The ECOMS User Data Gateway: Towards seasonal prediction data provision and research reproducibility in the era of Climate Services

Companion R examples of the paper published in Climate Services (2018, DOI:10.1016/j.cliser.2017.07.001)

Antonio Cofiño, Joaquín Bedia, Maialen Iturbide, Manuel Vega, Sixto Herrera, Jesús Fernández, M. Dolores Frías, Rodrigo Manzanas & Jose Manuel Gutiérrez

18/07/2017

Abstract

In this document we present the code that reproduces the analyses presented in the paper. It mostly builds on the R packages of the *climate4R* bundle, plus some additional packages for forecast verification that have been conveniently bridged for a seamless integration. For the sake of brevity and accesibility, we only use the UDG public datasets, whose access is automatically granted to anyone registering Thus, we illustrate the analyses using the CFSv2 seasonal hindcast (CFSv2, Saha et al 2013), the NCEP/NCAR reanalysis1 (NCEP, Kalnay et al 1996) and the gridded observations from the EOBSv14 dataset (EOBS, Klein Tank et al 2002). Furthermore, post-processed datasets are available for direct loading to facilitate the reproducibility of figures. in the UDG via the THREDDS administration Panel (TAP).

Contents

R PACKAGE REQUIREMENTS

The following R packages (R Core Team 2017) are required to follow the examples. Note the last **Session Information Section** of this manual, where the specific package versions used to build these examples are indicated.

Installing the adequate version

All the *climate4R* packages are currently under active development. Thus, to ensure the reproducibility of these examples, it is recommend to install the same versions of the packages that produced this document, even though in most cases more recent versions exist. The package versions used are indicated in the Session Information Section at the end of this document. The function `install_github` from package `devtools` (Wickham and Chang 2016) is recommended to install a particular package version from the GitHub repositories. For, instance, to install `transformeR` v0.0.14, it suffices with pointing to the specific version tag after the `@` symbol.

Thus, in order to install the proper versions of the packages of the *climate4R* bundle needed to run these examples:

```
devtools::install_github(c("SantanderMetGroup/transformeR@v0.0.14",
                           "SantanderMetGroup/loadeR@v1.0.9",
                           "SantanderMetGroup/loadeR.ECOMS@v1.2.3")
```

R packages for data loading

The R package `loadeR.ECOMS` will perform the data loading task, including authentication against the UDG server (see the installation instructions). In addition, a quick overview of all datasets and variables available in the ECOMS-UDG is available in this table

The `loadeR.ECOMS` package extends the capabilities of the `loadeR` package for data access, and enables access to any remote dataset via OPeNDAP (more details in the loadeR's wiki page), as well as creating and accesing datasets locally. An example of remote access to a public dataset outside the ECOMS-UDG is provided in Section 4.

```
library(loadeR.ECOMS)
```

R package for data post-processing and plotting

In addition, the R package `transformeR` (Bedia and Iturbide 2017) enables climate data transformation (plotting, aggregation, subsetting, PCA/EOF analysis ...). Further details on its capabilities and installation instructions are available in the wiki page. It is seamlessly integrated with the data structures provided by `loadeR.ECOMS`.

```
library(transformeR)
```

R packages for verification

Two packages for verification have been developed in the frame of ECOMS: `SpecsVerification` (Siegert 2017), in SPECS, and `easyVerification` (MeteoSwiss 2017), in EUPORIAS. Both packages are available

on CRAN. The former is a dependency of the latter, so installing `easyVerification` will also install `SpecsVerification` if not already present:

```
if (!require("easyVerification")) install.packages("easyVerification")
```

```
library(easyVerification)
```

Additional packages

The package `RColorBrewer` is used to replicate the spectral color palette used in the paper in the correlation maps. Next, the palette `veri.colors` is defined, that will be used in the verification maps:

```
library(RColorBrewer)
cols <- brewer.pal(n = 11, name = "Spectral")
veri.colors <- colorRampPalette(rev(cols))
```

ANALYSIS OF THE WINTER NAO INDEX

Data Loading

NOTE: the already post-processed datasets used in this section are available for direct download (skip this section and jump to Section 2.2 if directly loaded with the next lines)

```
## NCEP reanalysis SLP
url <- "http://meteo.unican.es/work/UDG/NCEP_psl_DJF_annual_1949_2010_LD.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
## CFSv2 SLP
url <- "http://meteo.unican.es/work/UDG/CFS_24_lm1_psl_DJF_annual_1983_2010_LD.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
```

Prior to data access, authentication is required to access the UDG. The authentication is performed in one step:

```
loginUDG(username = "jDoe", password = "*****")
```

NCEP Reanalysis1 data

The spatial domain to compute the PC-based NAO index is indicated in NCAR's Climate Data Guide

```
lonLim = c(-90,40)
latLim = c(20,80)
var = "psl"
```

Note that `"psl"` is the standard name defined in the UDG vocabulary for sea-level pressure. The UDG vocabulary can be inspected by typing `UDG.vocabulary()`. The sea-level pressure data from the NCEP reanalysis (`dataset = "NCEP_reanalysis1"`) is loaded in order to reconstruct the observed winter NAO

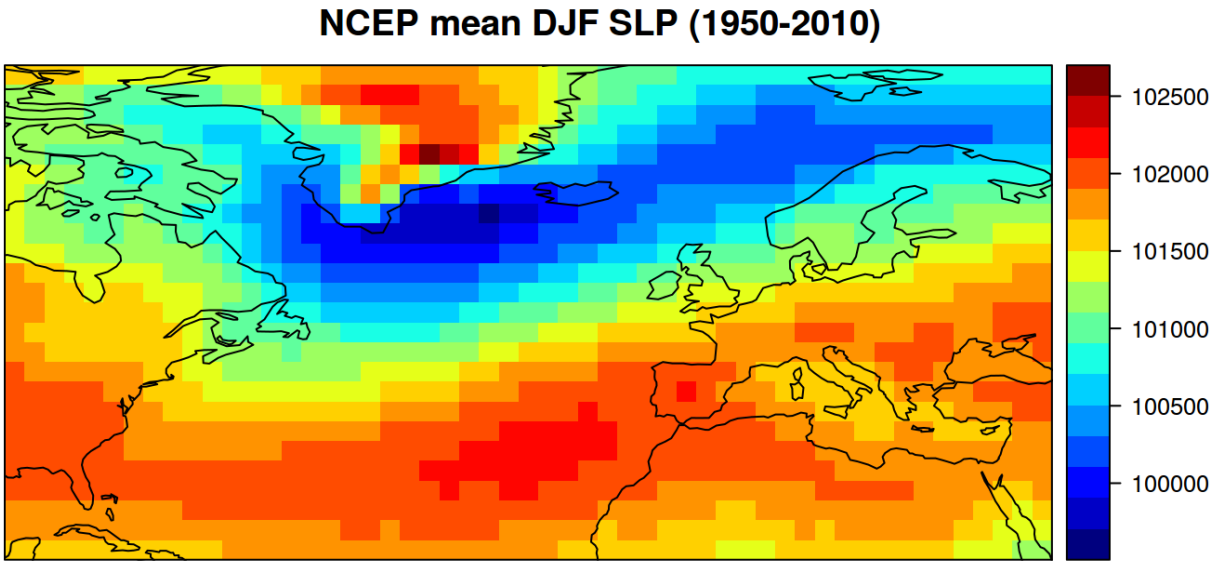
index (`season = c(12,1,2)`) for the entire reanalysis period (1950-2010) (`years = NULL`, the default, which retrieves all available years. This is equivalent to `years = 1950:2010` in this case). As the original data are 6-hourly, data are aggregated on-the-fly by `loadeR.ECOMS` by introducing the arguments `time = "DD"` (to conver the data from 6-h to daily), `aggr.d = "mean"` (to indicate the aggregation function) and `aggr.m = "mean"` (which indicates that daily data will be monthly averaged).

```
ncep.psl <- loadECOMS(dataset = "NCEP_reanalysis1",
                      var = var,
                      lonLim = lonLim,
                      latLim = latLim,
                      season = c(12,1,2),
                      years = NULL,
                      time = "DD",
                      aggr.d = "mean",
                      aggr.m = "mean")
```

Next we represent the mean sea-level pressure winter climatology:

```
plotClimatology(climatology(ncep.psl),
                backdrop.theme = "coastline",
                main = "NCEP mean DJF SLP (1950-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



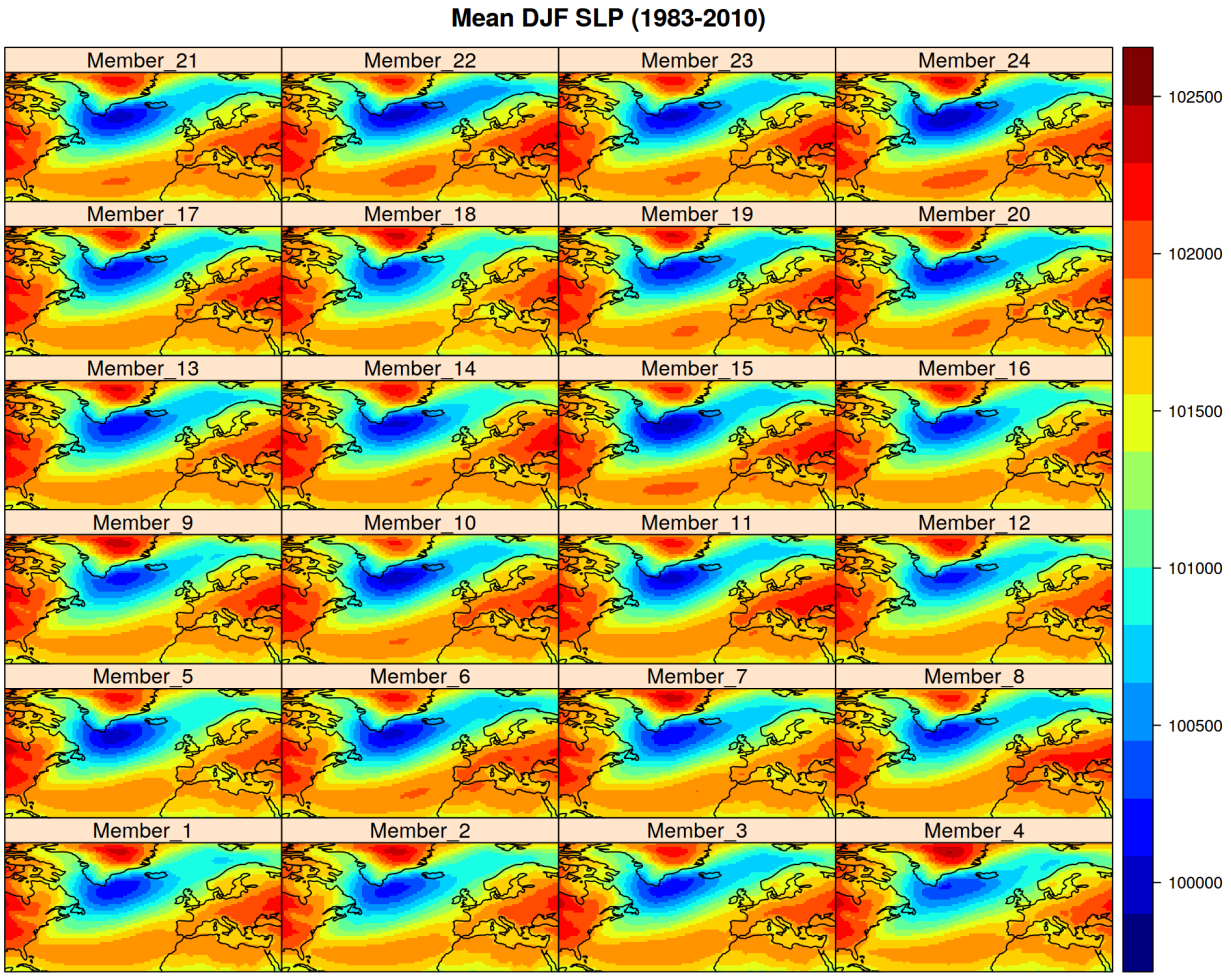
CFSv2 seasonal hindcast data

The corresponding hindcast is loaded in a similar way, but taking into account that in this case, it is necessary to specify two additional parameters to unequivocally define the query: the members (for instance, the first 24 members: `members = 1:24`) and the initialization time (e.g., 1-month ahead predictions: `leadMonth = 1`). Here, we illustrate this step using the data from the CFSv2 hindcast (`dataset = "CFSv2_seasonal"`). Note that the only difference between this example call to `loadECOMS` and a request for the other two hindcasts compared in the paper, would lie just in the dataset argument: `dataset = "Glosea5_seasonal_24"` for the UKMO Glosea5 hindcast of 24 members, or `dataset = "System4_seasonal_51"` for the ECMWF System4 seasonal hindcast of 51 members. Thanks to the data harmonization performed by `loadeR.ECOMS`, the rest of parameters (variable naming...) remain exactly the same. Member selection is also transparent to users, who don't need to worry about the different member configurations. As an example, see the CFSv2 lagged runtime member configuration and the definition of members undertaken via `loadECOMS`.

```
cfs.psl <- loadECOMS(dataset = "CFSv2_seasonal",
                     var = var,
                     lonLim = lonLim,
                     latLim = latLim,
                     season = c(12,1,2),
                     years = NULL,
                     time = "DD",
                     aggr.d = "mean",
                     aggr.m = "mean",
                     members = 1:24,
                     leadMonth = 1)
```

```
plotClimatology(climatology(cfs.psl),
                backdrop.theme = "coastline",
                main = "Mean DJF SLP (1983-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



Temporal aggregation

The monthly data are next annually aggregated:

```
ncep.psl <- aggregateGrid(grid = ncep.psl, aggr.y = list(FUN = "mean"))
cfs.psl <- aggregateGrid(grid = cfs.psl, aggr.y = list(FUN = "mean"))
```

NAO INDEX

The PC-based NAO index is defined as the leading EOF of the sea-level pressure anomalies. The methodological details and reference time-series are presented in the NCAR's Climate Data Guide.

Observed NAO Index

After data loading, the NAO-Index can be calculated in two simple steps using the transformation tools available in `transformeR`:

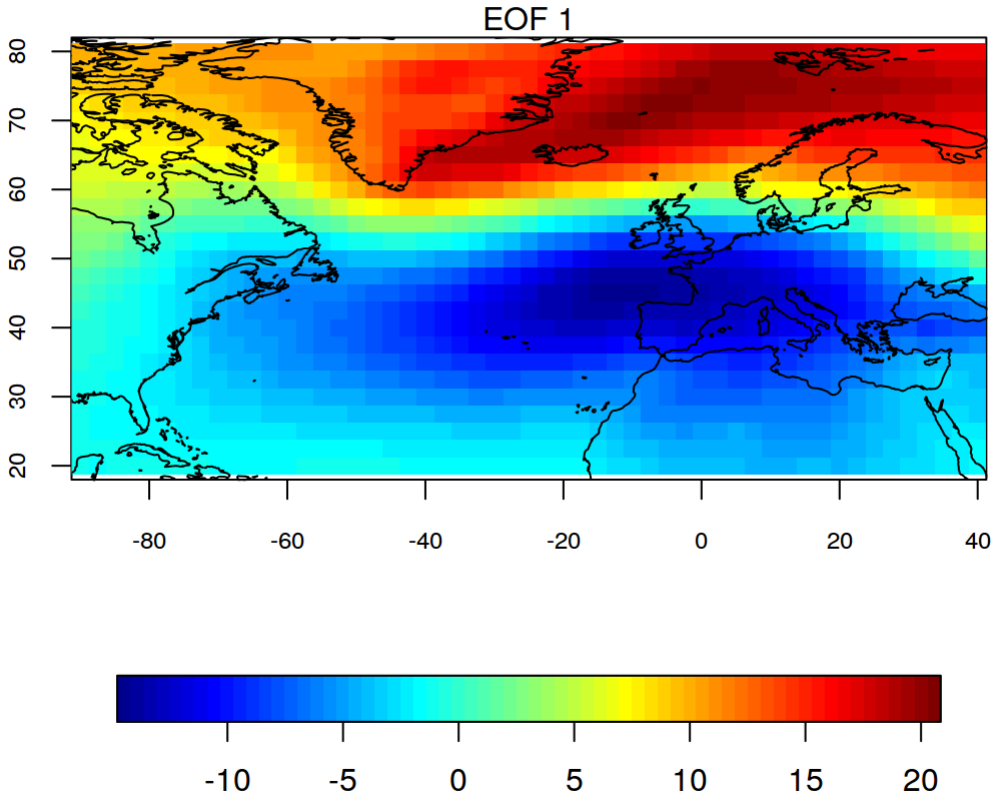
1. The anomalies are computed using `rescaleGrid` with its default arguments:

```
psl.anom <- rescaleGrid(grid = ncep.psl)
```

2. The EOFs (in this case only the first one) are computed using the `prinComp` function, and then plotted with `plotEOF`:

```
pca1 <- prinComp(psl.anom, n.eofs = 1)
plotEOF(pca1, var = "psl", n.eofs = 1)
```

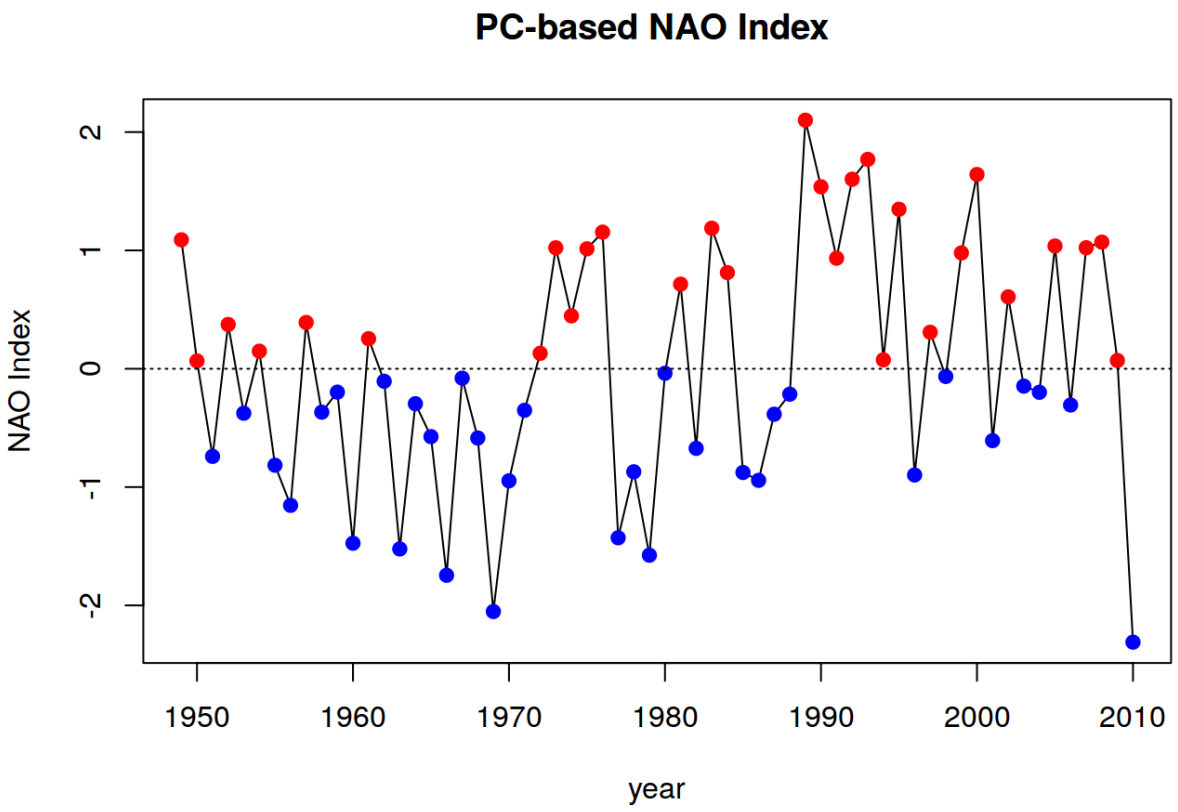
```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



The next lines of code extract the leading PC as a time-series grid, and plots it as a NAO-index time series (see this link for a visual comparison with Hurrel's PC-based NAO-Index).

```
nao <- PC2grid(pca1, scale = TRUE, opp = TRUE)
nao.index.ncep <- nao[["Data"]][1,,1,1]
years <- getYearsAsINDEX(ncep.psl)
plot(years, nao.index.ncep, ty = 'l', ylab = "NAO Index", xlab = "year")
pos <- which(nao.index.ncep > 0) ## Index of positive NAO years
```

```
neg <- setdiff(1:length(nao.index.ncep), pos) ## Index of negative NAO years
points(years[pos], nao.index.ncep[pos], pch = 19, col = "red")
points(years[neg], nao.index.ncep[neg], pch = 19, col = "blue")
abline(h = 0, lty = 3)
title(main = "PC-based NAO Index ")
```



Hindcast NAO Index

First, the CFSv2 data are interpolated from the original model grid to the NCEP grid:

```
psl.cfs.interp <- interpGrid(grid = cfs.psl, new.coordinates = getGrid(ncep.psl))
```

Next, the same steps than in Sec. 1.2.1. are followed:

1. Anomaly calculation

```
psl.cfs.anom <- rescaleGrid(psl.cfs.interp)
```

2. PC calculation. In this case, the PCs are obtained using the NCEP EOF previously calculated, using the function `grid2PCs` as follows:

```
cfs.pc.list <- grid2PCs(pca1, grid = psl.cfs.anom)
```

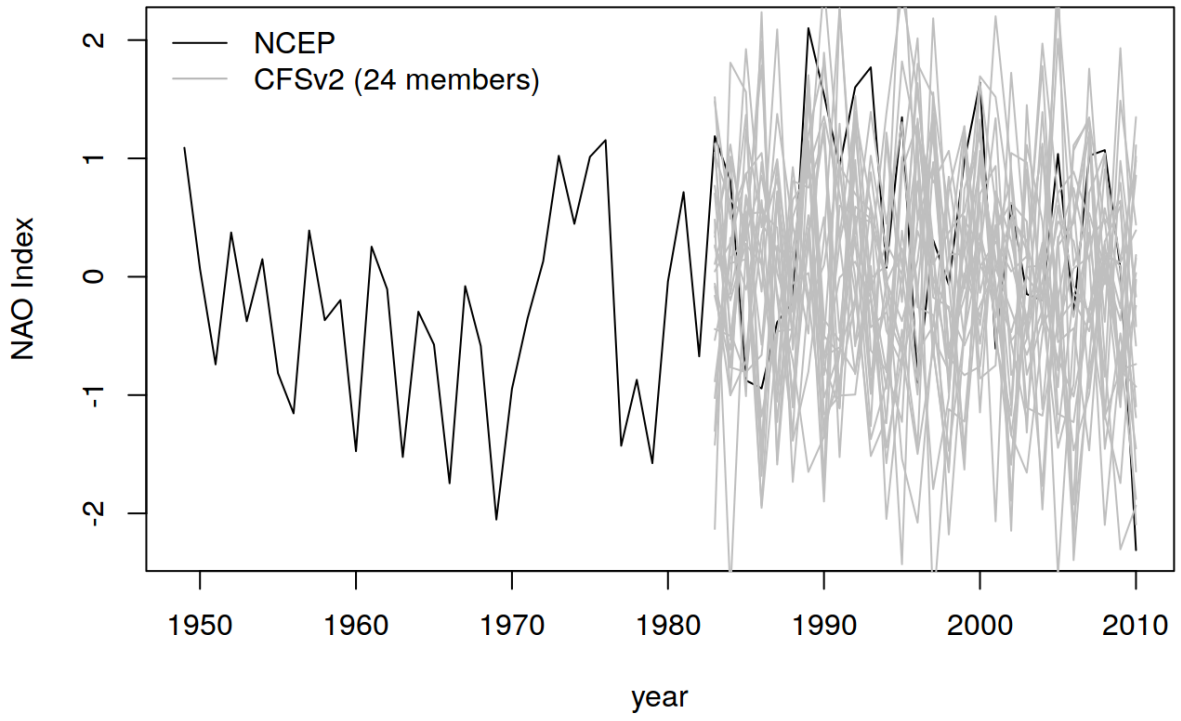
The output is a list with the first PC for each of the 24 members. It is then converted into a `matrix` (members in columns, years in rows), which is also scaled to get the final NAO-Index:

```
nao.index.cfs <- sapply(cfs.pc.list, function(x) scale(x)*-1)
```

```
plot(years, nao.index.ncep, ty = 'l', ylab = "NAO Index", xlab = "year")
years.hind <- getYearsAsINDEX(cfs.psl)
apply(nao.index.cfs, MARGIN = 2, FUN = function(x) lines(years.hind, x, col = "grey75"))
```

```
## NULL
```

```
legend("topleft", c("NCEP","CFSv2 (24 members)"), lty = 1, col = c(1, "grey70"), bty = "n")
```



Correlation analysis

For simplicity, only the NCEP years overlapping the hindcast time period (1983-2010) are retained:

```
ref <- intersect(years, years.hind)
nao.index.ncep <- nao.index.ncep[years %in% ref]
```

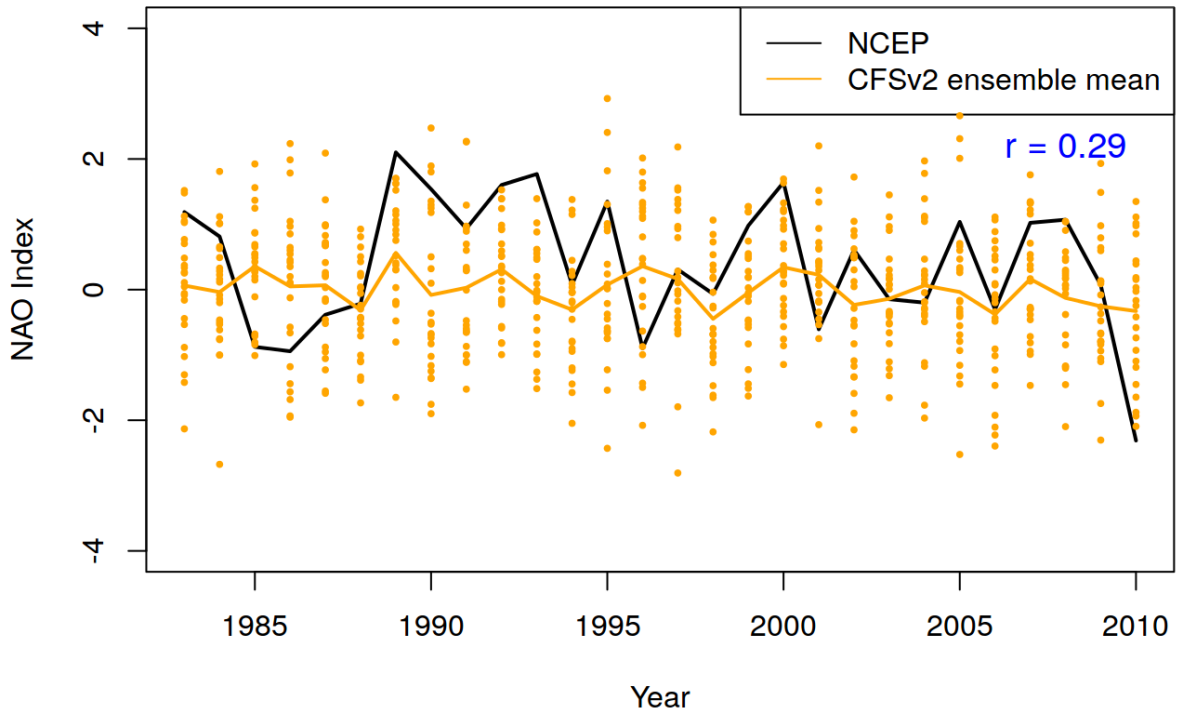
Next, the correlation between the ensemble mean and NCEP is calculated:

```
cor(rowMeans(nao.index.cfs), nao.index.ncep)
```

```
## [1] 0.4124628
```

The next lines of code, plot the observed Winter NAO Index against the hindcast predictions. Both the ensemble mean (orange line) and the individual members (red dots) are displayed:

```
plot(ref, nao.index.ncep, ty = 'l', ylab = "NAO Index", xlab = "Year", ylim = c(-4,4), lwd = 2)
lines(ref, rowMeans(nao.index.cfs), col = "orange", lwd = 2)
legend("topright", c("NCEP","CFSv2 ensemble mean"), lty = 1, col = c(1,"orange"))
for (i in 1:length(ref)) {
    points(rep(ref[i], ncol(nao.index.cfs)), nao.index.cfs[i,], cex = .4, pch = 19, col = "orange")
}
text(2008, 2.2, cex = 1.2, col = "blue", bty = "n", paste("r =", round(cor(rowMeans(nao.index.cfs), nao
```



Bootstrap analysis

For convenience, we next create a matrix (`nao.mat`) containing the NCEP NAO-Index in its first column, and the 24 NAO-Index members from CFSv2 in the remaining columns.

```
nao.mat <- cbind(nao.index.ncep, nao.index.cfs)
```

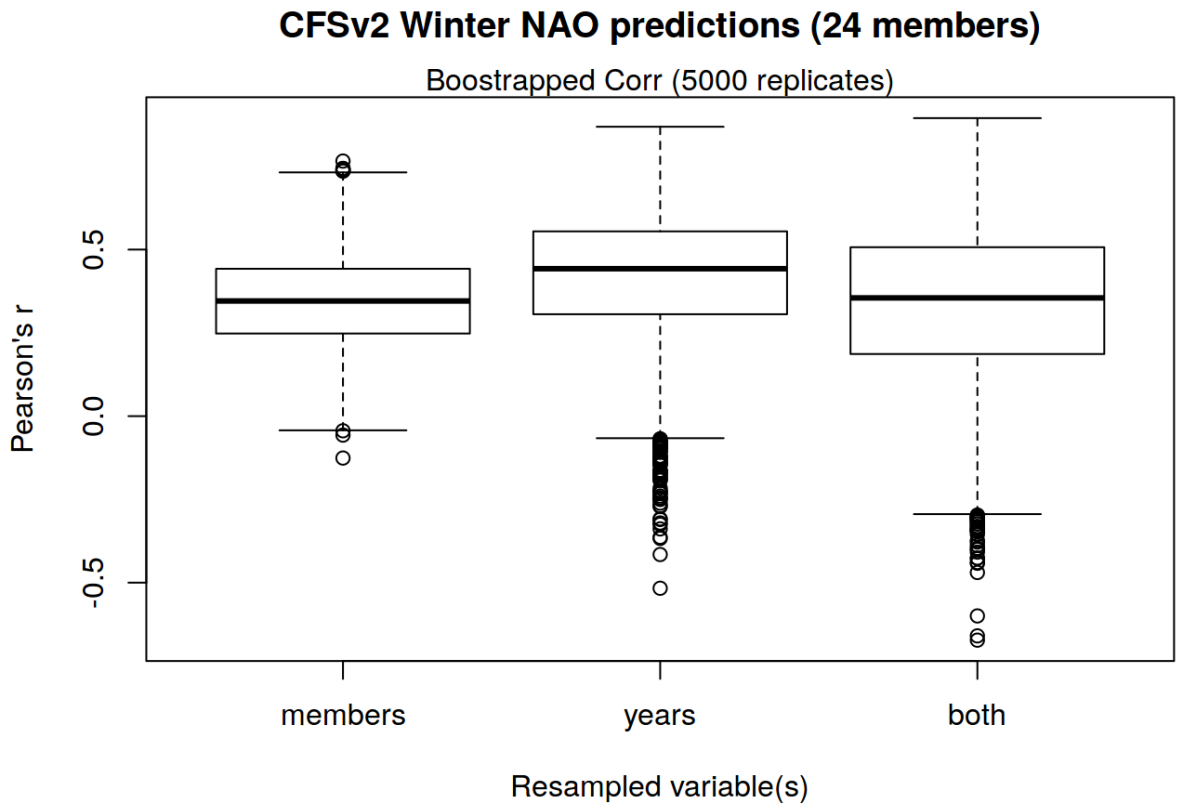
The *ad-hoc* function `boot.corr` is envisaged to perform the bootstrapping either on members, on years or on both variables simultaneously, as indicated by the argument `what`:

```
boot.corr <- function(df, R = 1000, what = c("members", "years")) {
    df <- df[complete.cases(df), ]
    what <- match.arg(what, c("members", "years"), several.ok = TRUE)
    mem <- 1:ncol(df)
    yrs <- 1:nrow(df)
    out <- rep(NA, R)
```

```
    iter <- 1
    ind.col <- 1:length(mem)
    ind.row <- 1:length(yrs)
    while (iter <= R) {
        if ("members" %in% what) ind.col <- sample(mem, size = length(mem), replace = TRUE)
        if ("years" %in% what) ind.row <- sample(yrs, size = length(yrs), replace = TRUE)
        out[iter] <- cor(rowMeans(df[ind.row,ind.col]), df[ind.row,1])
        iter <- iter + 1
    }
    return(out)
}
```

The boostrapped correlations are next computed, and the results summarized in a boxplot, similar to Fig. 4 in the paper:

```
R = 5000 # Number of replicates
what.list <- list("members", "years", c("members","years"))
res <- vapply(1:length(what.list), FUN.VALUE = numeric(R), FUN = function(x) {
    boot.corr(nao.mat, R, what = what.list[[x]])
})
boxplot(res, names = c("members", "years", "both"), ylab = "Pearson's r", xlab = "Resampled variable(s)"
title("CFSv2 Winter NAO predictions (24 members)")
mtext(paste0("Boostrapped Corr (", R, " replicates)"))
```



VERIFICATION OF MEAN WINTER TEMPERATURE FORECASTS IN THE NORTH-ATLANTIC DOMAIN

NOTE: the already post-processed datasets used in this section are available for direct download (skip this section and jump to Section 3.2 if loaded through the next lines of code)

```
## NCEP
url <- "http://meteo.unican.es/work/UDG/NCEP_tas_DJF_annual_1983_2010_LD.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
## CFSv2
url <- "http://meteo.unican.es/work/UDG/CFS_24_lm1_tas_DJF_annual_1983_2010_LD.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
```

The geographical window used for the NAO Index calculation is defined:

```
lonLim = c(-90,40)
latLim = c(20,80)
```

Data loading

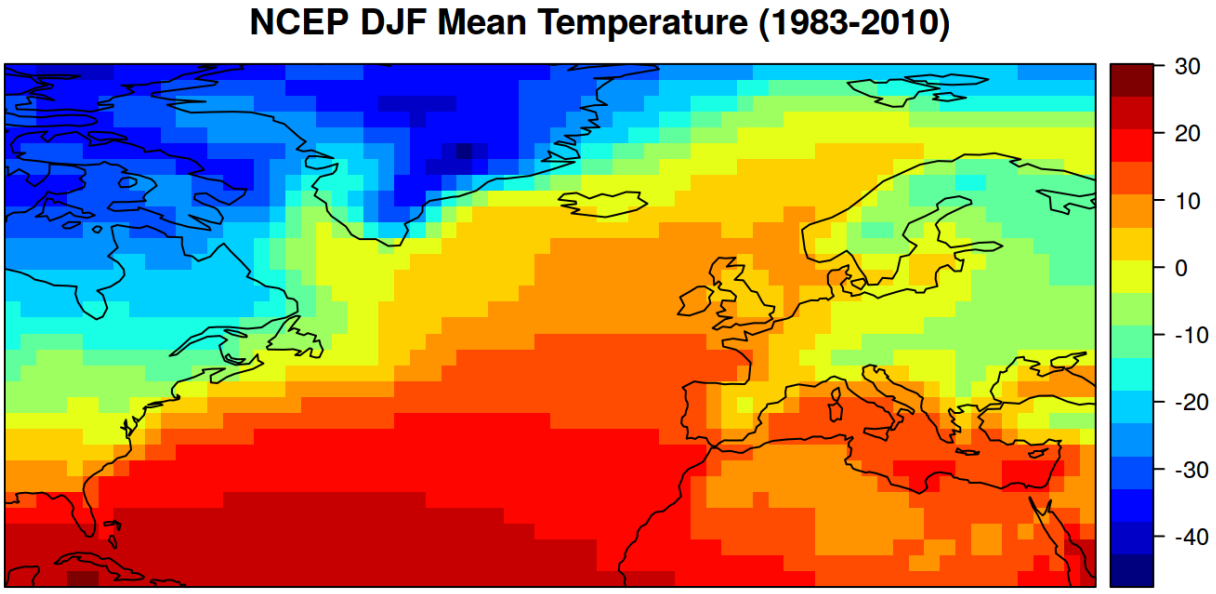
NCEP reanalysis

```
tas.ncep <- loadECOMS(dataset = "NCEP_reanalysis1",
                      var = "tas",
                      lonLim = lonLim,
                      latLim = latLim,
                      season = c(12,1,2),
                      years = NULL,
                      time = "DD",
                      aggr.d = "mean",
                      aggr.m = "mean")
```

Next we represent the mean temperature winter climatology:

```
plotClimatology(climatology(tas.ncep), backdrop.theme = "coastline", main = "NCEP DJF Mean Temperature
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



CFSv2 seasonal hindcast

Next, the CFSv2 temperature hindcast predictions are loaded. Note that the arguments passed to `loadECOMS` are the same as in the previous call, but changing the value of `var` to `tas`, which is the harmonized name for near-surface air temperature:

```
tas.cfs <- loadECOMS(dataset = "CFSv2_seasonal",
                     var = "tas",
                     lonLim = lonLim,
                     latLim = latLim,
                     season = c(12,1,2),
                     years = NULL,
                     time = "DD",
                     aggr.d = "mean",
                     aggr.m = "mean",
                     members = 1:24,
                     leadMonth = 1)
```

Temporal aggregation and interpolation

Monthly data are annually aggregated:

```
tas.ncep <- aggregateGrid(grid = tas.ncep, aggr.y = list(FUN = "mean"))
tas.cfs <- aggregateGrid(grid = tas.cfs, aggr.y = list(FUN = "mean"))
```

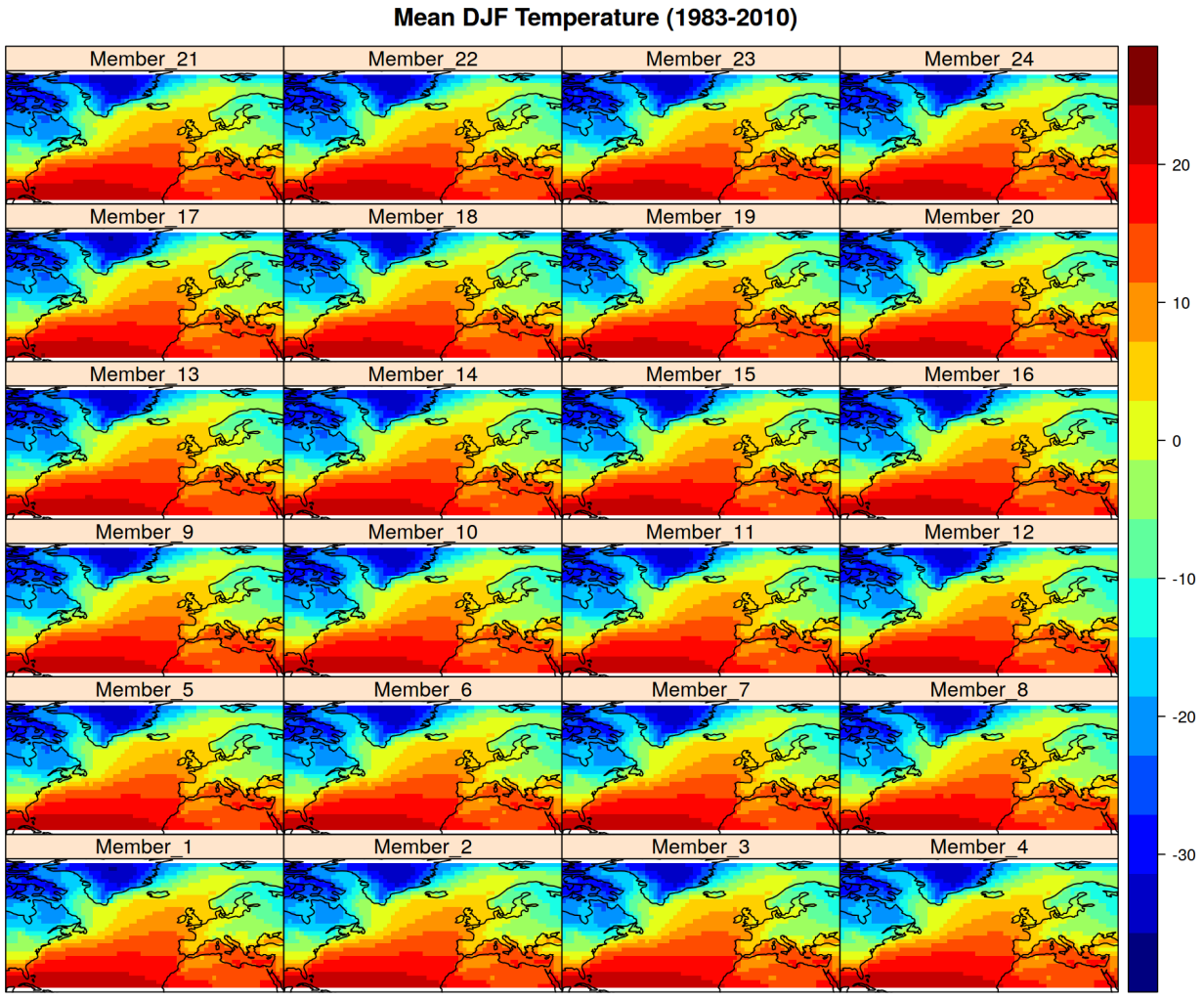
Furthermore, the CFSv2 data are interpolated to match the NCEP reanalysis grid:

```
tas.cfs <- interpGrid(tas.cfs, new.coordinates = getGrid(tas.ncep), method = "bilinear")
```

The resulting hindcast data are next plotted:

```
plotClimatology(climatology(tas.cfs), backdrop.theme = "coastline", main = "Mean DJF Temperature (1983-
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



Correlation analysis

```
obs <- tas.ncep[["Data"]]
fcst <- tas.cfs[["Data"]]
cor <- veriApply(verifun = "EnsCorr", fcst = fcst, obs = obs, ensdim = 1, tdim = 2)
```

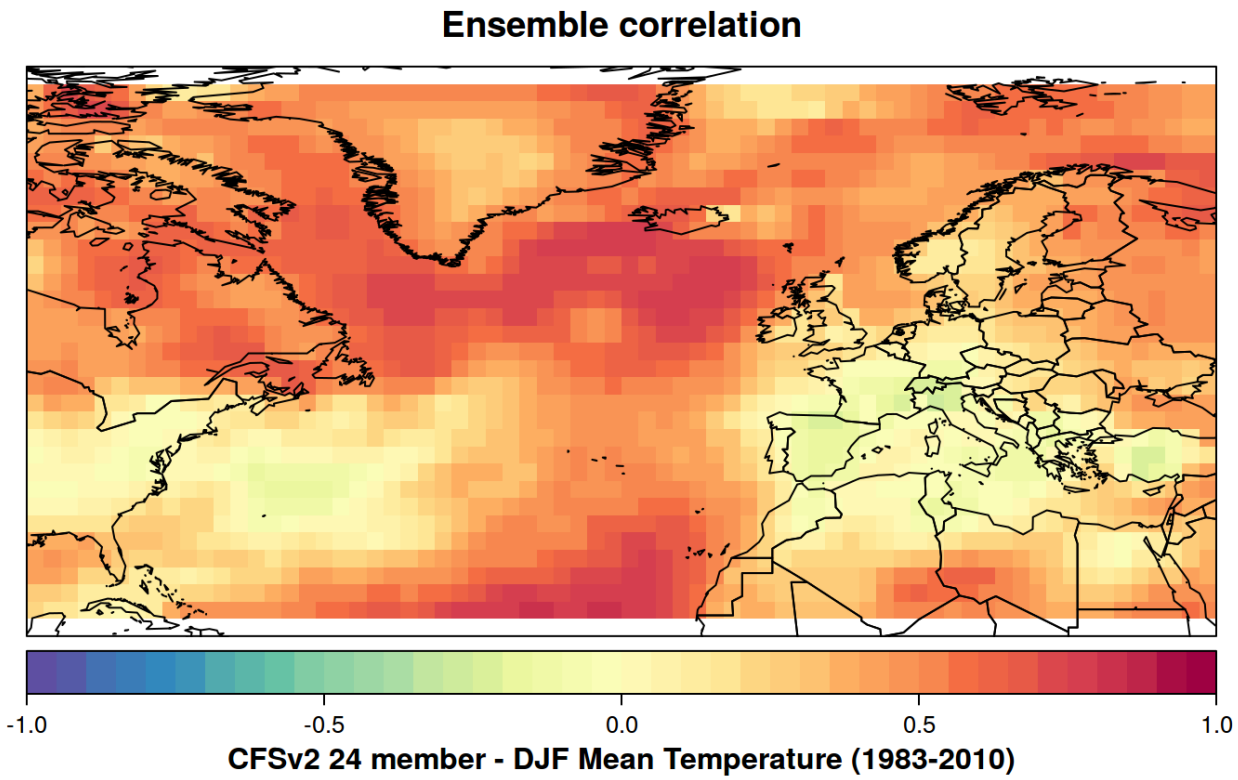
The object `corr` contains the correlation coefficients for each grid point. The bridging function `easyVeri2grid` from package `transformeR` performs the transformation from the raw verification matrices produced by `easyVerification` to the grid structure with all the required metadata for data collocation and representation handled by the `climate4R` bundle packages (including `transformeR`):

```
cor.grid <- easyVeri2grid(easyVeri.mat = cor, obs.grid = tas.ncep, verifun = "EnsCorr")
```

It is now straightforward the representation of the correlation map, that is treated as a climatology (i.e., the time dimension is a singleton). Additional arguments are needed in order to mimic the spectral color palette used in the paper (previously defined in Sec. 1.4). Extremely flexible plotting options are available, since `plotClimatology` is a wrapper for the powerful `spplot` method for spatial representation of geospatial `sp` objects in R, thus accepting all its arguments.

```
plotClimatology(climatology(cor.grid),
                backdrop.theme = "countries",
                col.regions = veri.colors(51), ## Define color palette
                at = seq(-1,1,.05), ## Set range of values
                colorkey = list(space = "bottom"), ## Place legend at bottom
                main = "Ensemble correlation",
                sub = "CFSv2 24 member - DJF Mean Temperature (1983-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



VERIFICATION OF MEAN WINTER TEMPERATURE OVER WESTERN EUROPE

NOTE: the already post-processed datasets used in this section are available for direct download (skipt this section and jump to Section 4.2 if loaded this way):

```
## EOBS mean temperature
url <- "http://meteo.unican.es/work/UDG/EOBS_tas_DJF_annual_1983_2010.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
## CFSv2
url <- "http://meteo.unican.es/work/UDG/CFS_24_tas_DJF_annual_1983_2010.Rdata"
temp_file <- tempfile()
download.file(url, destfile = temp_file)
load(temp_file, .GlobalEnv, verbose = TRUE)
```

Data loading

CFSv2 temperature

A spatial window centered in western Europe is next defined. The harmonized name for near-surface temperature is `"tas"`, being its units degrees Celsius (see `UDG.vocabulary()`):

```
lonLim = c(-10,30)
latLim = c(35,70)
```

```
tas.cfs <- loadECOMS(dataset = "CFSv2_seasonal",
                     var = "tas",
                     lonLim = lonLim,
                     latLim = latLim,
                     members = 1:24,
                     years = 1983:2010,
                     leadMonth = 1,
                     season = c(12,1,2),
                     time = "DD",
                     aggr.d = "mean",
                     aggr.m = "mean")
```

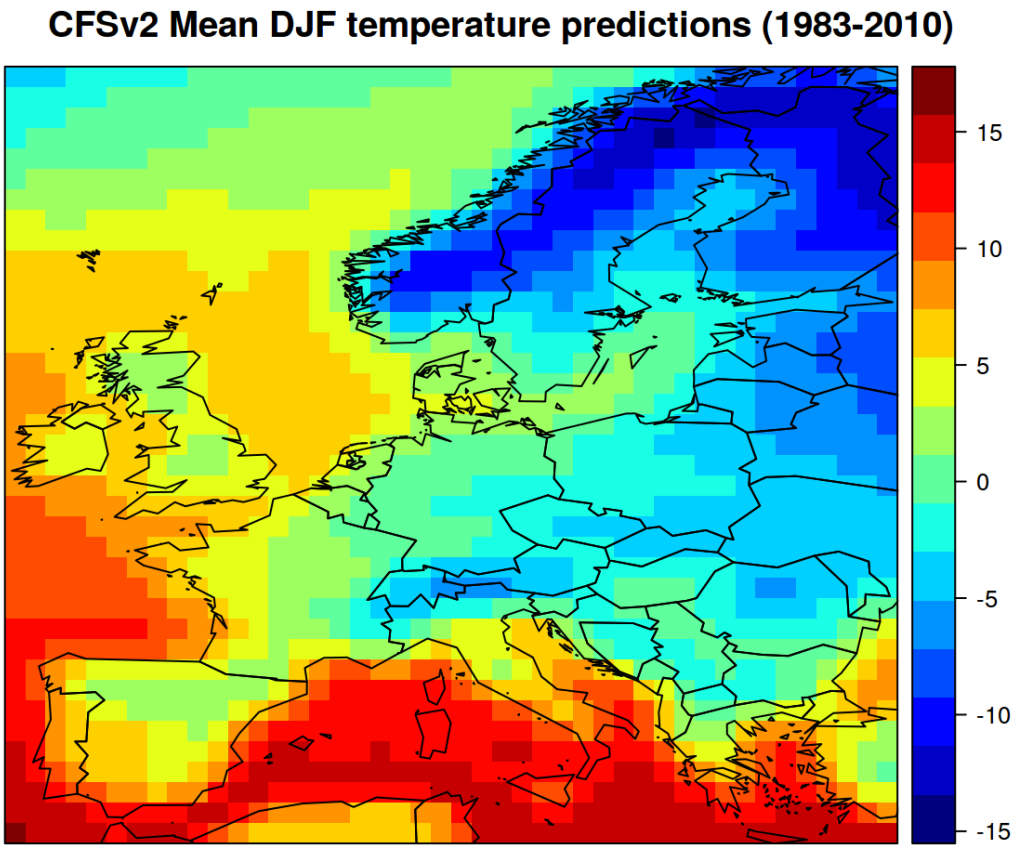
Data are next annually aggregated:

```
tas.cfs <- aggregateGrid(tas.cfs, aggr.y = list(FUN = "mean"))
```

The ensemble mean climatology is next plotted. Note the argument `by.member = FALSE` passed to function `climatology` to this aim. The default behaviour is to compute the climatology for each member sepparately (as in Sec. 2.1.3).

```
plotClimatology(grid = climatology(tas.cfs, by.member = FALSE), backdrop.theme = "countries",
                main = "CFSv2 Mean DJF temperature predictions (1983-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



Loading remote data outside the UDG: example with EOBSv14

In this example, gridded observations from the EOBS database are directly loaded from the KNMI's OPeNDAP server. Unlike the previous loading examples with `loadECOMS`, in this case, by default, the original variable naming of the native dataset has to be used, although any dataset can be harmonized by preparing a user defined *dictionary*, able to translate the dataset names to the UDG vocabulary convention, undertaking the necessary variable transformations to match the canonical units (see `UDG.vocabulary()`). This process is further detailed with examples in the loadeR's wiki.

In order to find out the characteristics of the remote dataset prior to data loading, a preliminary inventory of available data can be undertaken. This inventory will provide all the necessary information for data collocation (temporal and spatial extent of the dataset, units, names of the variable(s) ...):

NOTE: The EOBS dataset has periodical version updates affecting the target URL. Thus, the link below may be not valid anymore after EOBS update. Thus, it is recommended to visit the E-OBS catalog prior to data loading/inventoring.

```
dataset = "http://opendap.knmi.nl/knmi/thredds/dodsC/e-obs_0.50regular/tg_0.50deg_reg_v14.0.nc"
di <- dataInventory(dataset)
str(di)
```

```
## List of 1
```

```
##  $ tg:List of 4
##   ..$ Description: chr "mean temperature"
##   ..$ DataType   : chr "float"
##   ..$ Units      : chr "Celsius"
##   ..$ Dimensions :List of 3
##   .. ..$ time:List of 4
##   .. .. ..$ Type      : chr "Time"
##   .. .. ..$ TimeStep  : chr "1.0 days"
##   .. .. ..$ Units     : chr "days since 1950-01-01 00:00"
##   .. .. ..$ Date_range: chr "1950-01-01T00:00:00Z - 2016-08-31T00:00:00Z"
##   .. ..$ lat :List of 3
##   .. .. ..$ Type  : chr "Lat"
##   .. .. ..$ Units : chr "degrees_north"
##   .. .. ..$ Values: num [1:101] 25.2 25.8 26.2 26.8 27.2 ...
##   .. ..$ lon :List of 3
##   .. .. ..$ Type  : chr "Lon"
##   .. .. ..$ Units : chr "degrees_east"
##   .. .. ..$ Values: num [1:232] -40.2 -39.8 -39.2 -38.8 -38.2 ...
```

Data loading uses the same arguments as the previous call to `loadECOMS`, but the generic `loadGridData` from package `loadeR` is used instead, that can be also used to load locally stored datasets. Note that `lonLim` and `latLim` arguments are now omitted, meaning that the whole spatial extent of the dataset (Europe in this case) will be loaded. Furthermore, the inventori reveals that the name of the variable, described as "mean temperature", is "tg" (`var = "tg"`). Thus, no data harmonization is being undertaken in this case (`dictionary = FALSE`). Also, the argument `years` is set to the period 1983-2010, to match the CFSv2 hindcast period already loaded:

```
eobs.tas <- loadGridData(dataset = dataset,
                         var = "tg",
                         years = 1983:2010,
                         season = c(12,1,2),
                         time = "DD",
                         aggr.d = "mean",
                         aggr.m = "mean",
                         dictionary = FALSE)
```

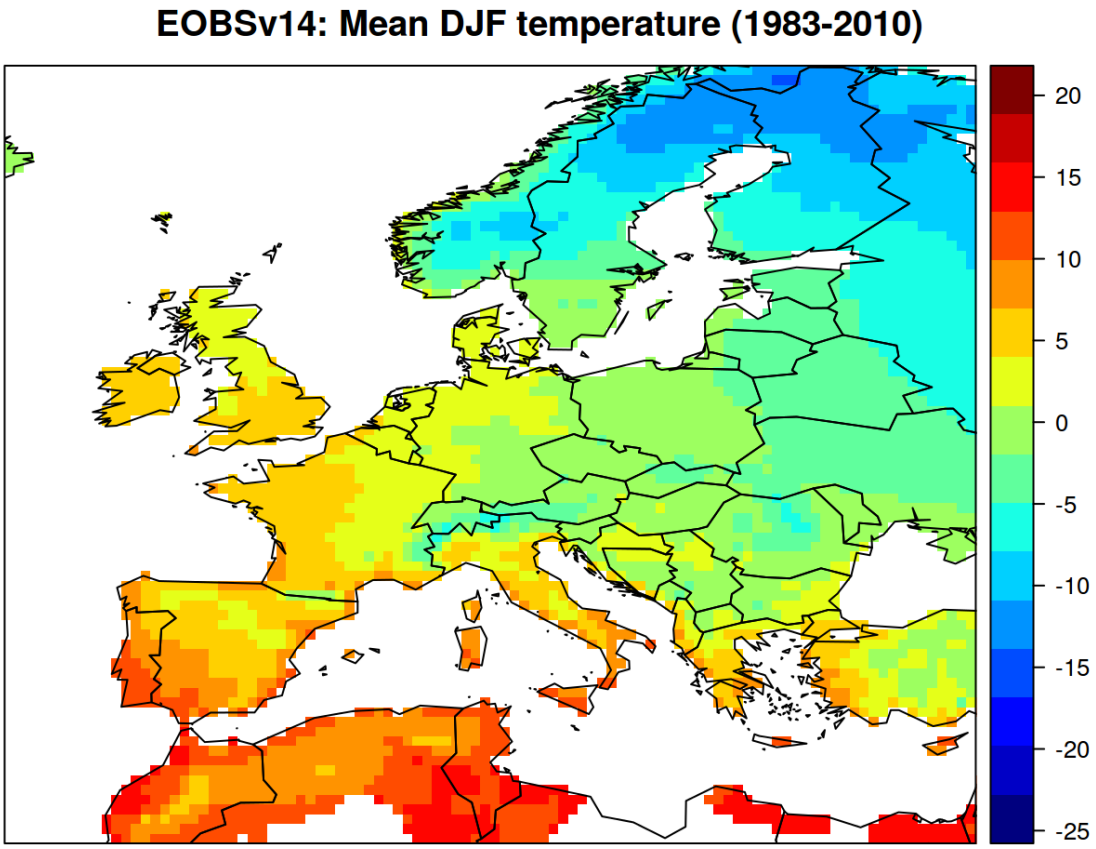
Annual aggregation of loaded data:

```
eobs.tas <- aggregateGrid(eobs.tas, aggr.y = list(FUN = "mean"))
```

This is the climatology of the EOBS dataset for mean winter temperature:

```
plotClimatology(climatology(eobs.tas),
                backdrop.theme = "countries", ## Add map
                main = "EOBSv14: Mean DJF temperature (1983-2010)", ## Add title
                xlim = c(-15,35), ylim = c(30,70)) ## map boundaries
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



Interpolation

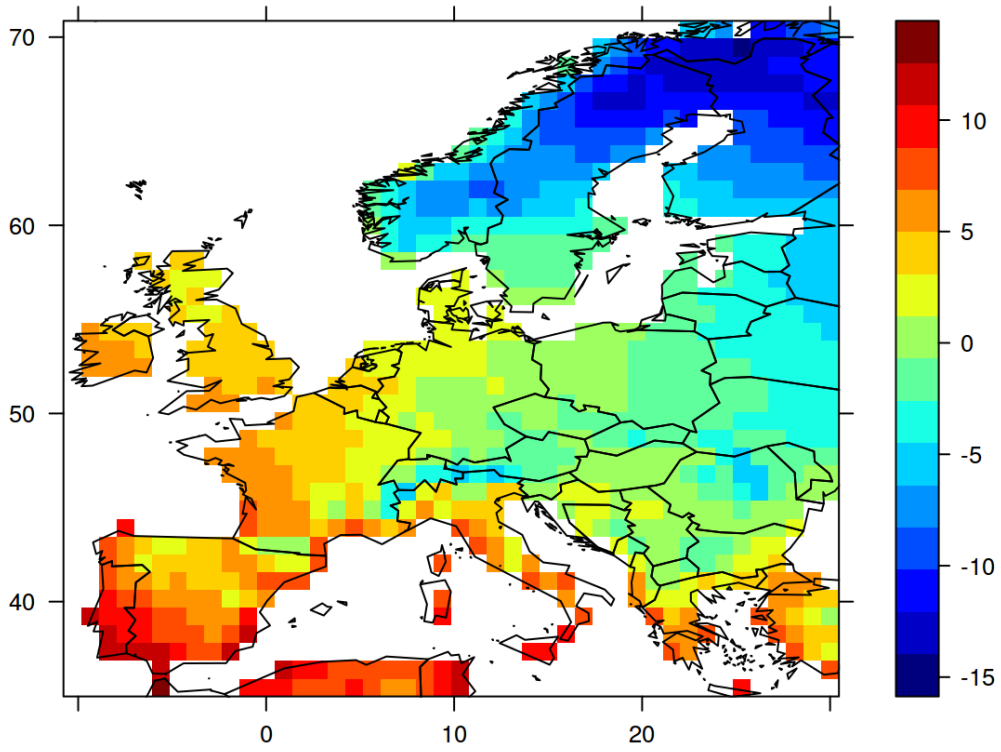
Prior to verification, the EOBS data is interpolated to the model grid, using nearest neighbors:

```
eobs.tas <- interpGrid(eobs.tas, new.coordinates = getGrid(tas.cfs))
```

, and next the reference climatology is plotted.

```
plotClimatology(climatology(eobs.tas),
                backdrop.theme = "countries", ## Add map
                scales = list(draw = TRUE),  ## Add geo coordinates to plot frame
                main = "EOBSv14 in the CFSv2 grid")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



Ensemble Correlation

The function `veriApply` from package `easyVerification` can be directly applied to the climate grids. In this case, we apply the Ensemble correlation verification function (`verifun = "EnsCorr"`).

```
obs <- eobs.tas[["Data"]]
fcst <- tas.cfs[["Data"]]
cor <- veriApply(verifun = "EnsCorr", fcst = fcst, obs = obs, ensdim = 1, tdim = 2)
```

The object `corr` contains the correlation coefficients for each grid point. The function `easyVeri2grid` of package `transformeR` performs the transformation from the raw verification matrices to the grid structure with all the required metadata for data collocation and representation:

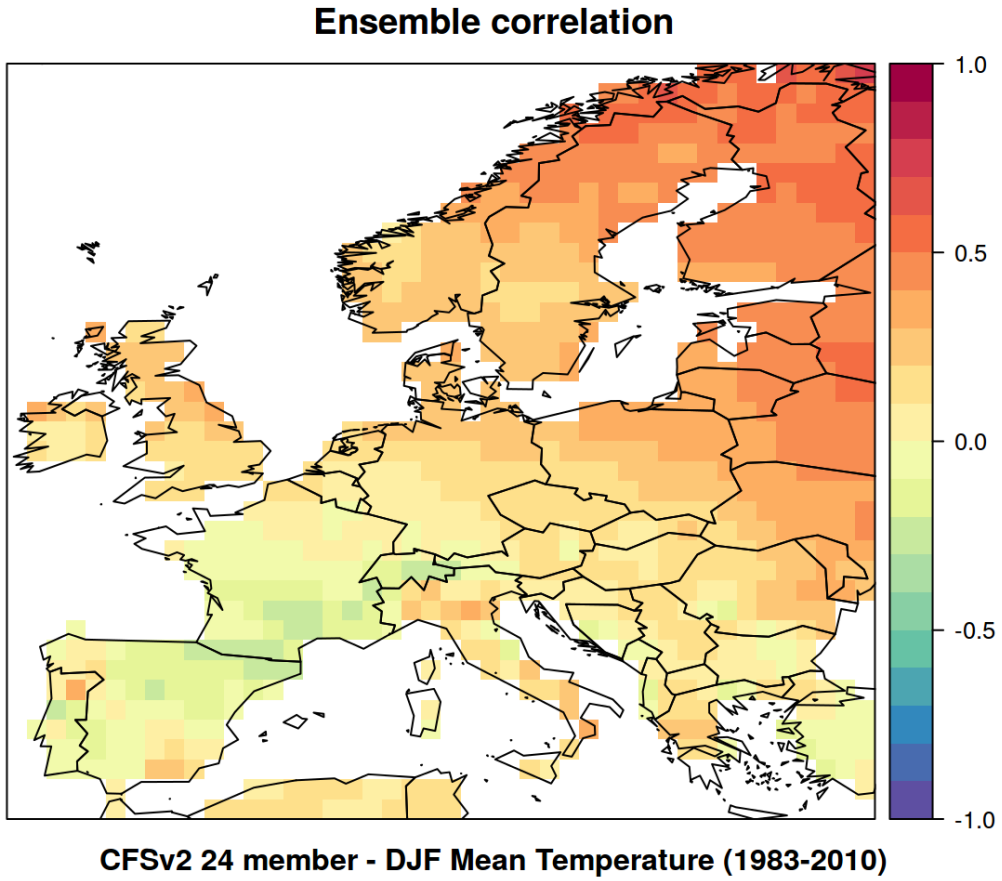
```
cor.grid <- easyVeri2grid(easyVeri.mat = cor, obs.grid = eobs.tas, verifun = "EnsCorr")
```

It is now straightforward the representation of the correlation map, that is treated as a climatology (i.e., the time dimension is a singleton):

```
plotClimatology(climatology(cor.grid),
                backdrop.theme = "countries",
                col.regions = veri.colors(21), ## Define color palette
                at = seq(-1,1,.1), ## Set range of values
                main = "Ensemble correlation",
                sub = "CFSv2 24 member - DJF Mean Temperature (1983-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
```

```
## Use 'spatialPlot' from package visualizeR instead.
```



ROC Area

Next, we compute the Area under the ROC curve, considering the mean temperature terciles:

```
roca <- veriApply(verifun = "EnsRoca",
                  fcst = fcst,
                  obs = obs,
                  prob = c(1/3,2/3), ## Tercile probabilities
                  tdim = 2,
                  ensdim = 1)
```

For the spatial representation, we create a transformer `multiGrid`, that stacks the ROC area maps for the three terciles in the same grid object (note that only the three first columns of object `roca` are considered, since the remaining three correspond to the standard error for each category):

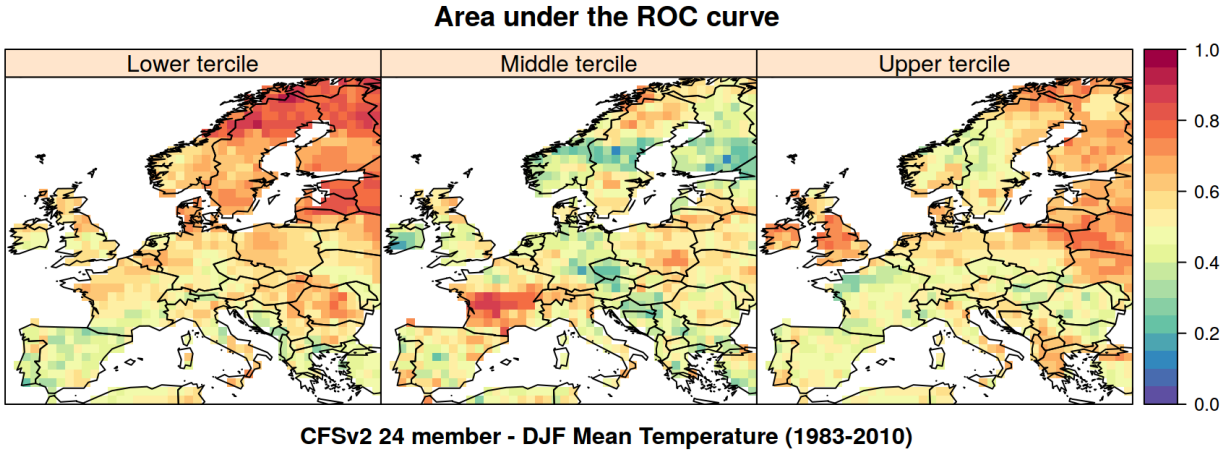
```
l <- lapply(roca[1:3], "easyVeri2grid", eobs.tas)
mg <- makeMultiGrid(l)
str(mg)
```

```
## List of 4
##  $ Variable:List of 2
##   ..$ varName: chr [1:3] "tg" "tg" "tg"
##   ..$ level  : logi [1:3] NA NA NA
```

```
##   ..- attr(*, "use_dictionary")= chr [1:3] "FALSE" "FALSE" "FALSE"
##   ..- attr(*, "description")= chr [1:3] "\"mean temperature\"" "\"mean temperature\"" "\"mean tempera
##   ..- attr(*, "units")= chr [1:3] "\"Celsius\"" "\"Celsius\"" "\"Celsius\""
##   ..- attr(*, "longname")= chr [1:3] "\"tg\"" "\"tg\"" "\"tg\""
##   ..- attr(*, "daily_agg_cellfun")= chr [1:3] "\"none\"" "\"none\"" "\"none\""
##   ..- attr(*, "monthly_agg_cellfun")= chr [1:3] "\"mean\"" "\"mean\"" "\"mean\""
##   ..- attr(*, "verification_time")= chr [1:3] "\"DD\"" "\"DD\"" "\"DD\""
##   ..- attr(*, "annual_agg_cellfun")= chr [1:3] "\"mean\"" "\"mean\"" "\"mean\""
##  $ Data    : num [1:3, 1, 1, 1:38, 1:44] NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "dimensions")= chr [1:5] "var" "member" "time" "lat" ...
##   ..- attr(*, "climatology:fun")= chr "easyVeri"
##  $ xyCoords:List of 2
##   ..$ x: num [1:44] -10.31 -9.38 -8.44 -7.5 -6.56 ...
##   ..$ y: num [1:38] 35.4 36.4 37.3 38.3 39.2 ...
##   ..- attr(*, "resX")= num 0.938
##   ..- attr(*, "resY")= num 0.945
##   ..- attr(*, "projection")= chr "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
##   ..- attr(*, "interpolation")= chr "nearest"
##  $ Dates   :List of 3
##   ..$ :List of 2
##   .. ..$ start: chr [1:28(1d)] "1982-12-01 GMT" "1983-12-01 GMT" "1984-12-01 GMT" "1985-12-01 GMT" .
##   .. ..$ end  : chr [1:28(1d)] "1983-03-01 GMT" "1984-03-01 GMT" "1985-03-01 GMT" "1986-03-01 GMT" .
##   .. ..- attr(*, "subset")= chr "subsetYears"
##   .. ..- attr(*, "season")= int [1:3] 12 1 2
##   ..$ :List of 2
##   .. ..$ start: chr [1:28(1d)] "1982-12-01 GMT" "1983-12-01 GMT" "1984-12-01 GMT" "1985-12-01 GMT" .
##   .. ..$ end  : chr [1:28(1d)] "1983-03-01 GMT" "1984-03-01 GMT" "1985-03-01 GMT" "1986-03-01 GMT" .
##   .. ..- attr(*, "subset")= chr "subsetYears"
##   .. ..- attr(*, "season")= int [1:3] 12 1 2
##   ..$ :List of 2
##   .. ..$ start: chr [1:28(1d)] "1982-12-01 GMT" "1983-12-01 GMT" "1984-12-01 GMT" "1985-12-01 GMT" .
##   .. ..$ end  : chr [1:28(1d)] "1983-03-01 GMT" "1984-03-01 GMT" "1985-03-01 GMT" "1986-03-01 GMT" .
##   .. ..- attr(*, "subset")= chr "subsetYears"
##   .. ..- attr(*, "season")= int [1:3] 12 1 2
##  - attr(*, "dataset")= chr "http://opendap.knmi.nl/knmi/thredds/dodsC/e-obs_0.50regular/tg_0.50deg_r
```

```r
plotClimatology(mg,
                backdrop.theme = "countries",
                col.regions = veri.colors(21), at = seq(0,1,.05),
                names.attr = c("Lower tercile", "Middle tercile", "Upper tercile"),
                layout = c(3,1),
                main = "Area under the ROC curve",
                sub = "CFSv2 24 member - DJF Mean Temperature (1983-2010)")
```

```
## Warning: 'plotClimatology' is deprecated and will eventually be removed from transformeR.
## Use 'spatialPlot' from package visualizeR instead.
```



REFERENCES

- Bedia, J. and Iturbide, M. (2017). transformeR: Climate data post-processing. R package version 0.0.9. https://github.com/SantanderMetGroup/transformeR/wiki.
- Kalnay, E. *et al.* 1996. The NCEP/NCAR 40-Year Reanalysis Project. *Bulletin of the American Meteorological Society* **77**, 437–471. doi:10.1175/1520-0477(1996)077<0437:TNYRP>2.0.CO;2
- Klein Tank, A.M.G. *et al.* 2002. Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment. *Int. J. Climatol.* **22**, 1441–1453. doi:10.1002/joc.773
- MeteoSwiss (2017). easyVerification: Ensemble Forecast Verification for Large Data Sets. R package version 0.4.1. https://CRAN.R-project.org/package=easyVerification
- R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
- Saha, S. *et al.* 2013. The NCEP Climate Forecast System Version 2. *J Clim.* doi:10.1175/JCLI-D-12-00823.1
- Siegert, S. (2017). SpecsVerification: Forecast Verification Routines for Ensemble Forecasts of Weather and Climate. R package version 0.5-2. https://CRAN.R-project.org/package=SpecsVerification
- Wickham, H. and Chang, W. (2016). devtools: Tools to Make Developing R Packages Easier. R package version 1.12.0. https://CRAN.R-project.org/package=devtools

SESSION INFORMATION AND PACKAGE VERSIONS

```
print(sessionInfo(package = c("loadeR",
                              "loadeR.ECOMS",
                              "transformeR",
                              "easyVerification",
                              "SpecsVerification")))
```

```
## R version 3.4.4 (2018-03-15)
```

```
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.4 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##  [3] LC_TIME=es_ES.UTF-8           LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=es_ES.UTF-8       LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8          LC_NAME=es_ES.UTF-8
##  [9] LC_ADDRESS=es_ES.UTF-8        LC_TELEPHONE=es_ES.UTF-8
## [11] LC_MEASUREMENT=es_ES.UTF-8    LC_IDENTIFICATION=es_ES.UTF-8
##
## attached base packages:
## character(0)
##
## other attached packages:
## [1] loadeR_1.4.0            loadeR.ECOMS_1.4.3      transformeR_1.3.3
## [4] easyVerification_0.4.4  SpecsVerification_0.5-2
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16        raster_2.6-7        knitr_1.20
##  [4] magrittr_1.5        maps_3.3.0          grDevices_3.4.4
##  [7] lattice_0.20-35     pbapply_1.3-3       loadeR.java_1.1-0
## [10] stringr_1.3.0       fields_9.6          tools_3.4.4
## [13] utils_3.4.4         parallel_3.4.4      dotCall64_0.9-5.2
## [16] grid_3.4.4          spam_2.1-3          htmltools_0.3.6
## [19] stats_3.4.4         datasets_3.4.4      yaml_2.1.18
## [22] abind_1.4-5         rprojroot_1.3-2     digest_0.6.12
## [25] base_3.4.4          RcppEigen_0.3.3.4.0 Matrix_1.2-14
## [28] rJava_0.9-8         akima_0.6-2         RColorBrewer_1.1-2
## [31] codetools_0.2-15    graphics_3.4.4      bitops_1.0-6
## [34] RCurl_1.95-4.10     evaluate_0.10.1     rmarkdown_1.9
## [37] sp_1.2-7            stringi_1.1.7       compiler_3.4.4
## [40] methods_3.4.4       backports_1.1.2
```