions in `'esd'` designed to make climate analysis simple and user-friendly.

`'wetmean'`	Estimate the wet-day mean μ. Default threshold is 1mm/day.
`'wetfreq'`	Estimate the wet-day frequency f_w. Default threshold is 1mm/day.
`'exceedance'`	Select the data with values exceeding a critical threshold value.
`'count'`	Count data points in sample.
`'C.C.eq'`	Clausius-Clapeyron equation.
`'NE'`	Predict number of events, given a frequency and a sample size.
`'spell'`	Estimate the spell lengths (consecutive days/straights) of events.
`'precip.vul'`	Simple vulnerability index associated with precipitation: $V_p = \mu/f_w$.
`'t2m.vul'`	Simple vulnerability index associated with temperature based on consecutive number of hot days above a critical threshold (default 30 degrees C): $V_T = \overline{n_{chd}}$.
`'precip.rv'`	Simple and rough estimate of return value for weak-to-moderate 'extremes': $x_\tau = -\ln(1/(f_w\tau))\mu$.
`'nv'`	Number of valid data points in sample.
`'precip.Pr'`	Simple and crude estimate of the probability of precipitation exceeding a threshold (default: 10mm/day): $Pr(X > x) = f_w exp{-x/\mu}$ assumes exponential distribution.
`'t2m.Pr'`	Simple and crude estimate of the probability of temperature exceeding a threshold (default: 30 degree C): $Pr(X > x) = N(\mu, \sigma)$ assumes normal distribution.

Example 3.1.

```
# Load data for Bjornholt
data(bjornholt)
y <- bjornholt
# Annual wet-mean:
mu <- annual(y,FUN='wetmean')
# Plot the result for the period 1883 to 2014
plot(mu,ylim=c(4,14))
# Annual wet-freq:
fw <- annual(y,FUN='wetfreq')
# Plot the result for the period 1883 to 2014
plot(subset(fw,it=c(1883,2014)),ylim=c(0.2,0.5))
# Annual number of events with y > 10mm/day:
nw <- annual(y,FUN='count',threshold=10)
# Plot the result for the period 1883 to 2014
plot(subset(nw,it=c(1883,2014)),ylim=c(10,60))
# Compute wet/dry spells
sp <- spell(y,threshold=1)
# Plot the result
plot(sp)
```

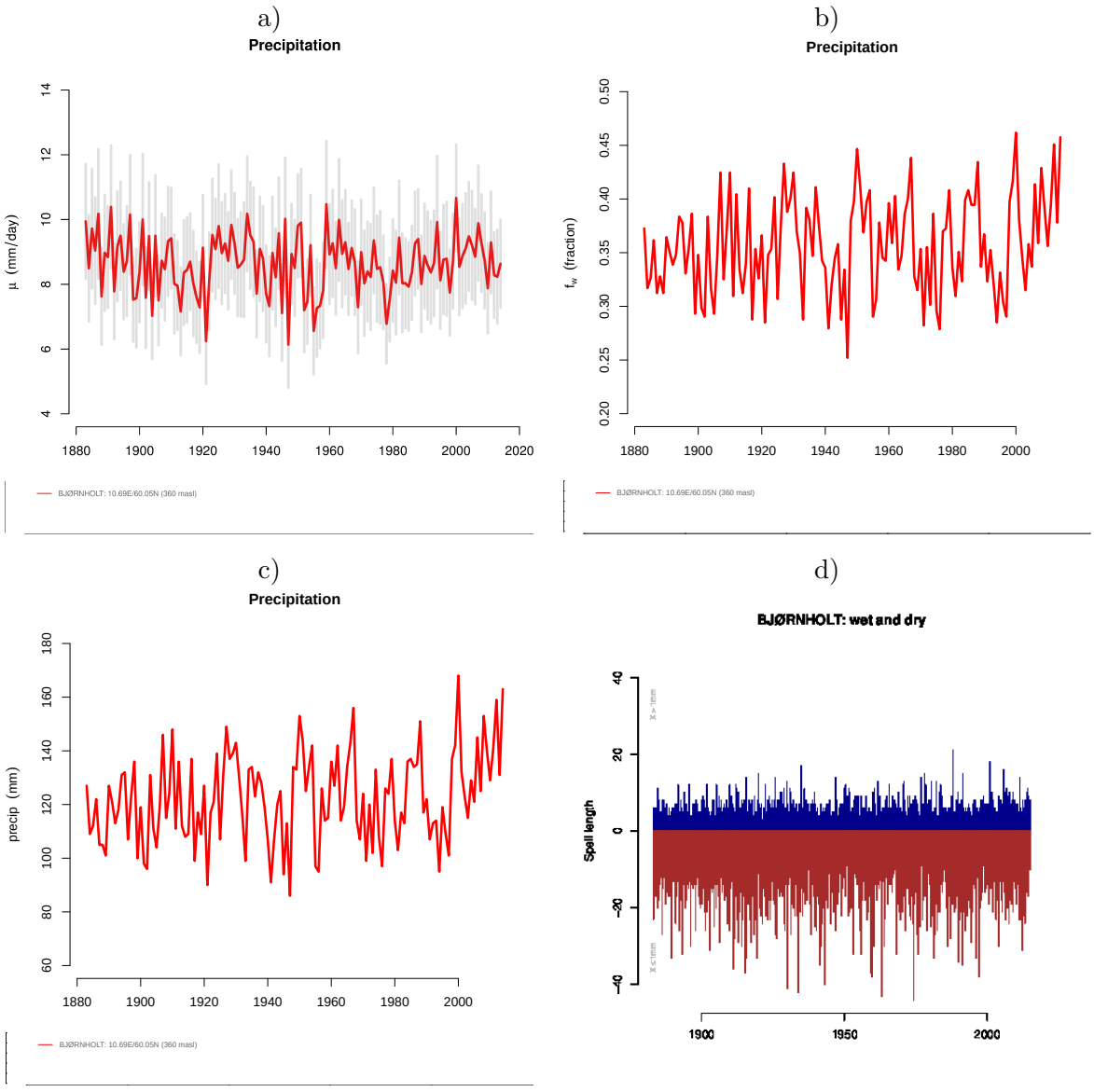


Figure 9: Plots of precipitation statistics at Bjornholt such as a) wet-day mean (μ), b) wet-day frequency (f_w), c) number of events with precipitation higher than 10mm, and d) wet/dry spells.

Example 3.2.

```
# Load ECAD dataset over Norway recording a 100 years of daily data
ecad <- station(param="precip",src="ecad",cntr="Norway",nmin=100)

# Map the vulnerability index:
map(ecad,FUN='precip.vul',cex=1.2,xlim=c(-10,40), col="black",colbar=list(col=heat.colors(20)))

# Map approximate 10-year return values:
map(ecad,FUN='precip.rv',cex=1.2,xlim=c(-10,40), col="black",colbar=list(col=heat.colors(20)))
```

a) b)

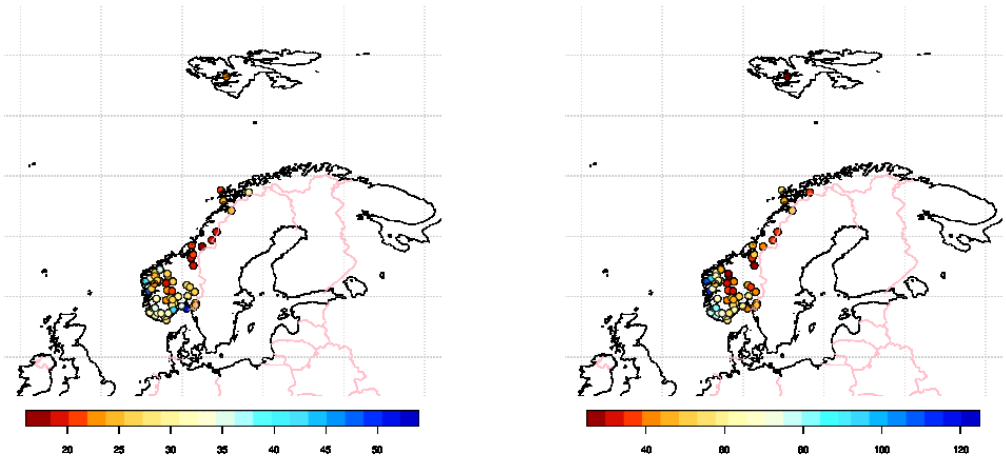


Figure 10: An example of a plot of the vulnerability index μ/f_w (left) and an approximation of daily 10-year return values (right) for a selection of station from the ECA&D recording at least a 100 years of data.

3.2 re-gridding

Gridded data often come on different spatial grids and/or resolutions, such as re-analyses, global climate models (GCMs) and regional climate models (RCMs). In order to compare these or apply a common set of analyses to the different model results or reanalysis, it is necessary to present the data on a common grid resolution first. This is achieved via the function called `'regrid'` which performs a bilinear interpolation from the original grid resolution to a new one. `'regrid'` is also an S3-built in function and can be applied on all `'esd'` objects. Another strength of `'regrid'` is that it allows also interpolating from gridded data or GCM/RCM model results into a specific location or station. Regrid can take several minutes and is time consuming depending on the selected region and grid resolution.

Again, the common syntax for the argument `'is'` is shown, where `'is'` refers to spatial indexing whereas `'it'` is for temporal indexing. The methods have been designed with some flexibility in mind, so that when `'is'` is assigned another field type, it will interpolate the data onto that specific grid. Similarly, if `'is'` is given a station object, then it will interpolate the data to the same coordinate as the station (the output will then be a station object). The general syntax of `'regrid'` is as follows

```
regrid(x,is,...)
```

There is a difference between `'is'` and `'it'` because the spatial indexing can include both longitude and latitude dimensions, as well as a number of stations. Hence, the `'is'` is often given a `'list'` object (Example 3.3).

Example 3.4 shows the re-gridding between the NCEP reanalysis and NorESM.M global climate model result for an area covering Europe (30W–30E/40N–70N). In this particular case, both the gridded products (NCEP and NorESM.M) have the same spatial resolution of 2.5 degrees, but `'regrid'` works also with different grid resolutions.

3.2.1 How the re-gridding works

The re-gridding is based on a bi-linear interpolation according to the linear algebra expression

$$X' = WX \tag{1}$$

W is a sparse matrix of weights since each new grid cell is a weighted sum of the four surrounding grid cells. The sparse character saves computer resources compared to the full weight matrix which would have the dimensions of the product of X' and X. Thus, the re-gridding becomes more efficient by (a) utilising the information about the sparseness (i.e. only needs the weights and the index of the surrounding grid cells to the point of interest) and (b) computing the weights only once, then use using the `'apply'` function rather than for loops to weight all time steps.

3.3 Nearest data point

An alternative to re-gridding is to select the nearest data point, as a bi-linear interpolation will have an influence on extremes through the estimation of values in terms of weighted sums of nearby points. The 'esd' package provides the function 'nearest' that selects a subset of the nearest point as

```
obs <- nearest(obs,is="any rcm or gcm object")
```

3.4 Subsetting

It is sometimes necessary to limit the range of data, either by selecting an interval in time or a subregion of the data. The method 'subset' in the 'esd' tool extends the R-base 'subset' method to 'esd' objects and classes and can be used to extract a time interval or a specific date, month, season, sub-group of stations, or a smaller region of a field. Its use is meant to be versatile and is based on the two arguments 'it' and 'is' to make a selection possible as

```
subset(x,it=NULL,is=NULL,...)
```

The subsetting works for both fields and group of stations. In the examples below, Y can be a (group of) station(s) or a field object. In Example 3.5, the argument 'it' is used to extract a specific time interval of the input data Y. It is also possible to select a subset according to a second data object as

```
# Select the same time period as covered by data object X
y <- subset(Y,it=X))
# Select the same spatial region/group of stations as in X
y <- subset(Y,is=X))
```

or to combine several selection criteria as in Example 3.6.

It is usually wise to use 'subset' before other data processing (e.g. aggregate) to reduce computation time.

Example 3.3.

```
# Load NCEP 2m air temperature
t2m <- t2m.NCEP(lon=c(-30,30),lat=c(40,70))
# map the original field
map(t2m)
# Regrid based on the new lon and lat values
y <- regrid(t2m,is=list(lon=seq(-5,15,by=0.5),lat=seq(55,65,by=0.5)))
# Map on the new grid
map(y)
```

Example 3.4.

```
# Get NCEP data
> ncep <- t2m.NCEP(lon=c(-15,45),lat=c(35,70))
# Display the latitude values
> lat(ncep)
[1] 35.0 37.5 40.0 42.5 45.0 47.5 50.0 52.5 55.0 57.5 60.0 62.5 65.0 67.5 70.0
# Compute the grid resolution
> diff(lat(ncep))
[1] 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5
# Display the longitude values
> lon(ncep)
[1] -15.0 -12.5 -10.0  -7.5  -5.0  -2.5   0.0   2.5   5.0   7.5  10.0  12.5
[13]  15.0  17.5  20.0  22.5  25.0  27.5  30.0  32.5  35.0  37.5  40.0  42.5
[25]  45.0
# Compute the resolution in the lon axis
> diff(lon(ncep))
[1] 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5
[20] 2.5 2.5 2.5 2.5 2.5
# Get GCM data
> gcm <- t2m.NorESM.M(lon=c(-15,45),lat=c(35,70))
> lat(gcm)
[1] 36.25 38.75 41.25 43.75 46.25 48.75 51.25 53.75 56.25 58.75 61.25 63.75 66.25 68.75
> diff(lat(gcm))
[1] 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5
> lon(gcm)
[1] -13.75 -11.25  -8.75  -6.25  -3.75  -1.25   1.25   3.75   6.25   8.75
[11]  11.25  13.75  16.25  18.75  21.25  23.75  26.25  28.75  31.25  33.75
[21]  36.25  38.75  41.25  43.75
> diff(lon(gcm))
[1] 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5
[20] 2.5 2.5 2.5 2.5
# Do the re-gridding # this line does not work
> gcm.regrid <- regrid(gcm, is=ncep)
# Make sure that both longitudes are set to data line (i.e. greenwich=FALSE)
> ncep <- g2dl(ncep,greenwich=FALSE)
> gcm <- g2dl(gcm,greenwich=FALSE)
# Do the re-gridding
> gcm.regrid <- regrid(gcm, is=ncep)
> lat(gcm.regrid)
[1] 35.0 37.5 40.0 42.5 45.0 47.5 50.0 52.5 55.0 57.5 60.0 62.5 65.0 67.5 70.0
> lon(gcm.regrid)
[1] -15.0 -12.5 -10.0  -7.5  -5.0  -2.5   0.0   2.5   5.0   7.5  10.0  12.5
[13]  15.0  17.5  20.0  22.5  25.0  27.5  30.0  32.5  35.0  37.5  40.0  42.5
[25]  45.0
```

Example 3.5.

```
data(Oslo)
# Extract an interval:
y <- subset(Oslo,it=as.Date(c("1883-01-01","2013-12-05")))
# Extract only the winter data (use aggregate for winter statistics)
djf <- subset(y,it='djf')
# Extract data for May:
may <- subset(y,it='May')
```

Example 3.6.

```
# Retrieve stations across Scandinavian regions from the
# ECA$\&$D dataset with a minimum of 50 years of data
y <- station(src="ecad",cntr="norway",nmin=50)
# Show the selected stations including all available stations
map(y,cex=1.4,col="red",bg="pink",showall=TRUE)
# Subset stations with altitude higher than 100m
y1 <- subset(y,is=list(alt=100))
# Show the stations with elevation greater than 100m above sea level:
map(y1,cex=1.4,col="darkred",bg="red",showall=TRUE)
# Show the stations with elevation below 100m above sea level:
y2 <- subset(y,is=list(alt=-100))
map(y2,cex=1.4,col="darkred",bg="orange",showall=TRUE)
```

a) b) c)

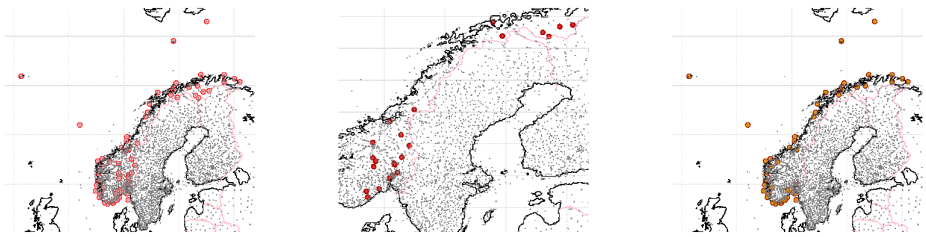


Figure 11: Maps of available weather stations from the ECA&D with a minimum of 50 year recorded values including a sub-selection of stations showing higher (b) and lower (c) elevation than 100m a.s.l.

3.5 Combining and synchronising

It is important to combine different data objects and make sure that they are synchronised before the analysis can be made to identify links and dependencies. It may also be necessary to combine several station objects into a group of objects, for instance in order to perform a canonical correlation analysis (CCA)CCA. The method is `'combine'`.

The function `'matchdate'` is used to synchronise any object `'x'` with the time index specified by `'it'` as

```
matchdate(x,it=y)
```

`'matchdate'` will return the subset of `'x'` that matches the time index of `'y'`.

3.6 Anomalies

Geophysical data tends to have a strong seasonal cycle, and often the seasonal variations are of less interest than year-to-year variations. Anomalies are the variations after the seasonal cycle has been removed. The method `'anomaly'` can be applied to stations and field objects, for daily, monthly, and seasonal data. The converse is `'climatology'`.

3.7 Aggregate: monthly, seasonal and annual statistics

It is also convenient to compute trends on annual values rather than daily values, as the latter are noisy. To do so the S3-built in method `'aggregate'` is used which allows splitting the data into subsets (e.g. years) and computes summary statistics for each subset (e.g. maximum values) as

```
aggregate(x,by=year,FUN='max',...)
```

Aggregate has been extended to deal with `'esd'` objects, and returns the result as an identical `'esd'` object with updates. Classes and attributes are updated accordingly (see Example 3.7). The aggregate function can also be used to create coarser grids where the boxes contain spatially aggregated values from a higher grid resolution (Example 3.8).

3.8 Spatial averaging of field objects - aggregate.area

It is interesting to investigate if there are any global signals affecting the local climate. For this purpose, the `'esd'` package makes it convenient to compute statistics on spatial averaging of an object over a specific spatial domain or an area of interest. The function `'aggregate.area'` is used to compute an area aggregate (e.g. average means, maximum, sum, ...) taking into account that the grid box area varies with latitude. The following equations are used:

$$\bar{x} = \frac{1}{I} \sum_{i=1}^{I} \left(\sum_{j=1}^{J} (\varphi_i \, x_{i,j}) / \sum \varphi \right)$$

$$\varphi_i = \cos(2\,\pi \; lat_i \,/\, 360)$$

where i and j are indices in the longitude and latitude respectively, and lat_i is the latitude value at point i.

```
aggregate.area(x,is=NULL,it=NULL,FUN='sum',na.rm=TRUE,smallx=FALSE)
```

Example 3.9 shows the spatial averaging of the projected global mean temperature from the CMIP5 NorESM-M RCP4.5 experiment.

It is also convenient to compare the global mean temperature as produced by several GCMs (Figure 12). A demo script is made available in the demo `'esd'` package called "`global_tas_anomaly.R`". The script computes the global mean anomaly temperature for all CMIP3 and CMIP5 experiments provided by the KNMI Climate-Explorer web portal. All GCM data need to be downloaded locally before the script is run. The results can then be compared to the Figure 1 of *Knutti and Sedláček* (2013) which shows the evolution of the global temperature change (mean and one standard deviation as shading) relative to 1986–2005 for the SRES scenarios run by the CMIP3 and the RCP scenarios run by the CMIP5 experiments. This figure gives a good summary of the global mean warming signal predicted by both experiments and the inter-model

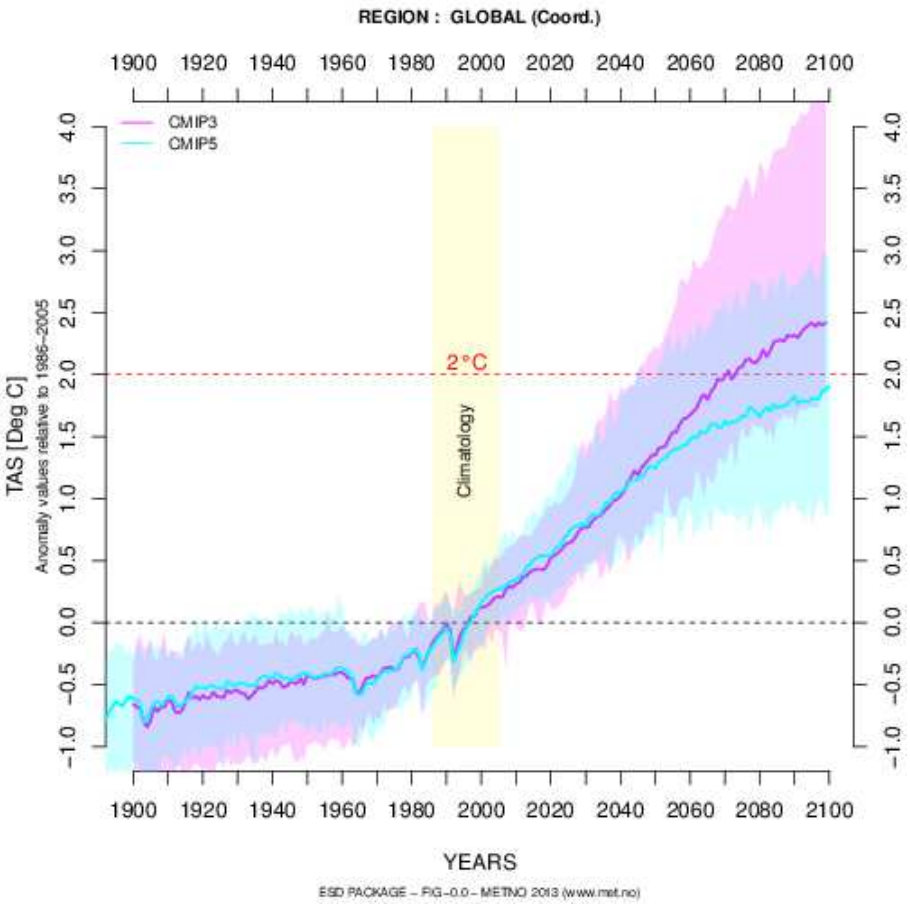


Figure 12: Global mean temperature change for the SRES scenarios run by CMIP3 and the RCP scenarios run by CMIP5 experiments, respectively, relative to the period 1986-2005. The shaded area shows one standard deviation from the mean based on all scenarios for each experiment.

spread. Note that the shaded area would be different if it was based on the ensemble model outputs for each CMIP experiment as the authors gave a confidence interval based on one standard deviation of the ensemble mean.

Example 3.7.

```
# Load data for "Bjornholt" station
data(bjornholt)
# Check the class of the object
class(bjornholt)
## [1] "station" "day"     "zoo"
# Aggregate on annual maximum values
bjornholt <- annual(bjornholt,FUN='max')
# Check the class of the aggregated object
class(bjornholt)
## [1] "station" "annual"  "zoo"
# Plot the results
plot(bjornholt)
```

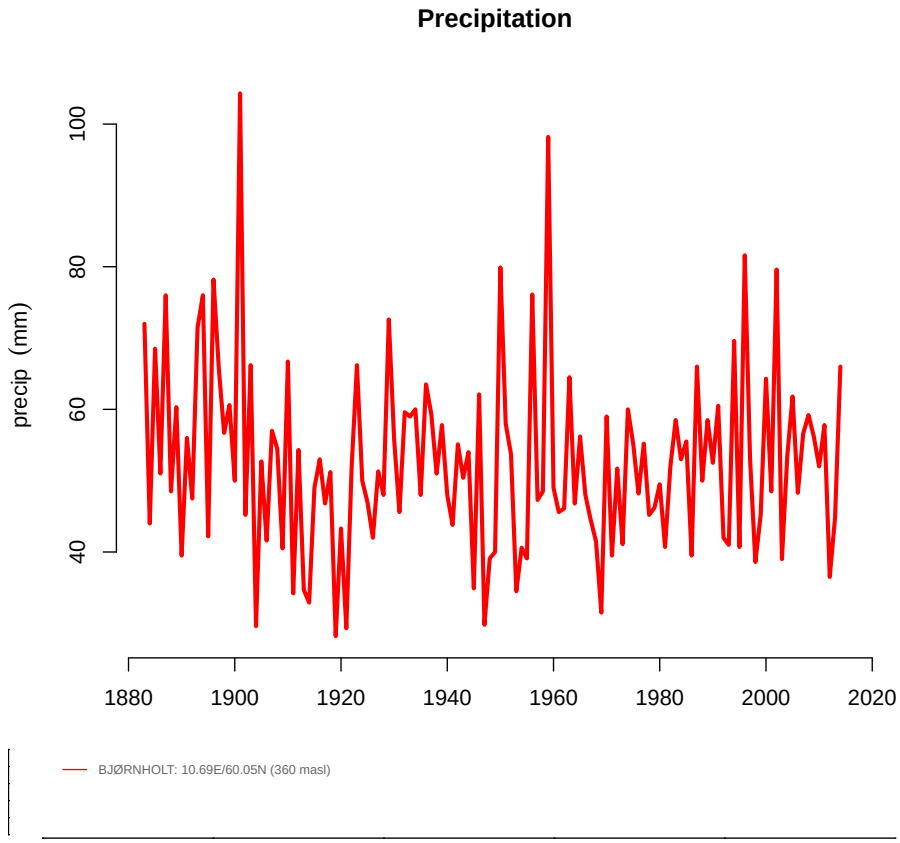


Figure 13: Plotting the annual maximum values of daily precipitation recorded at 'Bjornholt' weather station.

Example 3.8.

```
# Load 2m air temperature from NCEP reanalysis on a resoltuion of 2.5 deg.
t2m <- t2m.NCEP()
# Do the spatial aggregation
x <- aggregate(t2m,by=list(lon=seq(0,360,by=10),lat=seq(0,360,by=10)),FUN='mean')
# Map the results
map(x)
```

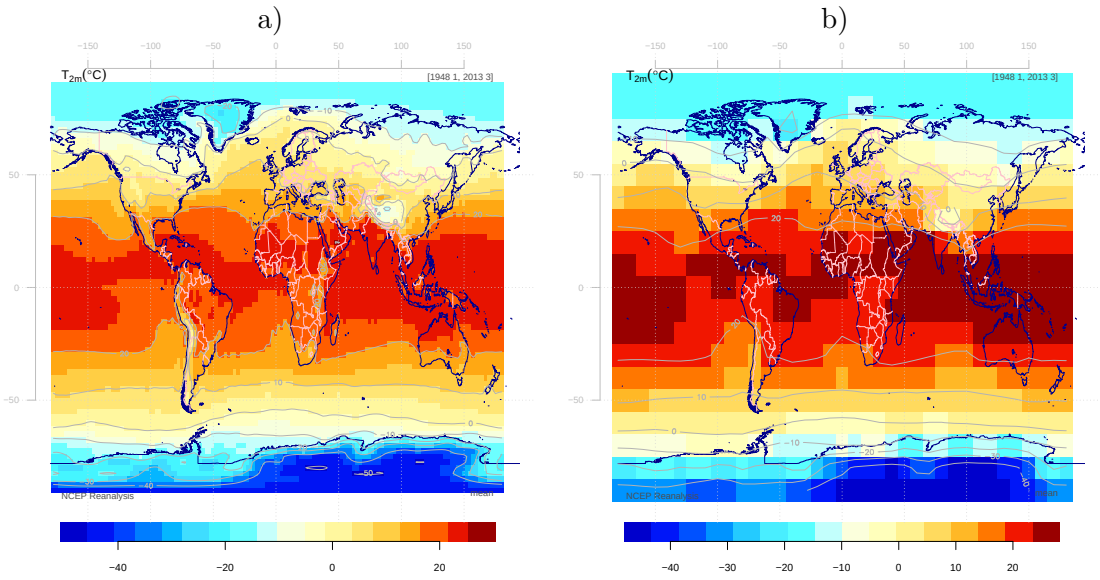


Figure 14: Maps of the original (a) and aggregated (b) spatial field of NCEP 2m air temperature.

Example 3.9.

```
# Load 2m air temperature from the NorESM.M global climate model
t2m <- t2m.NorESM.M()
# Compute the areal mean over the whole domain
T2m <- aggregate.area(t2m,FUN='mean')
# Plot the annual aggregated values
plot(annual(T2m),ylim=c(11.5,17.5))
```

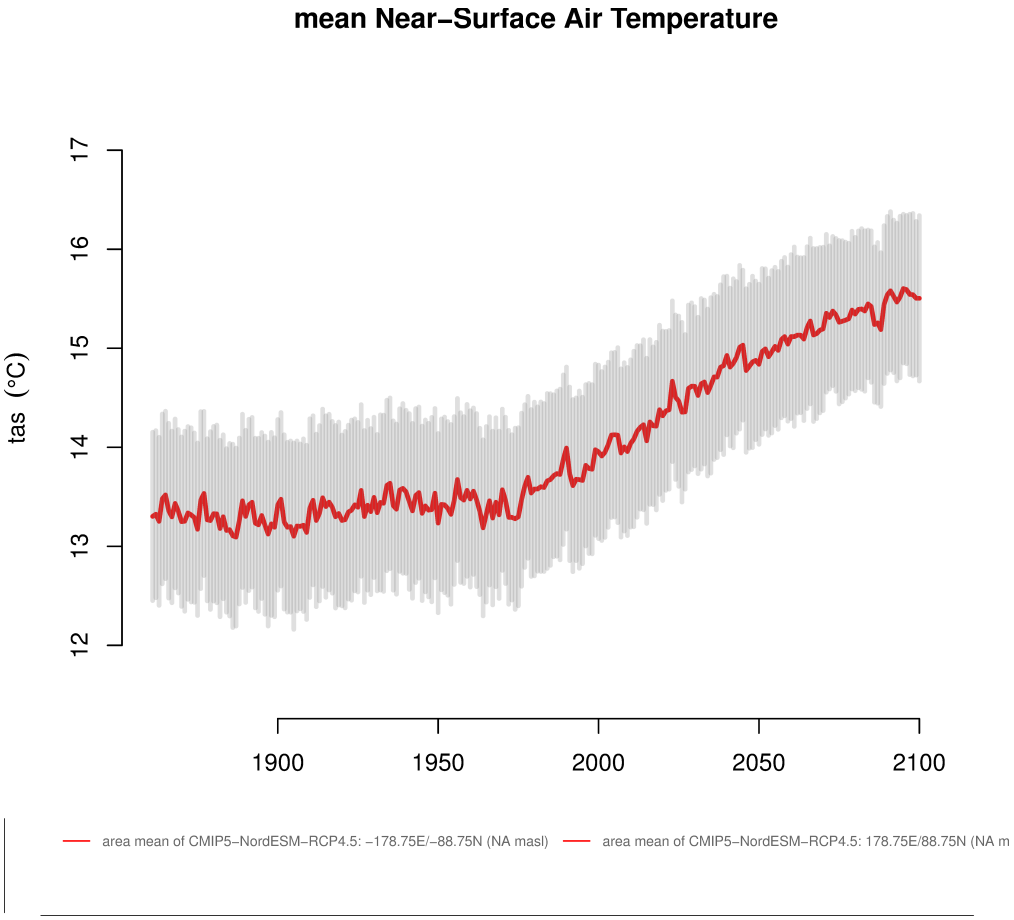


Figure 15: Global average of 2m air temperature from the NorESM global climate model. The error bars show 2 times the standard deviation computed based on observations.

3.9 Transformations and conversions: as.

A range of transformations between different type of objects can be done with the 'as' method.

Table 3: The ‘esd’ extension of the ‘as’ method. Some of these are the same as some other functions, e.g. ‘annual’ and ‘as.annual’

as.4seasons	as.field.station	as.4seasons.day	as.field.station
as.4seasons.default	as.field.zoo	as.4seasons.field	as.fitted.values
as.4seasons.spell	as.fitted.values.ds	as.4seasons.station	as.monthly
as.annual	as.original	as.annual.default	as.original.data
as.annual.integer	as.original.data.ds	as.annual.numeric	as.original.data.station
as.annual.spell	as.original.station	as.annual.yearqtr	as.pattern
as.anomaly	as.pattern.cca	as.anomaly.default	as.pattern.corfield
as.anomaly.field	as.pattern.ds	as.anomaly.station	as.pattern.eof
as.anomaly.zoo	as.pattern.field	as.appended	as.pattern.mvr
as.appended.ds.comb	as.pca	as.appended.eof.comb	as.pca.ds
as.appended.field.comb	as.pca.station	as.calibrationdata	aspect
as.calibrationdata.ds	as.residual	as.calibrationdata.station	as.residual.ds
as.climatology	as.residual.station	as.comb	as.seasons
as.comb.eof	as.stand	as.ds	as.stand.station
as.eof	as.station	as.eof.appendix	as.station.data.frame
as.eof.comb	as.station.ds	as.eof.eof	as.station.eof
as.eof.field	as.station.field	as.eof.zoo	as.station.list
as.field	as.station.pca	as.field.comb	as.station.spell
as.field.default	as.station.zoo	as.field.eof	as.eof.list

4 Analysis & Diagnostics

4.1 Empirical Orthogonal Functions

Empirical orthogonal functions (EOFs; *Lorenz* (1956)) provide a handy framework for multivariate data analysis. Here EOFs refer to a class of data objects, however, in a more general context, EOFs refer to the spatial coherent structures which maximise the variance, whereas the principal components (PCs) refer to time series describing the degree of their presence at any time. The eigenvalues refer to the variance of each EOF mode.

In `'esd'`, the EOFs are estimated using a singular value decomposition (SVD) (*Press et al.*, 1989a; *Strang*, 1988):

$$X = U\Lambda V^T, \tag{2}$$

where X is a matrix of data with two dimensions (space, time), U hold the EOF patterns, Λ is a diagonal matrix with the eigenvalues, and V contains the PCs. The EOFs are used to extract the essence of the information embedded in the data, taking advantage of redundancy and emphasising the most prominent characteristics. Example 4.1 shows how the EOFs can be estimated and visualised in `'esd'`.

The EOFs are used as input to other analysis, such as downscaling (`'DS'`) and canonical correlation analysis (`'CCA'`). It is also possible to recover the original data from EOFs through `'eof2field'`, however, the number of EOFs are usually truncated, and only the most prominent features are recovered. The function `'eof2field'` can be used to filter the data in terms of removing small scale and noisy features.

The EOFs can provide an indication of some of the most prominent phenomena in the climate system, such as the annual cycle, the El Niño Southern Oscillation, and the Arctic Oscillation.

4.2 Principal Component Analysis

The method called `PCA` - principal component analysis - is similar to EOF, but is designed for groups of stations rather than gridded fields (Example 4.2). The PCA can also be used to represent data on an irregular grid (such as rotated fields from regional climate models). It is possible to grid the spatial modes of the PCA onto a regular grid, and hence convert the `pca` class into a `eof` class (the gridding is currently not performed in `esd`, but could be done using optimal interpolation, or kriging taking geographical features into account (*Benestad et al.*, 2012)). Whereas EOF weights each grid box with its grid box area, PCA does not apply any weighting to the different series (which may imply that correlated variability from nearby stations is emphasised by the PCA).

PCAs are useful for investigating large-scale dependency, as the leading mode will pick up patterns with coherent variations across the stations. They are also used in CCA and identifying stations with suspect data.

4.3 Canonical Correlation Analysis

Canonical correlation analysis (CCA) can be used to explore dependencies between different data sets (Example 4.3). It is a useful tool for investigating suitable predictors for downscaling or identifying tele-connections. Sometimes it can provide some indications of suspect station data.

The computation of the CCA in `'esd'` is based on the method by Barnett-Preisendorfer (*Barnett and Preisendorfer*, 1987; *Wilks*, 1995). The inputs are either an `'pca'` or `'eof'` class.

4.4 Other types of analysis

Methods such as singular spectrum analysis (`'SSA'`) and `'coherence'` have been adapted from the `'clim.pact'` package, but have not been elaborated and tested yet for the `'esd'` objects. There is also a set of low-pass filters such as `'filt'`. Other type of analysis can be included such as performing a multivariate regression analysis (MVR) and using eof to do the downscaling.

4.5 Predict & project

In `'esd'`, the S3 method `'predict'` is extended to the `'ds'` class and may be extended to CCA (`'cca'`) and singular spectrum analysis (`'ssa'`) in the future. The call `'predict'` will return the downscaled results for the calibration method by default, but can also be used to return a projection if the downscaling was based on a common EOF or a prediction based on a new EOF. The method `'project'` is a more specific version of `'predict'` that returns results from a projection (Example 4.4). The downscaled results from a projection are also contained in the `'ds'` object.

Example 4.1.

```
# Load 2m air temperature from NCEP reanalysis
t2m <- t2m.NCEP()
# Compute the EOFs
eof <- EOF(t2m)
# Plot the eof
plot(eof)
```

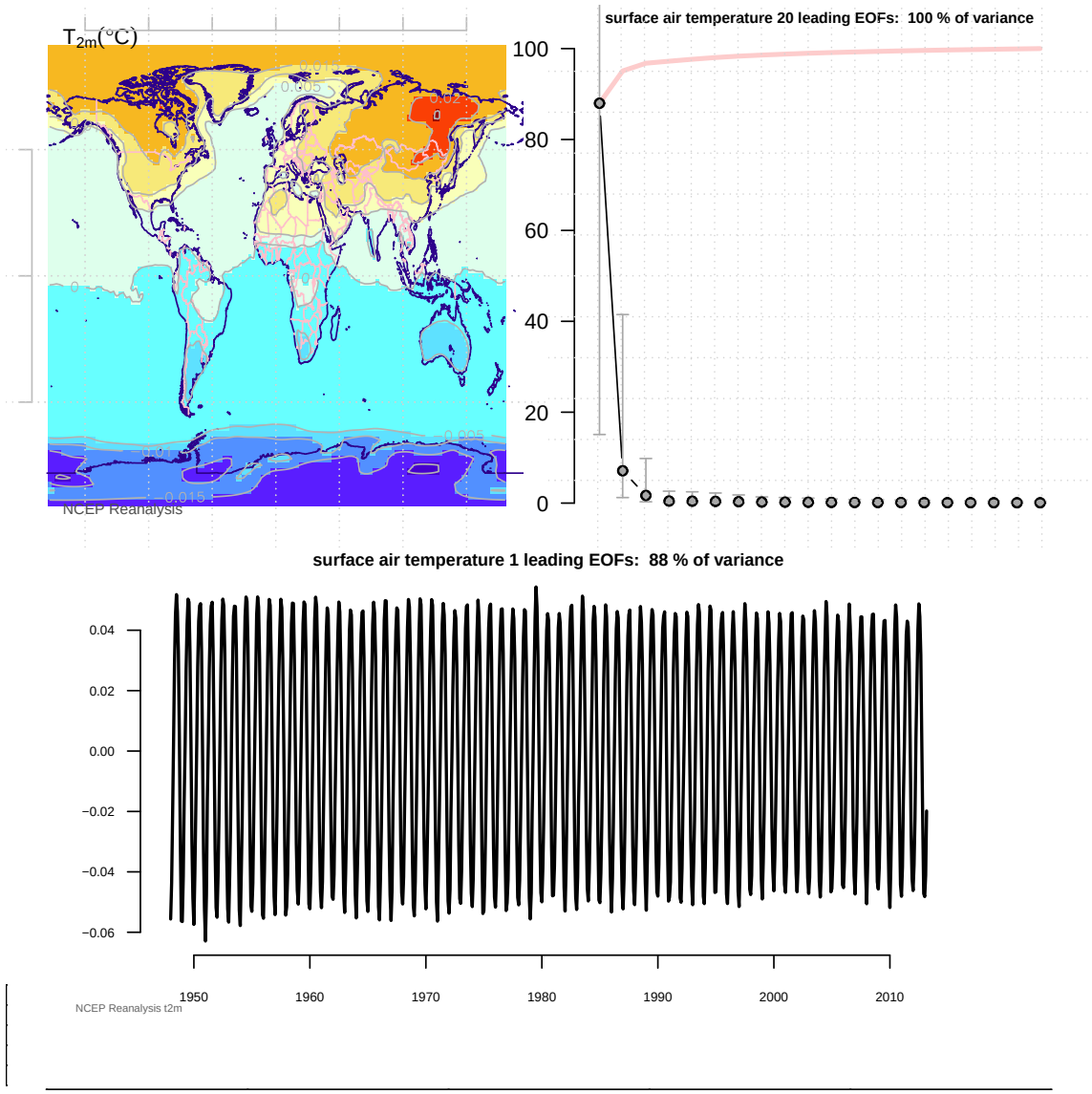


Figure 16: Plotting EOF analysis based on NCEP 2 meter surface temperature consisting of a map of the averaged field (top left), the explained variance by each EOF, and the principal component of the first leading EOF which accounts for almost 88% of the total variability.

Example 4.2.

```
# Retrieve NACD temperature weather stations
nacd <- station(src='nacd',param='t2m')
# Compute the annual mean values
NACD <- annual(nacd)
# Compute the number of valid data points for each station
ok<- apply(NACD,2,FUN='nv')
# Retain only the stations with minimum of 100 valid data points
NACD <- subset(NACD,is=(1:length(ok))[ok > 100])
# Do the PCA
pca <- PCA(NACD)
# Visualise the results
vis(pca)
```

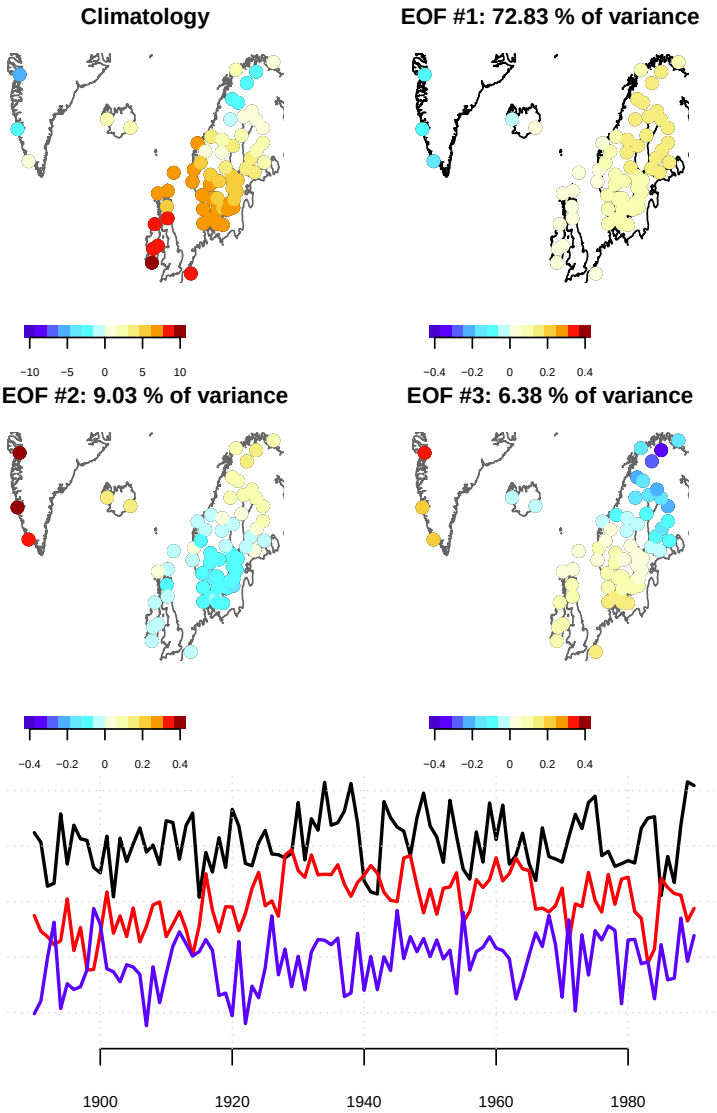
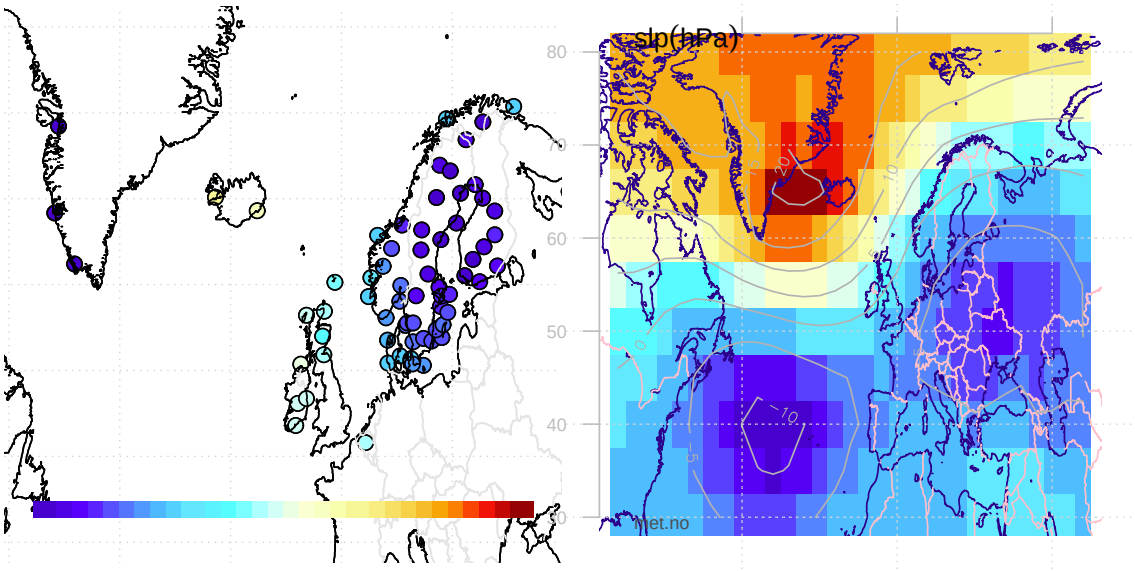


Figure 17: Summary of maps and plot showing the a) climatology, EOF 1 to 3, and the first three PCs computed from monthly surface temperature provided by the NACD dataset. The climatological map shows a clear south to north temperature gradient. The first leading EOF accounts for 72.83% of the total spatial variability followed by 9.03.31% and 6.38% for the second and third EOFs. The PCs do not show a significant trend.

Example 4.3.

```
# Get NACD stations
nacd <- station(src='nacd',param='t2m')
# Aggregate to annual values
NACD <- annual(nacd)
# Check for missing values
ok<- apply(NACD,2,FUN='nv')
# Subset NACD stations with more than 100 data points
NACD <- subset(NACD,is=(1:length(ok))[ok > 100])
# Compute the PCAs
pca <- PCA(NACD)
# Retrieve slp field and aggregate to annual values
slp <- annual(slp.DNMI())
# compute EOF of slp
eof <- EOF(slp)
# compute CCA on both pca and eof
cca <- CCA(pca,eof)
plot(cca)
```



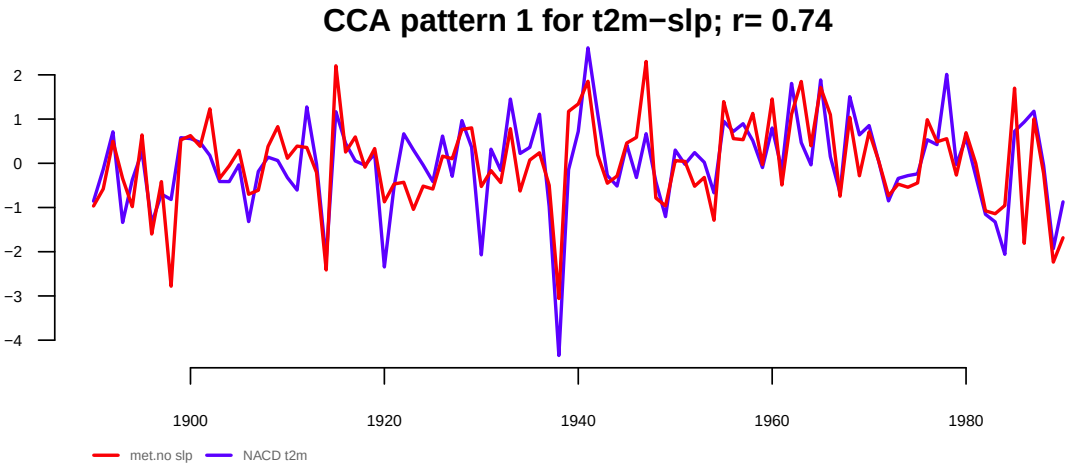


Figure 18: Plot of CCA analysis based on PCs of annual mean surface temperature and EOF of annual mean Sea level pressure from DNMI.

Example 4.4.
```
# Sample temperature data for Oslo
data(Oslo)
# Get ERA40 2m air temperature
t2m <- t2m.ERA40(lon=c(0,20),lat=c(50,70))
# Get NorESM.M 2m air temperature
T2m <- t2m.NorESM.M(lon=c(lon=c(0,20),lat=c(50,70)))
# Combine the two fields
comb <- combine(t2m,T2m)
# Compute the common eof t2m
ceof <- EOF(comb)
# Do the downscaling of y based on X
ds <- DS(Oslo,ceof)
# Subset the calibration
z.calibration <- predict(ds,newdata=EOF(t2m))
# Extract projected data sets
z.projection <- project.ds(ds)
```

4.6 Trajectory objects

The `esd` package includes functions for statistical analysis and visualisation of trajectory data, e.g., the paths of extra-tropical cyclones or ice bergs. Many of the standard tools and methods are applicable to `'trajectory'` objects, e. g., limiting the range of data by `'subset'`, visualising the trajectories by `'map'` or `'plot'`, as well as principal component analysis and downscaling (`'PCA'` and `'DS'`).

The trajectory methods have been designed with the analysis of storm tracks in mind, in particular cyclone path data produced under the IMILAST (Inter-comparison of mid latitude storm diagnostics) project (*Neu et al.*, 2012). Trajectory data that follow the format specified for IMILAST can be imported into `R` using the function `'read.imilast'`. The IMILAST data must then be transformed into a trajectory object with the function `'trajectory'` before other `esd` methods can be applied.

```
x <- read.imilast(filename,path=pathtofile)
y <- trajectory(x)
```

The function `'trajectory'` transforms a data frame containing spatio-temporal information about trajectories of variable length to a matrix with the trajectories interpolated to the same length (by default 10). The input to `'trajectory'` must for every time step include a `'Trajectory'` id number, as well as geographical (`'Lat'`*itude, `'Lon'`*gitude) and temporal (`'Year'`, `'Month'`, `'Day'`, and `'Time'`) information, and optionally a quality flag (`'Code99'`). Other parameters describing the evolution of the trajectory may also be included and will then be interpolated and passed on to the `'trajectory'` object. Meta data can be entered as arguments to the `'trajectory'` function, e.g. a description of the parameter (`'param'` or `'longname'`), the source of the data (`'src'`), or references to a paper, website or method (`'reference'`, `'URL'`, `'method'`).

A sample file, `'imilast.M03'`, containing trajectories of deep cyclones in the North Atlantic region comes with `esd` and is provided in its examples. These storm paths were obtained by

cyclone identification in a gridded data set (ERAinterim, 1.5° resolution) using a calculus based cyclone identification (CCI) algorithm (*Benestad and Chen*) (method 3 of IMILAST). In the future, `'esd'` will be extended to cyclone identification and tracking using this CCI method.

Example 4.5 demonstrates how to select a subset of trajectories and plot the annual storm count using the function `'plot.trajectory'`. The spatial extent of the trajectories can be plotted either as individual tracks on a map by the function `'map.trajectory'`, or as the number density (unit: year^{-1} 1000km^{-2}) by `'map.density.trajectory'` (Example 4.6). Principal component analysis of `'trajectory'` objects is by default applied to the longitude and latitude displacement with regards to the starting point of the individual trajectories (Example 4.7).

Example 4.5.

```
# Sample trajectory object
data(imilast.M03)
# Select storm trajectories in region 40-60N
x <- subset(imilast.M03,is=list(lat=c(40,60)))
# Select winter storms
djf <- subset(x,it='djf')
# Calculate the seasonal storm count
n <- count.trajectory(djf,by='year')

# Plot the annual storm count
plot(x,new=TRUE,ylim=c(0,60),col='black')
# Add storm count for winter (djf)
lines(n,col='blue',lty=2)
legend('topright',c('all year','djf'),lty=c(1,2),col=('black','blue'))
```

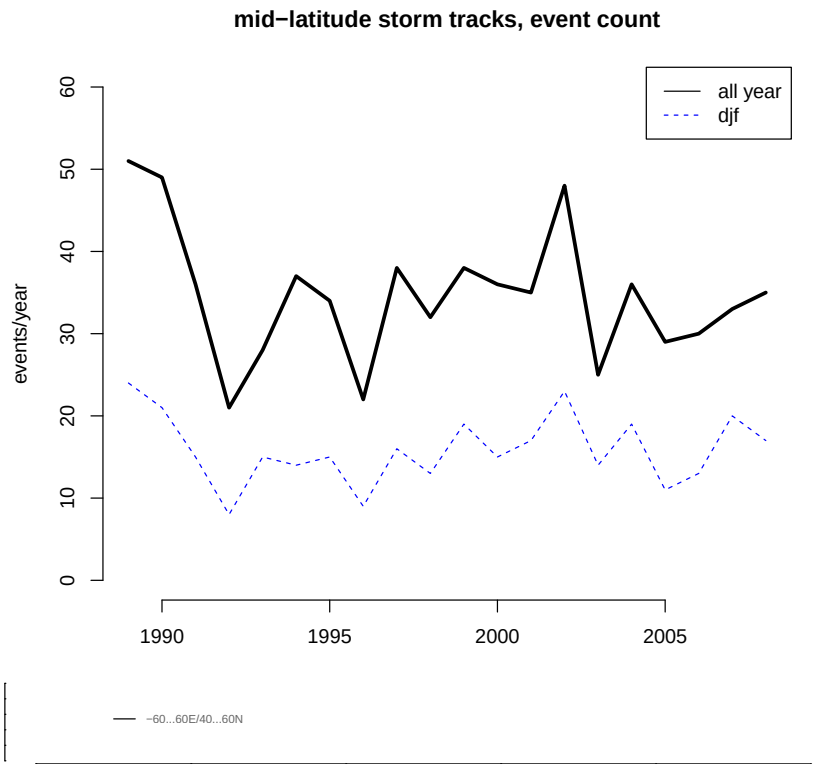


Figure 19: The annual storm count for all months (black) is plotted using `'plot.trajectory'`. A storm count for the winter season (DJF, blue) is then obtained by `'count.trajectory'` and added to the figure.

Example 4.6.

```
# Sample trajectory object
data(imilast.M03)
#Map storm trajectories for the winter season (djf)
map(imilast.M03,it='djf',projection='sphere')
# Map number density of storms (per year and 1000km2)
map.density.trajectory(imilast.M03,it='djf', xlim=c(-90,60),ylim=c(30,90),dx=4,dy=2)
```

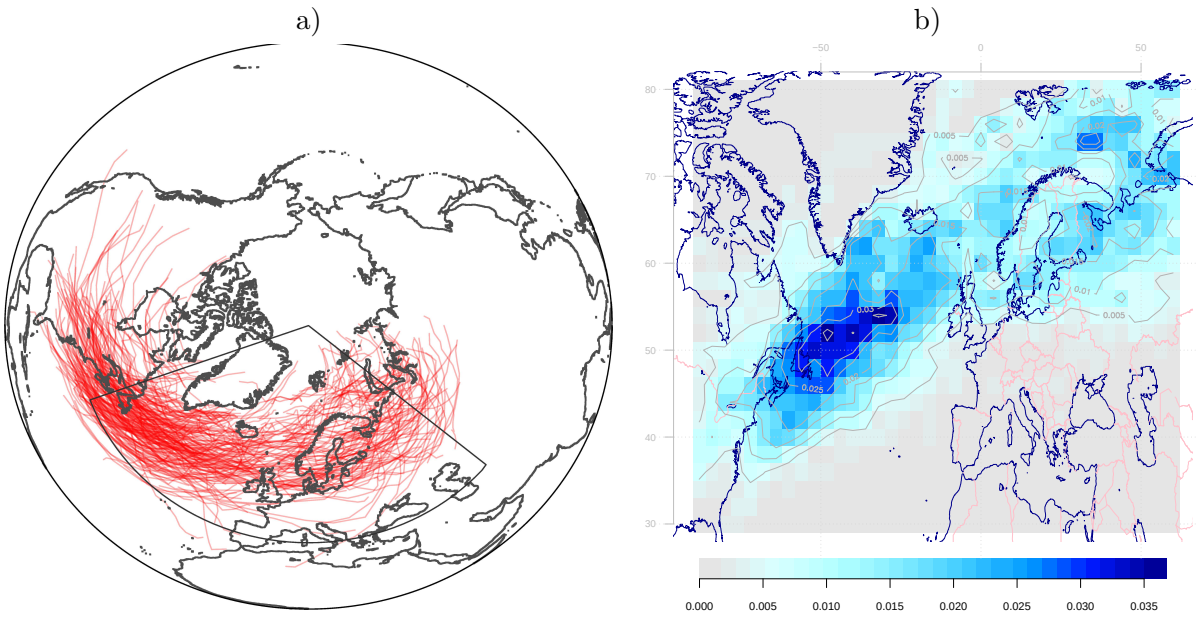
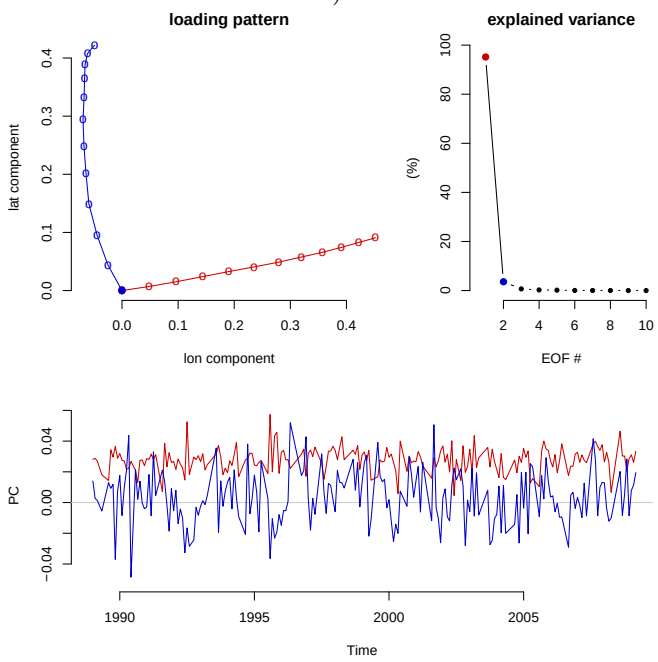


Figure 20: Maps of storm trajectories in the North Atlantic region created with '`map.trajectory`' (left) and '`map.density.trajectory`' (right).

Example 4.7.

```
# Sample trajectory object
data(imilast.M03)
# Perform PCA of storm tracks
pca <- PCA.trajectory(imilast.M03,anomaly=TRUE)
# Plot PCA
plot.pca.trajectory(pca)
# Show PCA on map
map.pca.trajectory(pca)
```

a)



b)

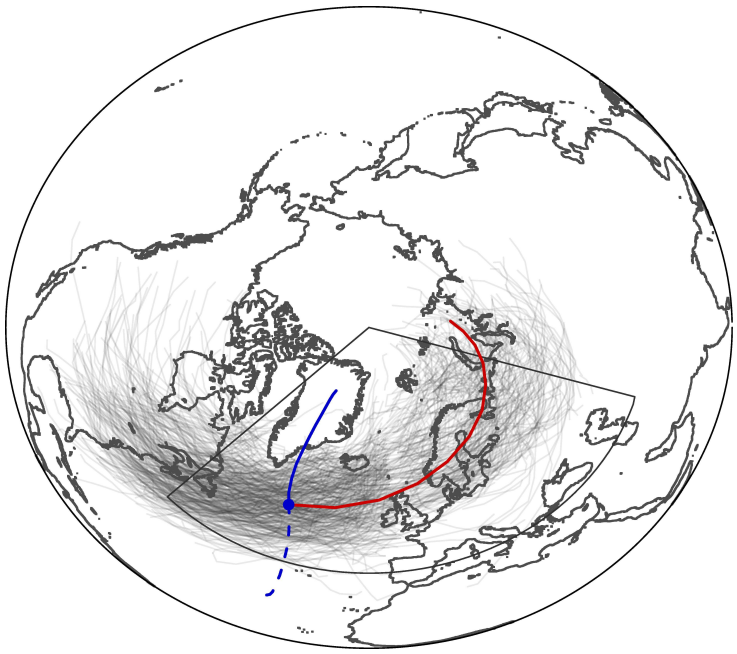


Figure 21: Plot of the first (red) and second (blue) principal component obtained by PCA of storm tracks in the North Atlantic region (a). In the map (b), the solid and dashed lines represent the maximum extent of the component in the positive and negative direction, respectively. The pca figures are produced with the functions '`plot.pca.trajectory`' (a) and '`map.pca.trajectory`' (b).

5 Downscaling

Downscaling tries to identify (statistical) existing relationships between large-scale spatial climate patterns (called predictors) and local climate variations (predictands) in climate variables, parameters, and/or indicators (indices).

5.1 Defining predictands

In downscaling based on `'esd'`, the predictand can be a station data (`'station'` class), a groups of stations (`'pca'`), gridded data (`'eof'`), or trajectory statistics. The strategy differs from that of `'clim.pact'` by downscaling seasonal or annual mean statistics, rather than monthly and daily. The reason for this is the idea that there is a stronger connection between local and large-scale climate than between local and large-scale weather. Here 'climate' is defined as weather statistics and involves parameters derived from a sample of some variable, e.g. the mean or frequency over a season or a year. It follows that statistics such as the mean is estimated more accurately with larger sample sizes according to the central limit theorem, reducing the effect of both sampling fluctuations and random errors. Furthermore, according to the central limit theorem, the mean converges to being normally distributed with increasing sample size. The seasonal and annual scales also imply more data points, especially for precipitation - it doesn't rain every day, and a monthly estimate may typically involve about 10 rain events - or 10 data points.

One motivation for using PCA to represent the predictand is that it brings forth the coherent variations and the signal in the data that is expected to have the strongest connection with the large scales (*Benestad et al.*, 2015). Furthermore, the use of PCA takes care of spatial cross correlation among the different stations.

The downscaling model only considers variability - the mean level is prescribed based on the observations and a given reference period. Hence, it is not affected by biases in the mean (a form for systematic errors). The climatology of the end-result is set by the climatology of the observations (base period) and the downscaled results describe the deviation from this climatology.

The downscaling uses common EOFs (*Barnett*, 1999) as a basis to ensure that the covariates used in the regression represent the exact same spatial structures (modes) when calibrating the regression model against reanalysis and when using the model to make predictions based on GCM results. The use of common EOFs is described in *Benestad* (2001).

5.2 Defining predictors

The large-scale predictors are represented in terms of EOFs and common EOFs in 'esd', where a regression analysis is used to estimate best-fit weights (regression coefficients β_i) whose

products with the PCs ($\hat{y}(t) = \sum_{i=1}^{n} \beta_i V_i(t)$) give the closest representation of the original time series y.

$$\hat{y}(t) = \beta_0 + \sum_{i=1}^{n} \beta_i V_i(t) \tag{3}$$

The term 'covariate' will henceforth be used to refer to the time-dependent variables on the right hand side of the multiple regression equation ($V_i(t)$).

The default is that the regression is based on ordinary linear model, but there is an option in the argument `'method'` to use generalised linear models or any other method that produces results with similar structure. The downscaling will also involve a forward-backward stepping, removing the PCs that do not increase the skill.

The downscaling can combine several predictors of different types by using a list object as the predictor argument, containing the EOFs describing the different sets of predictors. In Example 5.1, a combination of the T(2m) and SLP are used as predictors for downscaling.

The EOFs contained in the list are combined to a single `'eof'` class, and then subject to the default downscaling as if the predictor represented one single variable. The combination of the predictors involves a weighting of each principal component according to its eigenvalue, and then a singular value decomposition (*Strang*, 1988; *Press et al.*, 1989b) is used to distill the most pronounced and common variability embedded in the different data sets, in a similar way 'mixed' EOFs were used in *Benestad et al.* (2002) (or 'mixed' EOFs in `'clim.pact'`), described as 'CPCA' in *Bretherton et al.* (1992). Sets of several different types of predictors are combined in terms of their principal components (PC) according to the following strategy: the PCs for each set are weighted by a set of normalised weights to ensure that each type of predictor carry similar weight in addition to giving more weight to the leading EOFs that explain more of the variance (For each respective EOF, $\sum_i w_i = 1$). The weighted PCs are then combined and a singular value decomposition (SVD) is used to extract the common variability within the chosen predictors, and then the 20 leading modes are used to synthesise a data object that resembles the EOF object. This synthesised matrix is then used as a predictor that describes the most pronounced common/combined variability of the chosen predictor types.

5.2.1 Tuning

The downscaling can be tuned after the set of predictors is chosen. The predictor domain is by default set to a fixed spatial window (width to be specified by the user), but different choices for the domain will affect the results (*Benestad*, 2001). Also the number of EOFs included will affect the outcome, and the default is set to include the 7 leading modes. The number of EOFs can be determined manually as the first components with the maximum of variability (This can be achieved by looking at `'plot(eof)'`).

5.3 Options for downscaling

There are several downscaling approaches depending on the temporal resolution and the final objective of the impact study i.e. one can perform a DS for each year, season or month then combine the 12 months. For instance, annual downscaling is based on a new strategy which means extract annual statistics from daily or monthly data and do the downscaling.

5.3.1 Downscaling for single station and a single model

The basic downscaling is done for a single predictand (station) and a single predictor (EOF), and there are various 'wrappers' that use this elementary part as pieces of a more complicated process. The predictor can be a common EOF, constructed using `'combine'` on two fields and then an estimation of the EOFs. The downscaling can also be applied to a group of stations or gridded fields.

5.3.2 Downscaling a group of stations - PCA

One benefit of downscaling `'PCA'` objects (using `'DS.pca'` built-in function) rather than downscaling each single station separately is that the PCA-based analysis places more emphasis on the common signal found in adjacent stations and takes care about any strong spatial pattern that could be found in nearby stations. In other words, it may enhance the signal-to-noise ratio by separating out the signal (leading PCAs) from the noise (high-level PCAs). Another aspect is that the downscaling takes into account the spatial correlation structure, embedded in the PCA where each mode is orthogonal (*Benestad et al.*, 2015).

Example 5.2 and Figure 23 give a demonstration and a test of the downscaling of PCA.

5.3.3 Downscaling an ensemble of climate models

The method `'DSensemble'` is designed to deal with downscaling an ensemble of global or regional climate models such as CMIP3 and CMIP5 runs. The function `'DSensemble'` reads all data specified in a common folder (specified in the input arguments) to do the downscaling, and each result is saved separately for each climate model. Each file in the climate models' folder must be unique and must correspond to one climate model output stored in a NetCDF format. The GCM or RCM files should not be stored in several files spanning, for instance, different time slices. In this case, the files have to be concatenated before applying `'DSensemble'`. `'DSensemble'` has also been extended to deal with a group of stations' object (`'pca'`), with a speed-up and benefits associated with PCA.

The design of the `'DSensemble'` methods tries to meet some of the criticism that *Estrada et al.* (2013) presented against ESD that are summarised in the following issues.

a) The underlying probabilistic model assumed by an ESD method does not reflect the distribution of the underlying data.

b) The method requires *the variables* to be stationary.

c) ESD focuses on the short-term variability (Huth's dilemma).

d) Estimated relations of trend and/or auto-correlated series are at risk of being spurious.

e) Statistical downscaling is an empirical method and the statistical adequacy should be evaluated using appropriate tools.

f) Downscaling tool boxes seldom give any information regarding the significance of the estimated coefficients.

The downscaling is applied to samples of data aiming at predicting parameters describing the sample distribution. For temperature, the sample size is $N \approx 90$ data points (seasonal aggregates) whereas for precipitation, the downscaling is applied to annual aggregates because it doesn't rain every day (typical size $N \approx 100$). The idea is to downscale climate (defined as weather statistics) rather than weather, hence calibrating the models on aggregated parameters rather than day-to-day situations (issue 'c'). Furthermore, the primary parameters are the seasonal means for temperature and annual wet-day means and frequency for precipitation. According to the central limit theorem, the distribution of mean estimates is expected to converge to being normal with sample size (*Wheelan*, 2013). Hence, the default method (`'lm'`) is appropriate for such parameters (issue 'a'), although other methods such as GLM may be used for other types of distributions.

The usual requirement for ESD is that the transfer coefficients describing the link between the large and small scales are stationary, and not the variables themselves. Climate change by definition implies non-stationary variables, such a trend in temperature. In both `esd` and `clim.pact` the default is to de-trend the data before model calibration, in order to reduce the risk of spurious results (issues 'b' & 'd'). The auto-correlation between successive winters (or any other season) or years is low and forecasts for local temperature and precipitation aggregated for a year ahead is notoriously difficult as a result, which implies that auto-correlation has little effect on the outcome of the downscaled results in the default setting (issue 'd'). When it comes to model evaluation tools (issue 'e'), the method `'DS'`, on which `'DSensemble'` is based, uses cross-validation by default for assessing its skill. The R-environment also offers a set of tools for testing whether the data used for model calibration are normally distributed: e.g. `'qqnorm'`. A Shapiro-Wilk test of normality is also included in `'diagnose.ds'` . Furthermore, `'DS'` invokes a step-wise screening that removes the predictors which do not make a statistical significant contribution to the regression (issue 'f'), and `esd` includes a function that carries out an iid-test which can be applied to de-trended series: `'iid.test'` (*Benestad*, 2008).

5.4 Downscaling of trajectory objects

The function `'DS'` can be applied to trajectory objects, but it is not the paths themselves that are downscaled but rather some statistical measure of each trajectory (Example 5.4, Figure 25). By default, the downscaling is applied to the monthly trajectory count. Trajectory objects may also be downscaled with regards to other aspects by adding a parameter name

(‘param’) and a function to apply to the parameter for each path (‘FUN’). For example, we can calculate and downscale the genesis latitude (param=’lat’, FUN=’first’) or the mean longitude (param=’lon’, FUN=’mean’).

Example 5.1.

```
# Load eofs of NCEP 2m air temperature
data(eof.t2m.NCEP)
# Load eofs of NCEP sea level pressure
data(eof.slp.NCEP)
# Load temperature data for Oslo
data(Oslo)
# Do the downscaling
z <- DS(Oslo,list(t2m=eof.t2m.NCEP,slp=eof.slp.NCEP),mon=1)
# Plot the results
plot(z)
```

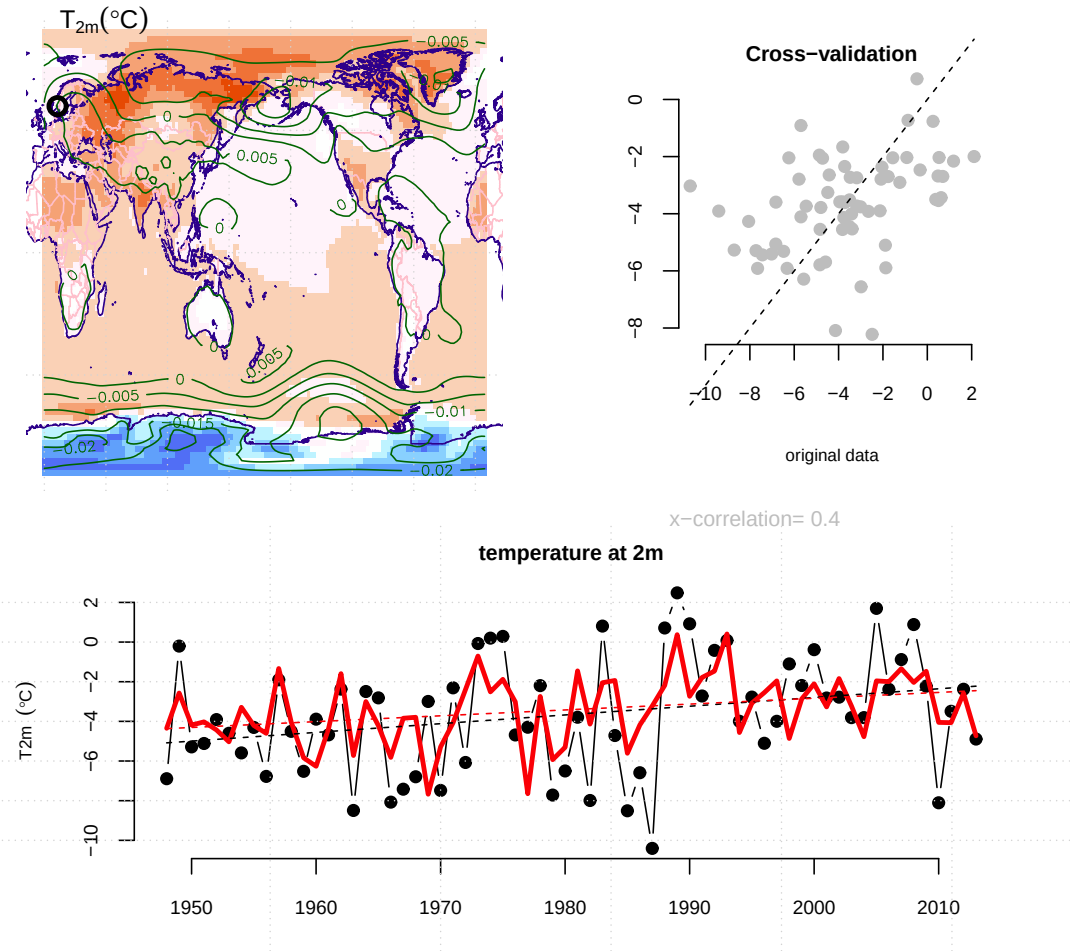


Figure 22: An example of downscaled annual mean temperature in Oslo, based on the ERAINT reanalysis and using PCA for downscaling a group of stations simultaneously.

Example 5.2.

```
# Load ERAINT 2m air temperature from file
T2m <- t2m.ERAINT(lon=c(-10,30),lat=c(50,70))
# Compute the EOFs on annual values
eof <- EOF(annual(T2m))
# Retrieve Nordklim data sets
y <- station(src='nordklim')
# Get rid of stations with little valid data
Y <- allgood(annual(y))
# Do the PCA
pca <- PCA(Y)
# DO the downscaling
z <- DS(pca,eof)
# Plot the result
plot(z)
# Get the observations corresponding to the first station in the list
obs <- subset(Y,is=1)
# Get the predictions
pre <- subset(as.station(z),is=1)
# Match dates between obs and pre
obs <- matchdate(obs,pre)
# Update the attributes
attr(obs,'location') <- 'Observed'
attr(pre,'location') <- 'Predicted'
# Combine the two
obspre <- combine.stations(obs,pre)
# Plot the result in a single plot
plot(obspre,plot.type="single")
```

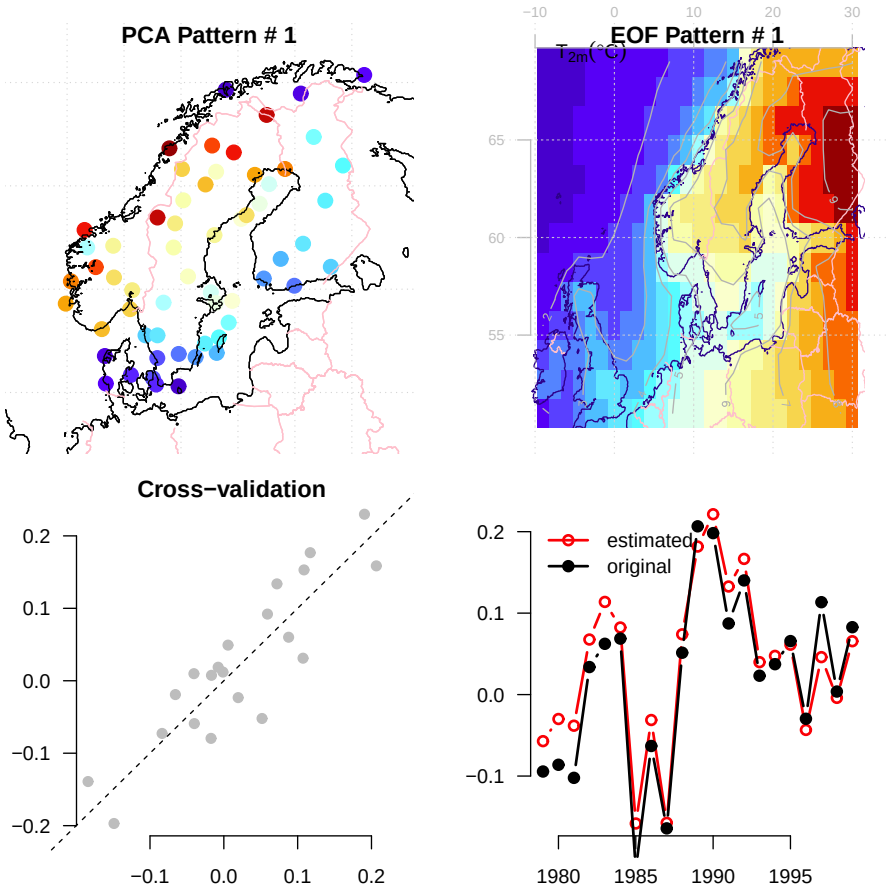


Figure 23: An example of downscaled annual mean temperature in Oslo, based on the ERAINT reanalysis and using PCA for downscaling a group of stations simultaneously.

Example 5.3.

```
# Get the predictand: Maximum temperature for some Norwegian sttions
download.file(url="http://files.figshare.com/2073466/Norway.Tx.rda",destfile="NorwayTx.rda")
load("NorwayTx.rda")
# Process the predictand (annual mean)
ma <- annual(Tx)
pca <- PCA(ma)
# Get the ERAINT T(2m):
t2m <- t2m.ERAINT(lon=c(-30,30),lat=c(50,75))
# Process the predictor: Compute EOFs of annual means
eof <- EOF(annual(X))
# Test: CCA:
cca <- CCA(pca,eof)
# Do the downscaling
z <- DS(pca,eof,rmtrend=FALSE)
# Plot the results for one station:
plot(subset(ma,is=1))
lines(subset(as.station(pca),is=is),lty=2,col="darkred")
lines(subset(as.station(z),is=is),lwd=2)
```

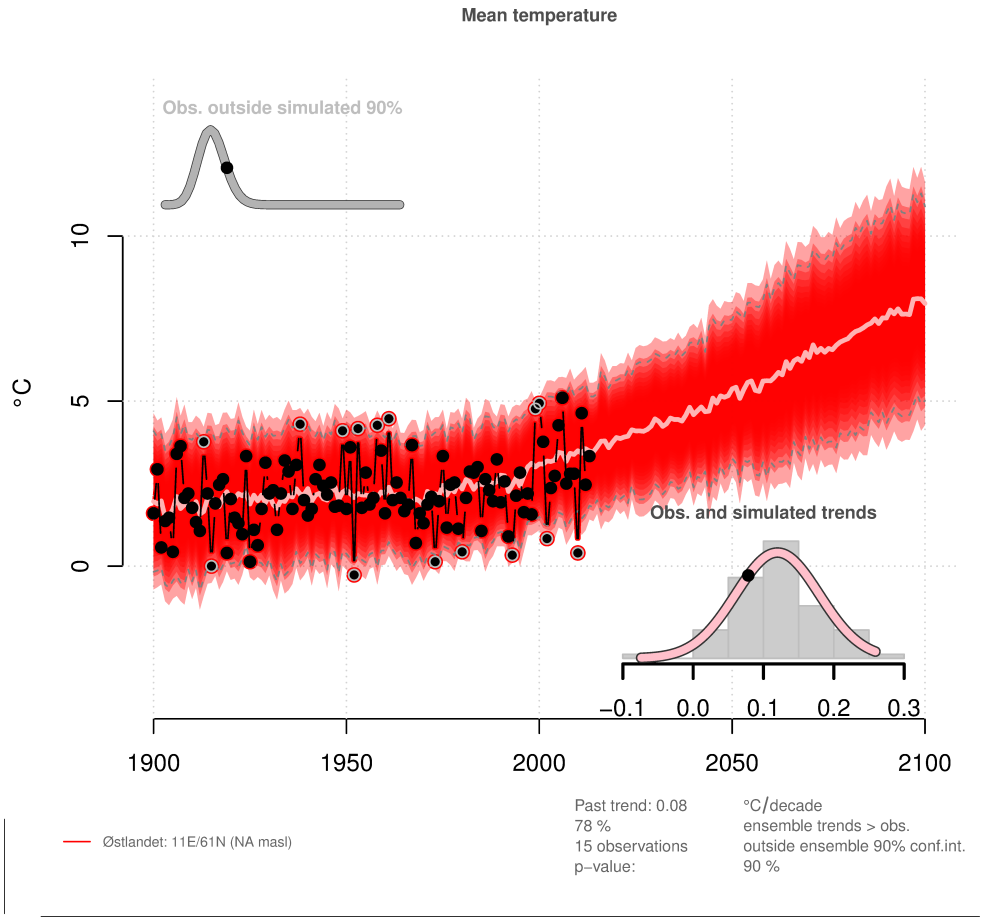
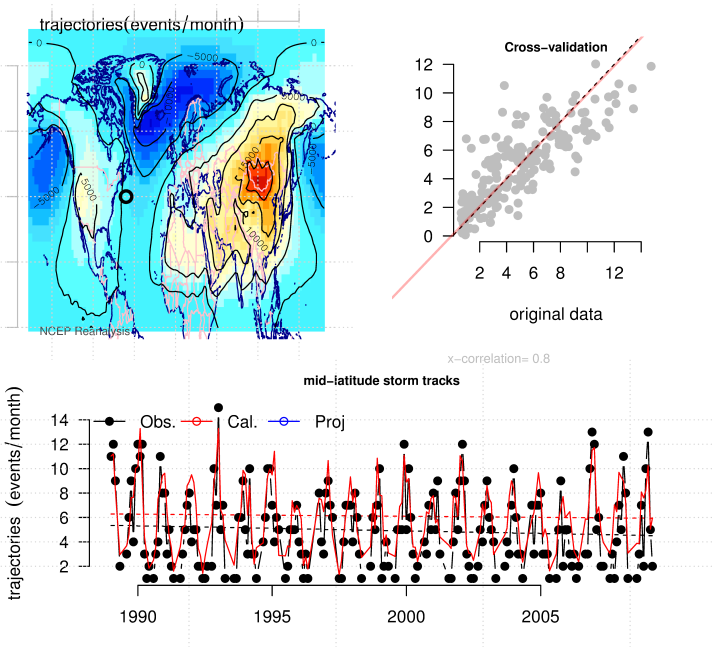


Figure 24: An example of downscaled CMIP5 RCP8.5 DJF seasonal mean temperature eastern Norway. The ERA40 reanalysis is used here for calibration. The inner plot in the right bottom of the figure shows a comparison between the estimated linear trend from both simulations (in terms of distribution) and observations (black point). While, the inner plot on the top right of the figure shows the distribution of the number of observations lying outside the 90% of the confidence interval based on simulations.

Example 5.4.

```
# Load storm trajectories
data(imilast.M03)
# Load NCEP sea level pressure data for the northern hemisphere
slp <- slp.NCEP(lat=c(0,90))
# Compute eof
eof.slp <- EOF(slp)
# Do downscaling for the storm count  (default trajectory downscaling)
ds <- DS(imilast.M03,eof.slp)
# Do downscaling from slp
ds.slp <- DS(imilast.M03,eof.slp,param='slp',FUN='min',unit='hPa')
# Plot downscaled results
plot(ds)
plot(ds.slp)
```

a)



b)

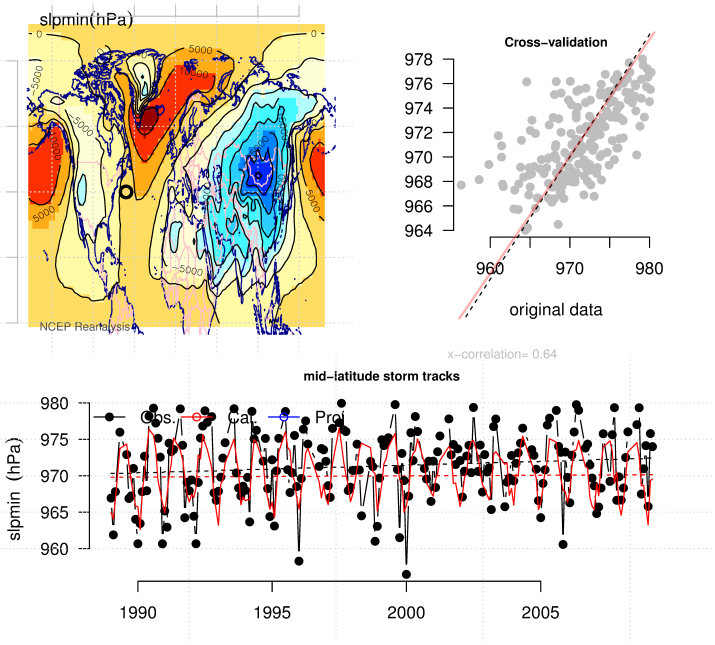


Figure 25: Two examples of downscaling the storm tracks of the North Atlantic region with regards to the storm count (a) and minimum slp (b).

5.5 Downscaling probabilities and number of events

The original time series can be summarised as the probability distribution of a climate variable or parameter, the latter can then be downscaled instead of the original time series. For instance, the 24-hour wet-day precipitation amount is expected to follows an exponential or gamma distribution, whereas the temperature is more likely to follow a normal distribution, and the wind-speed a Weibull distribution. The count of events is expected to follow a binomial or Poisson distribution and spell-length statistics tends to have a geometric distribution. The predictands here are the parameters of the probability distribution function rather than the original climate variable. These are mainly the first order and second order moments such as the mean (for exponential distribution), the mean and the standard deviation for the normal distribution, and the mean and the coefficient of variation for the gamma distribution.

Downscaling of μ, f_w, and n_{cwd} (wet spell length) require different choices in the type of regression model because they involve different types of data (*Estrada et al.*, 2013). The set-up may involve continuous values, fractions ($f \in [0, 1]$), and discrete counts. sometimes it may be difficult to decide which category to use. For example, the annual wet-day frequency is a mean estimated over a set of zeros and ones and exhibits a character closer to normal than logistic distribution, but while each day would be subject to a logistic distribution, its annual mean is subject to the central limit theorem.

5.6 Weather generators - synthesising daily time series

Weather generators are usually used to simulate realistic sequences of daily weather and climate variables such as precipitation, maximum and minimum temperature for present (*Semenov and Barrow*, 1997; *Wilks and Wilby*, 1999) and future climate conditions (*Wilks*, 1992). Weather generators can be seen as complementary tools and are used in conjunction with downscaling techniques. They use (downscaled) parameters of the several distribution functions of climate variables. The generation process differs between climate parameters, variables and statistics. For precipitation, most weather generators will generate first dry and wet sequences, and then for the wet days, the precipitation amount is generated following different probability and using different random numbers form different distributions (*Mezghani and Hingray*, 2009). The simulation process takes into account the various specific aspects of climate variability and change (*Soltani and Hoogenboom*, 2003). The simulated sequences must share the main statistical properties of the observed or original time series used for calibration (*Semenov and Brooks*, 1999). The general form of a WG can be written as follows:

$$Y = f(X, \epsilon), \tag{4}$$

where ϵ is an ensemble of random variables which contains the remaining information that has not been taken into account by X and the random processes that is involved.

There are various ways a weather generator can be designed, and `'esd'` may include more types in the future. Presently, there are two different types, one for precipitation and one for temperature. These are designed so that they are conditioned upon downscaled results for seasonal of annual scales, using vital parameters to disaggregate the results to daily values. The results will span a similar time interval as the provided observations x but may be shifted in time if a set of projections is provided (taking an interval that ends on the last projected year).

5.6.1 Stochastic precipitation

The stochastic or generation process for precipitation uses information about wet-day mean, frequency, and consecutive wet days to generate stochastic precipitation series. The call is `'WG.fw.day.precip'` and its features are (roughly):

- Estimate wet-day mean and frequency as well as the wet- and dry-spell statistics for a station.
- Uses phase-scrambled Fourier transform of annual observed/downscaled wet-day mean and frequency.
- Estimate the number of wet days per year.
- Wet days: amounts prescribed according to the exponential distribution and μ.
- Try to distribute wet days according to both the number of wet days per year and with a wet-spell statistics according to annual mean number of consecutive wet days assuming a geometric distribution.

5.6.2 Stochastic temperature

The weather generator for daily temperature is `'WG.FT.day.t2m'`, currently designed for single stations. The key features of this weather generator are (roughly):

- Uses phase-scrambled Fourier transform of observed or downscaled annual mean temperature anomaly to produce a random order of annual data.
- Uses phase-scrambled Fourier transform of observed or downscaled annual standard deviation of daily temperature anomalies to produce a random order of annual data.
- Uses quantile-quantile mapping to generate normally distributed stochastic variables conditioned on prescribed mean and standard deviation.

6 Evaluation, assessment & validation

6.1 Central limit theorem

The central limit theorem states that the distribution of the sum or average of a large number of independent, identically distributed variables (iid) will be approximately normal, regardless of the underlying distribution.
(src. `http://www.math.uah.edu/stat/sample/CLT.html`)

6.2 Huth's dilemma

Huth's dilemma is the situation where ESD is calibrated on short-term (fast) fluctuation but is not capable of predicting long-term (slow) changes. This is a problem if there are different processes responsible for variations at different time scales. One test to provide a diagnose on whether this is the case is to calibrate the model on de-trended data and then use the original field as input to see if it is able to predict the long-term trend over the calibration period.

In `'esd'`, the default process de-trends the data before calibration. The downscaled results are derived with the original field, however, and trends are compared against the original observations. Furthermore, the question of stationary signal should be examined in the context of climate model results, and the past changes seen in the observations should be compared with corresponding downscaled results from GCM historic runs. Ensembles of GCMs are particularly suitable for assessing the ESD model's ability to capture the long-term changes.

6.3 Non-stationarity check

The non-stationarity check in `'esd'` treats the GCM results as pseudo-reality, where results interpolated to the coordinates as a given station is used as the predictand. The same time interval as the reanalysis data used for calibrating the model is extracted from the GCM for representing the calibration predictors. The model trained on the subset of GCM results is then used to predict future values for the predictand, and the predictions from the downscaling of the pseudo-reality are compared with corresponding results derived by the interpolation.

The downscaled pseudo-reality can also be compared with the results downscaled for the actual station data. This is done in `'DSensemble'` when the argument `'non.stationarity.check'` is set to `'TRUE'`.

6.3.1 iid.test

The `iid.test` included in the `esd` package is used to test whether a variable is independent and identically distributed (iid) mainly for daily station records (*Benestad*, 2003, 2004). It has to be noted that this test is sensitive to missing data (NA) and can produce an under-count. A non i.i.d. behaviour appears when the 'forward' (resp. 'backward') analysis indicates higher (resp. lower) number of record-events than the confidence interval.

6.4 Diagnose

The `'diagnose'` method provides a range of different approaches for the evaluation of `'esd'` results depending on the type of objects. For instance, it returns diagnostics of common EOFs or cross-validation of DS results. The diagnostic also includes combined fields, MVR, and CCA results by applying appropriate internal tests to check for consistency. For instance, when applied to a combined `'eof'` object, `'diagnose'` investigates the difference in the mean between the PCs of the calibration data and the PCs of the GCMs over a common period in addition to the ratio of standard deviations and lag-one auto-correlation. A bias correction method can then be applied if the difference is significant as described in (*Imbert and Benestad*, 2005).

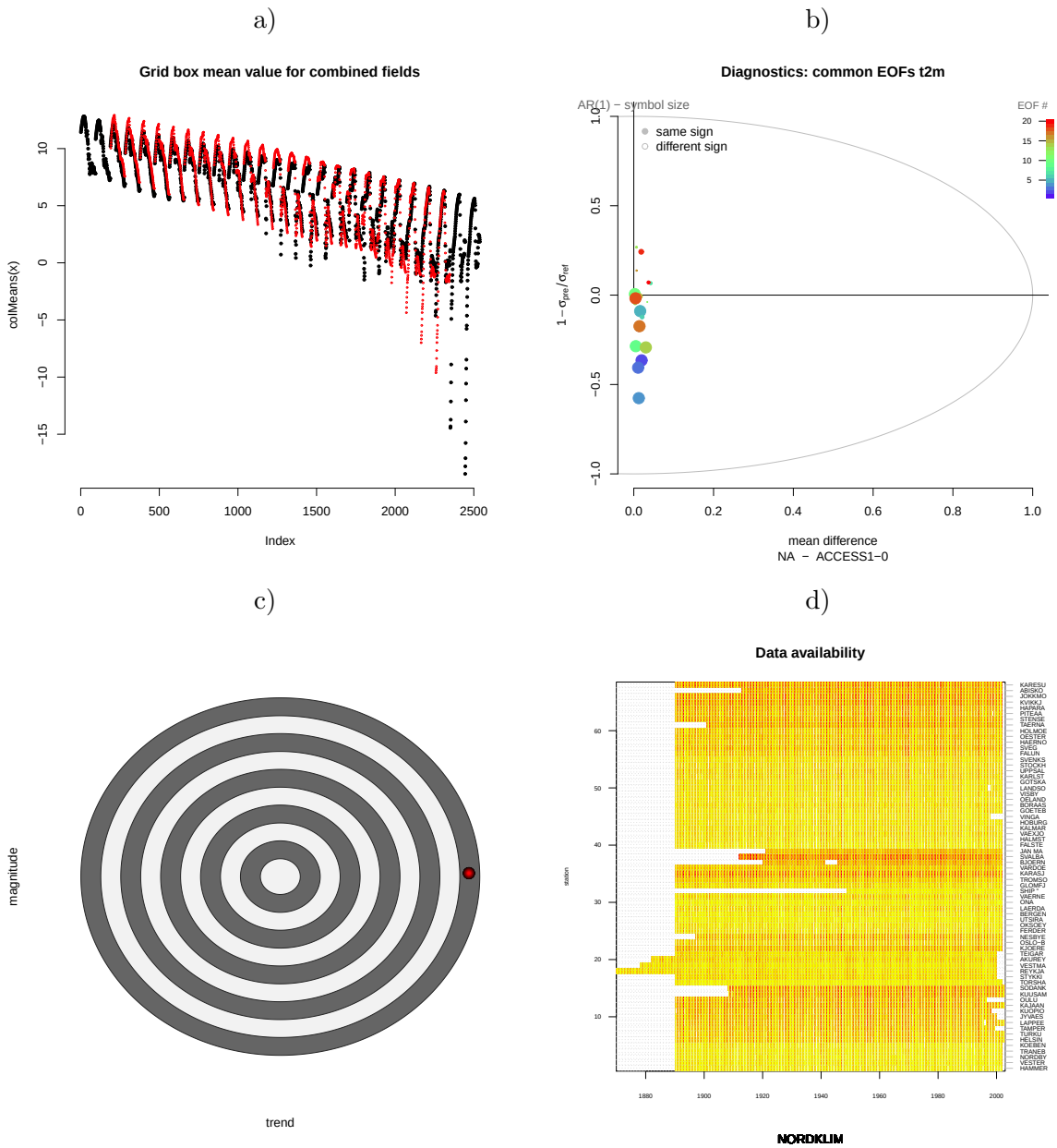


Figure 26: The results of `'diagnose'` applied to four different data objects: (a) common field, (b) to common EOFs, (c) downscaled results for an ensemble, and (d) a station.

6.4.1 Station

The `'diagnose'` for station object offers another way of plotting the information contents and is designed for a group of stations to indicate the availability of valid data (Example 6.1, Figure 26d).

6.4.2 Combined fields

The application of `'diagnose'` to a combined pair of field objects will produce two comparable curves that show their spatial mean values (Example 6.2, Figure 26a). This way, the spatial statistics of the two fields can be compared.

6.4.3 Common EOFs

For common EOFs, `'diagnose'` returns a set of different diagnostics, such as mean differences, the standard deviation ratio, and different auto-correlation for the principal components over an overlapping interval (6.3).

Example 6.1.

```
# Retrieve NACD data sets
nacd <- station(src='nacd')
# Diagnose for data availability
diagnose(nacd)
```

Example 6.2.

```
# Sample 2m air temperature ERAINT field from EOF's data
t2m <- t2m.ERAINT(lon=c(-30,40),lat=c(50,70))
# Retrieve GCM air temperature from file
# Sample 2m air temperature NorESM field from EOF's data
gcm <- t2m.NorESM.M()
# Combine annual fields
X <- combine(annual(t2m),annual(gcm))
# Diagnose the result
diagnose(X)
```

Example 6.3.

```
# Compute EOF, X is taken from previous example
eof <- EOF(X)
# Diagnose eof
a <- diagnose(eof)
# Plot the results
plot(a)
```

6.5 Downscaled results

The quality of the downscaled results can be investigated and checked with the function `'diagnose.ds'` (Example 6.4). In addition cross-validation and residuals can be used to evaluate the downscaling procedure.

6.5.1 Cross-validation

A default option is to perform a five-fold cross-validation test (`'crossval'`)), but it can also be set to perform the CORDEX-ESD experiment 1 or VALUE experiment 1 set-up. The results of

the cross-validation consist of a zoo-object that is stored in the attribute `'attr(*,'evaluation')`.

6.5.2 Residuals

The residuals of the downscaling should ideally be a white noise that has no auto-correlation, no trend, and is normally distributed. These conditions can be tested using auto-correlation functions and time series and qqnorm plots.

6.5.3 Downscaled ensemble results

For ensembles `'diagnose'` shows a comparison between DS results derived from GCMs and the past variations. It compares the number of observed values outside the predicted 90% confidence interval based on fitting a normal distribution to the ensemble member values for each year, in addition to carrying out an evaluation of trends based on the rank the observed trend has compared to corresponding trends simulated by the ensemble for the past. The diagnostics produces an info-graphic in the form of a target (Example 6.5, Fig 26d).

The count of the cases where observations fall outside the 90% confidence interval is expected to follow a binomial distribution if the model results reproduce the correct statistics (H_0). Hence, $p = 0.1$ for falling outside, and the test is to see if the actual count of cases outside the confidence interval is consistent with the H_0 binomial distribution (*Benestad and Mezghani*, 2015).

Example 6.4.
```
# Load temperature data for Oslo
data(Oslo)
# Get the 2m air temperature from NCEP reanalysis
t2m <- t2m.NCEP(lon=c(-5,20),lat=c(50,65))
# Compute the EOF on annual temperature values
eof <- EOF(annual(t2m))
# Downscale the annual values
z <- DS(annual(Oslo),eof)
# Do the diagnostic
diagnose(z,plot=TRUE)
```

Example 6.5.
```
# Download the file from the figshare link as
download.file(url="http://files.figshare.com/2081407/dse.oslo.rda",destfile="dseOslo.rda")
# Load the file into R session
load('dseOslo.rda')
# Do the diagnostic
diagnose(dse.oslo)
```

7 Trouble shooting

The ‘esd’ comes as it is and there is no guarantee that it’s free from bugs or incorrect coding. The user therefore must make sure that the functions work as expected through thorough testing. Any tool like this should never be used blindly. Neither is there any guarantee that the code will work flawlessly. The open-source nature, however, means that the problems can be diagnosed or fixed by anyone with insight and programming skills. Here we try to make potential trouble shooting easier.

7.1 General

In case of error messages, it may be wise to check all the attributes of the data object. The way to proceed is to activate the ‘verbose’ argument if it is included in the function options, and repeat the analysis with the sample data provided by ‘esd’. The objects can be assigned new attributes through the ‘attr’ command. Alternatively, the source code of the function can be modified locally, by typing the name of the function without ‘()’ and copied into a local file (Example 7.1). The code in the local file can then be modified and when it is called override the original ‘esd’ version.

Example 7.1.

```
# get the source code in 'myeof.R' R script file
dput(EOF.default,file="myeof.R")
# Modify this code, but keep the same function name by adding 'EOF.default' at the top of
# the file.
# Source the new code as
source("myeof.R",local=environment())
# now the call "EOF()" will use the modified version.
```

Debugging can be done by inserting ‘browser()’ inside the function code.

7.2 Functions ‘annual’ and ‘aggregate’

Missing data (‘NA’) can be a problem in the analysis and the downscaling of local temperature and precipitation. One example of an error is:

```
y <- annual(x)
Error in aggregate.data.frame(as.data.frame(x), ...) :
no rows to aggregate
```

However, if the number of missing data is small compared to the sample size, they may not have a large effect on the calibration of downscaling models or the aggregated statistics. For the temperature, a fix can be to use interpolation to fill in the missing values as in Example 7.2. For daily precipitation, a better option can be to replace missing data with zero.

Example 7.2.

```
# Retrieve 2m air temperature at Ferder Fyr weather station
data(ferder)
# Extract the coredata
z <- coredata(ferder)
# identify the missing days in the record
md <- is.na(ferder)
# Compute the climatology
clim <- as.climatology(ferder)
# Replace the missing values with the climatology as approximations
ferder[md] <- coredata(clim)[as.POSIXlt(index(ferder)[md])$yday+1]
# Note that as.POSIXlt()$yday function returns the day in the year
```

7.3 DSensemble

The method `'DSensemble'` uses `'try'` to avoid that the process stops due to errors when looping through the ensemble of GCM results. However, it cannot recover from errors in Fortran or C-based external modules, such as those based on external libraries. For instance, the following errors have been encountered:

```
Error in La.svd(x, nu, nv) : error code 1 from Lapack routine 'dgesdd'
```

and/or

```
Error in x[rep(1:NROW(x), length.out = length(index)), , drop = FALSE] :
 subscript out of bounds
```

The problem is that the data matrix to which an SVD is applied is close to singular. One work-around fix is to use a different domain selection (e.g. set `'lon'` and `'lat'` arguments to new values). Another way to get around this problem is to use the argument `'select'` to skip a particular model with the problem.

7.3.1 Poor fit

Sometimes a sub-selection of the predictand can improve the calibration of the downscaling model, especially if part of the data (older) has a lower quality than more recent measurements. One way to examine the data for quality can be to plot the time series of the annual wet-day mean and frequency for precipitation.

```
plot(annual(y,FUN='wetfreq'))
plot(annual(y,FUN='wetmean'))
```

7.3.2 Other error messages

There may occasionally be problems associated with different types of time stamps, e.g. when combining daily, monthly, seasonal, or annual data. The time index for annual data is a numeric - the year, and some of the methods handling the date may fail for annual data. E.g:

```
Error in prettyNum(.Internal(format(x, trim, digits, nsmall, width, 3L,  :
invalid 'trim' argument
```

A couple of solutions are:

```
  index(y) <- as.Date(paste(year(y),'01-01',sep='-'))
## or
  index(y) <- year(y)
```

7.4 Validate

The `'esd'` package also has a method `'validate'` which provides a simple test statistics describing how well the results match the expectations. The null hypothesis depends on the type of object.

8 Summary

We have presented a new and promising open source R package that includes a set of climate analysis and statistical downscaling methods for use by the scientific and academic communities. The tool, named `'esd'`, has been designed to be useful not only for the climate community but also for meteorologists, hydrologists, and impact users. It can also be tailored to meet with user needs and requirements. In addition, `'esd'` allows performing a quick and valuable analysis without much prior knowledge of programming.

The different analysis methods are quick, efficient, and easily reproducible in order to allow the users to work with more confidence.

One of the strength of the `'esd'` package is that it includes methods and functions to implement all the steps needed to construct a statistical downscaling framework specifically tailored to the user needs. For this purpose, the package allows

1. exploring the data to define a set of appropriate predictands and predictors,
2. searching for links and establishing connections between large scale predictors and local scale predictands,
3. and downscaling, evaluating and projecting weather climate variables and parameters into future.

Another strength of the tool is that it can retrieve meteorological observations from weather stations all over the world, perform various analyses such as those based on the principal components of a univariate or multivariate data sets, compute empirical orthogonal functions, and perform a statistical downscaling of global climate model outputs. All these capabilities are included in one package and thus share the same language. The tool also includes predefined downscaling methods for precipitation and temperature. Many of the methods are also applicable to trajectory data which can be useful in the analysis of, e. g., storm tracks.

A web page is available from which users can find valuable information on how to install the package, try some examples and check the results. The web page has been implemented using the social coding interface Github. The wiki-page containing valuable information can be found following the web link `https://github.com/metno/esd/wiki`. A user guide including a full list of functions included in the package is made available also at the Github wiki page `https://github.com/metno/esd/blob/master/esd.pdf`

Although the source code has been made open and free, all rights are reserved to the Norwegian Meteorological Institute that is supporting this initiative.

We believe that `'esd'` will be very useful for research as well as for teaching activities. It is a versatile and easy to use programming tool, allowing users in the climate and related communities to make direct and intuitive climate analysis and downscaling tailored to their needs. We encourage people from and outside the Norwegian Meteorological Institute to send us their feedback (https://goo.gl/agfK1X).

9 Acknowledgement

The development of the 'esd' package has been supported by the Norwegian Meteorological institute as well as a number of projects: SPECS, INDICE, Chase-PL, CLIPC, COST-VALUE, and STATKRAFT (MIST II). The tool will be further developed under CiXPAG and EU-Circle, but it has also been developed in conjunction with IMILAST and CORDEX-ESD. The financial support includes the Norwegian Research council, EU (FP7 & Horizon 2020) and bilateral Norway Grants/EEA Grants. We want to thank Dr. Lalu Das, Monami Dutta and Jitendra Kumar for their feedback and patience in being pilot users.

References

Barnett, T., and R. Preisendorfer (1987), Origins and levels of monthly and seasonal forecast skill for united states surface air temperatures determined by canonical correlation analysis, *Monthly Weather Review*, *115*, 1825–1850.

Barnett, T. P. (1999), Comparison of near-surface air temperature variability in 11 coupled global climate models, *Journal of Climate*, *12*, 511–518.

Benestad, R. (2008), A simple test for changes in statistical distributions, *Eos*, *89*(41), 389–390.

Benestad, R., and A. Mezghani (2015), On downscaling probabilities for heavy 24-hr precipitation events at seasonal-to-decadal scales, *Tellus A*, *67*(25954).

Benestad, R., D. Nychka, and L. O. Mearns (2012), Spatially and temporally consistent prediction of heavy precipitation from mean values, *Nature Climate Change*, *2*(doi: 10.1038/NCLIMATE1497).

Benestad, R., D. Chen, A. Mezghani, L. Fan, and K. Parding (2015), On using principal components to represents stations in empirical-statistical downscaling, *submitted to Tellus A*.

Benestad, R. E. (2001), A comparison between two empirical downscaling strategies, *Int. J. Climatology*, *21*, 1645–1668, DOI 10.1002/joc.703.

Benestad, R. E. (2003), How often can we expect a record-event?, *Climate Research*, *25*, 3–13.

Benestad, R. E. (2004), Record-values, non-stationarity tests and extreme value distributions, *Global Planetary Change*, *44*(doi:10.1016/j.gloplacha.2004.06.002), 11–26.

Benestad, R. E. (2011), A new global set of downscaled temperature scenarios, *Journal of climate*, *24*(8), 2080–2098.

Benestad, R. E., and D. Chen (), The use of a calculus-based cyclone identification method for generating storm statistics, *Tellus A*, *58*(4), 473–486.

Benestad, R. E., I. Hanssen-Bauer, and E. J. Førland (2002), Empirically downscaled temperature scenarios for svalbard, *Atmospheric Science Letters*, *3, Issue 2-4*(doi.10.1006/asle.2002.0051), 71–93.

Benestad, R. E., I. Hanssen-Bauer, and D. Chen (2008), *Empirical-statistical downscaling*, World Scientific.

Bretherton, C., C. Smith, and J. Wallace (1992), An intercomparison of methods for finding coupled patterns in climate data, *Journal of Climate*, *5*, 541–560.

Estrada, F., V. M. Guerrero, C. Gay-García, and B. Martínez-López (2013), A cautionary note on automated statistical downscaling methods for climate change, *Climatic change*, *120*, 263–276.

Field, C., V. Barros, T. Stocker, D. Qin, D. Dokken, K. Ebi, M. Mastrandrea, K. Mach, G.-K. Plattner, S. Allen, M. Tignor, and P. Midgley (Eds.) (2012), *Managing the Risks of Extreme Events and Disasters to Advance Climate Change Adaptation. A Special Report of Working Groups I and II of the Intergovernmental Panel on Climate Change*, 582 pp., Cambridge University Press, Cambridge, UK, and New York, NY, USA.

Frich, P., H. Alexandersson, J. Ashcroft, B. Dahlström, D. G.R, A. Drebs, v. E. A.F.V, F. E.J, I. Hanssen-Bauer, R. Heino, T. Jónsson, K. Jonasson, N. P.Ø, T. Schmidth, P. Steffensen, H. Tuomenvirta, and O. Tveito (1996), North atlantic climatological dataset (NACD version 1) - final report, *Scientific report 1*, DMI, Copenhagen, Denmark.

Førland, E. J. (), *Long-term variations in atmospheric circulation and climate in the Arctic*.

Girma, M. M., B. Schütt, S. B. Awulachew, M. Mccartney, and S. S. Demissie (2010), Downscaling rainfall in the upper blue nile basin for use in hydrological modelling, *The Global Dimensions of Change in River Basins*, p. 73.

Hov, Ø., C. Cubasch, E. Fischer, P. Höppe, T. Iversen, N. Kvamstø, Z. Kundzewicz, D. Rezacova, D. Rios, F. Santos, B. Schädler, O. Veisz, G. Zerefos, R. Benestad, J. Murlis, M. Donat, G. Leckebusch, and U. Ulbrich (2013), Extreme weather events in europe: preparing for climate change adaptation, *Tech. rep.*, MET Norway, The Norwegian Academy of Sciences and Letters (DNVA), European Academies Science Advicery Council (EASAC).

IBARRA-BERASTEGI, G., U. GANZEDO, J. SAENZ, A. EZCURRA, I. ERRASTI, A. ELIAS, A. BARONA, and L. INSAUSTI (), Linking high education and research using free software: Two experiences with r.

Imbert, A., and R. E. Benestad (2005), An improvement of analog model strategy for more reliable local climate change scenarios, *Theoretical and Applied Climatology*, *82*(DOI: 10.1007/s00704-005-0133-4), 245–255.

Klein Tank, A. J. B. W., G. P. Konnen, R. Böhm, G. Demarée, A. Gocheva, M. Mileta, S. Pashiardis, L. Hejkrlik, C. Kern-Hansen, R. Heino, P. Bessemoulin, G. Müller-Westermeier, M. Tzanakou, S. Szalai, T. Pálsdóttir, D. Fitzgerald, S. Rubin, M. Capaldo, M. Maugeri, A. Leitass, A. Bukantis, R. Aberfeld, A. F. V. v. Engelen, E. Førland, M. Mietus, F. Coelho, C. Mares, V. Razuvaev, E. Nieplova, T. Cegnar, J. A. López, B. Dahlström, A. Moberg, W. Kirchhofer, A. Ceylan, O. Pachaliuk, L. V. Alexander, and P. Petrovic (2002), Daily dataset of 20th-century surface air temperature and precipitation series for the european climate assessment, *International Journal of Climatology*, *22*, 1441–1453, data and metadata available at http://eca.knmi.nl.

Knutti, R., and J. Sedláček (2013), Robustness and uncertainties in the new CMIP5 climate model projections, *Nature Clim. Change*, *3*(4), doi:10.1038/nclimate1716.

Lorenz, E. N. (1956), Empirical orthogonal functions and statistical weather prediction, *Sci. rep. 1*, Department of Meteorology, MIT, USA.

Mezghani, A., and B. Hingray (2009), A combined downscaling-disaggregation weather generator for stochastic generation of multisite hourly weather variables over complex terrain: Development and multi-scale validation for the Upper Rhone River basin, *J. Hydrol.*, *377*(3-4), doi:10.1016/j.jhydrol.2009.08.033.

Neu, U., M. G. Akperov, N. Bellenbaum, R. Benestad, R. Blender, R. Caballero, A. Cocozza, H. F. Dacre, Y. Feng, K. Fraedrich, J. Grieger, S. Gulev, J. Hanley, T. Hewson, M. Inatsu, K. Keay, S. F. Kew, I. Kindem, G. C. Leckebusch, M. L. R. Liberato, P. Lionello, I. I. Mokhov, J. G. Pinto, C. C. Raible, M. Reale, I. Rudeva, M. Schuster, I. Simmonds, M. Sinclair, M. Sprenger, N. D. Tilinina, I. F. Trigo, S. Ulbrich, U. Ulbrich, X. L. Wang, and H. Wernli (2012), IMILAST: A Community Effort to Intercompare Extratropical Cyclone Detection and Tracking Algorithms, *Bull. Amer. Meteor. Soc.*, *94*(4), 529–547, doi:10.1175/BAMS-D-11-00154.1.

Pebesma, E., D. Nüst, Bivand, and R. (2012), The r software environment in reproducible geoscientific research, *Eos*, *93*(16), 163–164.

Peterson, T., and R. S. Vose (1997), An overview of the global historical climatology network temperature data base, *Bull. Amer. Meteor. Soc.*, *78*(doi:10.1175/1520-0477), 2837–2849.

Pocernich, M. (2003), R's role in the climate change debate, *The Newsletter of the R Project Volume 6/4, October 2006*, *14*(6), 17.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1989a), *Numerical Recipes in Pascal*, Cambridge University Press, Cambridge, UK.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1989b), *Numerical Recipes in Pascal*, Cambridge University Press, Cambridge, UK.

R Core Team (2014), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Schöner, W., and E. D. S. Cardoso (2005), reclip: more.

Semenov, M., and E. M. Barrow (1997), Use of a stochastic weather generator in the development of climate change scenarios, *Climate Change*, *35*, 397–414.

Semenov, M., and R. J. Brooks (1999), Spatial interpolation of the LARS-WG stochastic weather generator in great britain., *Climate Research*, *11*, 137–148.

Soltani, A., and G. Hoogenboom (2003), Minimum data requirements for parameter estimation of stochastic weather generators, *Climate Research*, *25*, 109–119.

Spak, S., T. Holloway, B. Lynn, and R. Goldberg (2007), A comparison of statistical and dynamical downscaling for surface temperature in north america, *Journal of Geophysical Research: Atmospheres (1984-2012)*, *112*(D8).

Spiegelhalter, D., M. Pearson, and I. Short (2011), Visualizing uncertainty about the future, *Science*, *333*(6048), 1393–1400.

Stocker, T., D. Qin, G.-K. Plattner, M. Tignor, S. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, and P. Midgley (2013), *Summary for Policymakers*, 1–30 pp., Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, doi:10.1017/CBO9781107415324.004.

Strang, G. (1988), *Linear Algebra and its Application*, Harcourt Brace & Company, San Diego, California, USA.

Tuomenvirta, H., A. Drebs, E. Førland, O. E. Tveito, H. Alexandersson, E. V. Laursen, and T. Jónsson (2001), Nordklim data set 1.0, *KLIMA 08/01*, met.no, P.O.Box 43 Blindern, N-0313 Oslo, Norway (www.met.no).

Tyler, N. J. C., J. M. Turi, M. A. Sundset, K. Strøm Bull, M. N. Sara, E. Reinert, N. Oskal, C. Nellemann, J. J. McCarthy, S. D. Mathiesen, and others (2007), Saami reindeer pastoralism under climate change: applying a generalized framework for vulnerability studies to a sub-arctic social-ecological system, *Global Environmental Change*, *17*(2), 191–206.

Wheelan, C. J. (2013), *Naked statistics: stripping the dread from the data*.

Wilks, D. (1992), Adapting stochastic weather generation algorithms for climate change studies, *Climatic Change*, *22*, 67–84.

Wilks, D. S. (1995), *Statistical Methods in the Atmospheric Sciences*, Academic Press, Orlando, Florida, USA.

Wilks, D. S., and R. L. Wilby (1999), The weather generation game: a review of stochastic weather models, *Progress in Physical Geography*, *23*(3), 329–357, doi:10.1177/030913339902300302.

Winkler, J. A., G. S. Guentchev, M. Liszewska, P.-N. Tan, and others (2011), Climate scenario development and applications for local/regional climate change impact assessments: An overview for the non-climate scientist, *Geography Compass*, *5*(6), 301–328.

Index

'as.field', 5
'as.trajectory', 5
'spell', 21
'wetfreq', 21

aggregate, 21, 29
aggregate.area, 29
alt, 21
annual, 9, 20, 31
anomaly, 28
arguments, 21
as, 33, 34
as.station, 5
attributes, 9
auto-correlation, 48, 58

browser(), 59

C.C.eq, 22
canonical correlation analysis, 36
CCA, 35, 36, 39, 51
cca, 13
CCI, 41
central limit theorem, 45, 48, 55
CHASE-PL, 3
CIM, 2
class, 9
clim.pact, 2, 36, 46
climate, 48
climate change, 48
climate services, 1
climatology, 28, 45
climvar, 18, 19
CMIP5, 51
cntr, 21
coherence, 36
combine, 28, 40, 47, 57
common EOFs, 40, 45, 56, 57
common information model, 5
count, 21, 22, 53
count.trajectory, 42
covariate, 46
criticism, 47
cross-validation, 48, 57
crossval, 57
cumugram, 18, 19

data, 11
data structure syntax, 5
day, 9
de-trend, 48
de-trending, 55
dependencies, 36
diagnose, 13, 56–58
diagnose.ds, 48
diagram, 18, 19
distill, 18
downscale, 55–57
downscaling, 36, 45, 47, 48, 53
DRS, 2
DS, 40, 48, 49, 52
ds, 13, 36
DSensemble, 47, 48
dsensemble, 13

ECA&D, 6, 8
empirical orthogonal function, 35
empirical orthogonal functions, 35
ensembles, 47
EOF, 35, 37, 39, 51, 57
eof, 13
eof2field, 35
EOFs, 47
EOFs, mixed, 46
exceedance, 21, 22

field, 9, 13
filt, 36
filter, 35, 36
flow field, 14

g2dl, 21
GCMs, 25, 55
GHCN, 6, 8, 10
github, 62
global mean temperature, 30
gridded data, 5
gridding, 35

history, 21
Huth's dilemma, 48, 55

iid, 55
iid.test, 48, 55
IMILAST, 40
imilast.M03, 40, 43, 52
indexing, 25
India, 10
INDICE, 3
infographics, 18
is, 21
it, 21

lat, 21
linear models, 46
loc, 21
lon, 21

map, 14, 15, 28, 40
map.density.trajectory, 41
matchdate, 28
meta data, 7
missing data, 59
month, 9
MVR, 36
mvr, 13

NACD, 8, 38
NARP, 8
NCEP, 16
NE, 22
nearest, 26
NetCDF, 6
non-stationarity, 55
Nordklim, 8
nv, 21, 22

parameters, 53
PC, 35
PCA, 35, 38–40, 45, 47, 50, 51
pca, 13
PCA.trajectory, 44
plot, 5, 14, 15, 40, 42
precip.Pr, 22
precip.rv, 22, 24
precip.vul, 22, 24
predict, 36
predictand, 47
predictands, 45
predictor, 46
predictord, 45
principal component analysis, 35
probabilistic model, 47
probabilities, 53
probability distribution, 53
project, 36
pseudo-reality, 55

RCMs, 25
regression analysis, 36, 45
regrid, 25, 27
residuals, 57, 58
retrieve, 5, 6
rotated grid, 6

S3-method, 9
sample size, 45
seasonal, 9
select.station, 8
singular, 60
singular spectrum analysis, 36
singular value decomposition, 35
sp2np, 21
spatial average, 29
spell, 13, 22, 23
SSA, 36

standard terminology, 9
station, 5, 6, 9, 13
stationarity, 47
stid, 21
str, 11
sub-daily, 9
sub-samples, 21
subset, 26–28
summary, 10, 12
SVD, 35, 60

t2m.Pr, 22
t2m.vul, 22
tele-connections, 36
time scale, 9
trajectories, 5
trajectory, 13, 40, 48
trend, 18, 58
trouble shooting, 59
try, 60

validate, 61
varid, 21
vec, 17
vector plots, 14
verbose, 59
vis, 18
vis.trends, 18, 20
visualisation, 18
vulnurability, 24

weather generator, 53
wetfreq, 22, 23
wetmean, 21–23
WG.FT.day.t2m, 54
WG.fw.day.precip, 54
wheel, 18, 19

xsection, 13
xval, 13

zoo, 9