Regional-Continental domain selections

In this example we will load data for Europe for the variable surface (2m) minimum temperature (`var = tasmin`), for the first two members (`members = 1:2`) of the CFSv2 hindcast (dataset = `CFSv2_seasonal_16`), considering the wintertime (DJF, `season = c(12,1,2)`) for the 10-year period 2001-2010 (`years = 2001:2010`), according to the forecast of previous September (`leadMonth = 3`). The original variable is stored as 6-hourly data for this particular dataset. We will retrieve the daily mean values, by setting the argument `time = "DD"`, that internally computes the daily minimum from the 6-hourly instantaneous values (see the NOTE below when executing the command):

```
> ex2 <- loadECOMS(dataset = "CFSv2_seasonal_16", var = "tasmin", members = 1:2, lonLim = c(-15,35), latLim = c(32, 75), s
[2014-09-02 16:45:58] Defining homogeneization parameters for variable "tasmin"
NOTE: daily minimum will be calculated from the 6-h model output
[2014-09-02 16:45:58] Defining geo-location parameters
[2014-09-02 16:45:58] Defining initialization time parameters
[2014-09-02 16:46:03] Retrieving data subset ...
[2014-09-02 16:52:57] Done
> print(object.size(ex2), units = "Mb")
35 Mb
```

In this case, the data are stored in a 4D-array, with the dimensions indicated by the `dimensions`attribute, always following the canonical ordering of dimensions:

```
> str(ex2$Data)
num [1:2, 1:902, 1:47, 1:54] 17.4 17.2 16.4 18.7 17.4 ...
- attr(*, "dimensions")= chr [1:4] "member" "time" "lat" "lon"
```

Members can be plottede individually by setting `multimember = TRUE` in the `plotMeanField` function of the `downscaleR`package:

```
plotMeanField(ex2, multi.member = TRUE)
```

We load now the reference observations for the spatio-temporal domain previously chosen:

```
> ex2.obs <- loadECOMS(dataset = "WFDEI", var = "tasmin", lonLim = c(-15,35), latLim = c(32, 75), season = c(12,1,2), year
[2014-09-02 17:07:43] Defining homogeneization parameters for variable "tasmin"
[2014-09-02 17:07:44] Defining geo-location parameters
[2014-09-02 17:07:44] Defining time selection parameters
[2014-09-02 17:07:44] Retrieving data subset ...
[2014-09-02 17:07:58] Done
> print(object.size(ex2.obs), units = "Mb")
60.6 Mb
```

This is the map of the observed mean minimum surface temperature observed for DJF 2001-2010:

```
> plotMeanField(ex2.obs)
```

Note that WFDEI provides data for land areas only, and its spatial resolution is finer than CFS (1º vs 0.5º). In order to compare both datasets, it is first necessary to put them in the same grid (i.e., to interpolate). We use bilinear interpolation to this aim, using the `downscaleR` function `interpGridData` in combination with the `getGrid` method, useful to recover the parameters defining the grid of a dataset to pass them to the interpolator:

```
> obs.regridded <- interpGridData(gridData = ex2.obs, new.grid = getGrid(ex2), method = "bilinear")
[2014-09-02 17:22:11] Performing bilinear interpolation... may take a while
[2014-09-02 17:22:30] Done
Warning messages:
1: In interpGridData(gridData = ex2.obs, new.grid = getGrid(ex2), method = "bilinear") :
 The new longitudes are outside the data extent
```

```
2: In interpGridData(gridData = ex2.obs, new.grid = getGrid(ex2), method = "bilinear") :
 The new latitudes are outside the data extent
```

Note the warnings reminding us that the extent of the input grid is wider that that of CFS. However, in this case we can safely ignore this warnings, since all the land areas we are interest in are within the CFS domain.

```
> plotMeanField(obs.regridded)
```

Now that both model data and observations are in the same grid, we can compute the bias. First, we calculate the spatial mean of WFDEI, which is the reference against which to compute the biases:

```
> ref <- apply(obs.regridded$Data, MARGIN = c(3,2), mean, na.rm = TRUE)
```

The following lines of code compute the bias of each member w.r.t. the reference and plot them:

```
# Now we compute the difference agains each of the multimember spatial means:
> require(fields)
> n.members <- dim(ex2$Data)[1]
> par(mfrow = c(1,2))
> for (i in 1:n.members) {
+       member <- apply(ex2$Data[i, , , ], MARGIN = c(3,2), mean, na.rm = TRUE)
+       bias <- member - ref
+       image.plot(ex2$xyCoords$x, ex2$xyCoords$y, bias, xlab = "lon", ylab = "lat", asp = 1)
+       title(paste("Bias member", i))
+       world(add = TRUE)
+ }
> par(mfrow = c(1,1)) # To reset the graphical window
```