Full code for Example 1 of the paper 'climate4R: An Ecosystem of R packages for Climate Data Access, Post-processing and Bias Correction'

M. Iturbide, J. Bedia, S. Herrera, J. Baño, J. Fernández, M. D. Frías, R. Manzanas, D. San Martín, E. Cimadevilla, A.S. Cofiño, J. M. Gutiérrez

2018-04-11

Contents

1 Introduction

This worked example contains the full code that reproduces the 1rst example of the paper "climate4R: An Ecosystem of R packages for Climate Data Access, Post-processing and Bias Correction" (Sec. 5 of the manuscript). Here, the same titles are used for the main sections (see index). These are divided in additional subsections to help with the understanding of the different code chunks. All operations hereinafter are performed with the core packages of climate4R, excepting package installation and the creation of color palettes, for which packages `devtools` and `RColorBrewer` are used respectively. climate4R packages are installed by means of the 'devtools' package:

```
library(devtools)
install_github(c("SantanderMetGroup/loadeR",
                 "SantanderMetGroup/loadeR.java",
                 "SantanderMetGroup/transformeR",
                 "SantanderMetGroup/visualizeR",
                 "SantanderMetGroup/downscaleR",
                 "SantanderMetGroup/climate4R.climdex")
```

```
library(loadeR)
library(transformeR)
library(visualizeR)
library(downscaleR)
library(climate4R.climdex)
```

2 Example 1: Climate Indices from CORDEX Projections

2.1 Loading, collocating and harmonizing data

The domain of the study area is defined by the following bounding coordinates:

```
lon <- c(-10, 20)
lat <- c(35, 46)
```

2.1.1 Cliamte data loading from OPeNDAP server: E-OBS observational data

As described in the paper, the SU index (summer days) can be obtained on-the-fly by loading maximum temperature data with function `loadGridData` and by the following argument settings: `aggr.m = "sum"`, `condition = "GT"` and `threshold = 25`. First we load E-OBS observational data by pointing to a NetCDF file via OPeNDAP. Previous to loading, function `dataInventory` might be applied for an overview of the dataset, which returns an inventory (object `di`) of the available variables names, units, coordinates, etc.

```
eobs<-"http://opendap.knmi.nl/knmi/thredds/dodsC/e-obs_0.25regular/tx_0.25deg_reg_v16.0.nc"
di <- dataInventory(eobs)
```

In this case, the NetCDF file contains maximum temperature data named as "tx", thus, we set `var = "tx"` when calling to `loadGridData`:

```
SU <- loadGridData(eobs, var = "tx",
                   season = 1:12,
                   years = 1971:2000,
                   lonLim = lon,
                   latLim = lat,
                   aggr.m = "sum",
                   condition = "GT",
                   threshold = 25)
```

2.1.1.1 Using a dictionary

In order to load and work with harmonized data we can repeat the above operation using a dictionary file, that defines the necessary name and unit transformations to the standard parameters. Function `C4R.vocabulary` displays the climate4R standard variable naming and units:

```
C4R.vocabulary()
```

```
##    identifier                            standard_name          units
## 1        hurs               2-meter relative humidity              %
## 2     hursmax       maximum 2-meter relative humidity              %
## 3     hursmin       minimum 2-meter relative humidity              %
## 4         hus                       specific humidity        kg.kg-1
## 5        huss               2-meter specific humidity        kg.kg-1
## 6     hussmax       maximum 2-meter specific humidity        kg.kg-1
## 7     hussmin       minimum 2-meter specific humidity        kg.kg-1
## 8          lm                        land binary mask              1
## 9        orog                        surface altitude              m
## 10         ps              air pressure at surface level             Pa
## 11        psl                 air pressure at sea level             Pa
## 12       rlds  surface downwelling longwave radiation          W.m-2
## 13       rlut                toa outgoing longwave flux          W.m-2
## 14       rsds surface downwelling shortwave radiation          W.m-2
## 15      sftlf                        land area fraction               1
## 16         ta                   air temperature degrees Celsius
## 17        tas          2-meter air temperature degrees Celsius
## 18     tasmax         maximum 2-m air temperature degrees Celsius
## 19     tasmin         minimum 2-m air temperature degrees Celsius
## 20       tdps      2-meter dewpoint temperature degrees Celsius
## 21         pr            total precipitation amount              mm
## 22        prr                 total rainfall amount              mm
## 23       prsn                 total snowfall amount              mm
## 24         ua                         eastward wind           m.s-1
## 25        uas            eastward near-surface wind           m.s-1
## 26         va                        northward wind           m.s-1
## 27        vas           northward near-surface wind           m.s-1
## 28        wss               near-surface wind speed           m.s-1
## 29     wssmax       maximum near-surface wind speed           m.s-1
## 30        wsg                    wind speed of gust           m.s-1
## 31     wsgmax            maximum wind speed of gust           m.s-1
## 32          z                          geopotential          m2.s-2
## 33         zg                   geopotential height               m
## 34         zs                 surface geopotential          m2.s-2
## 35        zsg           surface geopotential height               m
```

In this case, the only non-standard parameter in the E-OBS dataset is the variable name ("tx"), however, we could perform further loading requests using the standard name if a dictionary file is crated previously (see the loadeR wiki). This can be done easily, for instance, in the following manner:

```
file.create("dicEOBS.dic")
writeLines(c("identifier,short_name,time_step,lower_time_bound,upper_time_bound,
             cell_method,offset,scale,deaccum,derived,interface",
             "tasmax,tx,24h,0,24,max,0,1,0,0,"), "dicEOBS.dic")
```

Next the loading operation is repeated but using the standard name for the maximum temperature (`var = "tasmax"`) and by passing the path to our *.dic file ("dicEOBS.dic") in argument `dictionary`:

```
SU <- loadGridData(eobs,
                         var = "tasmax",
                         season = 1:12,
                         lonLim = lon,
                         latLim = lat,
                         years = 1971:2000,
                         aggr.m = "sum",
                         threshold = 25,
                         condition = "GT",
                         dictionary = "dicEOBS.dic")
```

2.1.1.2 Transformation and visualization

Note that `loadGridData` returns monthly summer days (SU). To compute the annual index we only need to apply function `aggregateGrid` that performs the aggregation of the desired data dimension (in this case `time`). We use argument `aggr.y` to perform annual aggregation with function `sum`:

```
SU.annual <- aggregateGrid(SU, aggr.y = list(FUN = "sum"))
```

Type `?aggregateGrid` to see other aggregation options.

At this point we can plot the first map by using function `spatialPlot`, which by default incorporates a color palette for drawing maps. However, we could use the desired color range. We recommend package

RColorBrewer to create palettes with function brewer.pal. The ones used in the manuscript are the following:

```
library(RColorBrewer)
colstx <- rev(brewer.pal(n = 9, "Spectral"))
colsindex <- rev(brewer.pal(n = 9, "RdYlBu"))
colsdelta <- brewer.pal(n = 9, "Reds")
colsbias <- brewer.pal(n = 9, "PiYG")
colssd <- brewer.pal(n = 9, "Blues")
```

In this case we set col.regions = colorRampPalette(colsindex) to visualize the mean annual SU for the reference period (1971-2000). As a result Figure 1 is generated (Fig. 2a in the manuscript).

```
spatialPlot(climatology(SU.annual), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

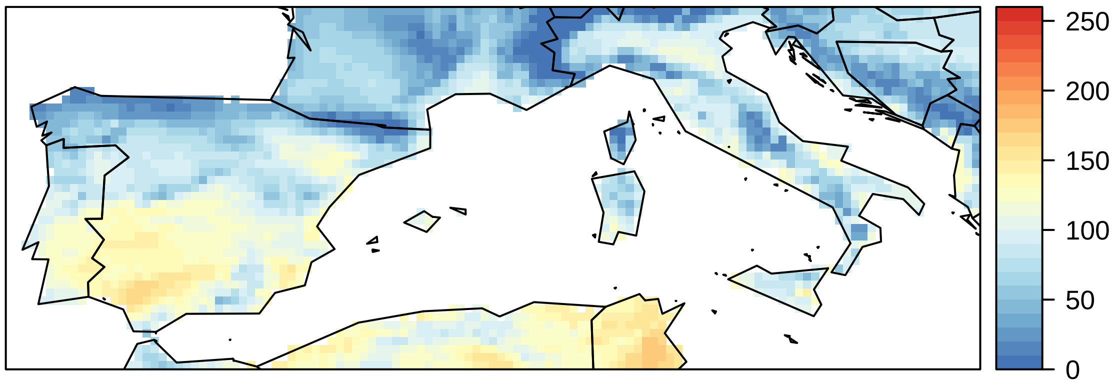

Figure 1: Southern Europe summer days for E-OBS and the historical period 1971-2000. Fig. 2a in the manuscript.

2.1.2 Cliamte data loading from local files: CORDEX climate change projections

Next, projection data (for both the historical and the RCP8.5 scenarios) is loaded from local NetCDF files, which correspond to a particular RCM (Regional Climate Model ICHEC-EC-EARTH_r12i1p1_SMHI-RCA4_v1) from EURO-CORDEX. These files were downloaded from ESGF (see Appendix A in the manuscript) and stored locally. Next we list them in objects dir and dirf, the first corresponding to the historical scenario and the second to the future RCP8.5.

```
#historical data
dirh <- "/myDirectoryOfHistoricalData/"
#climate change data
dirf <- "/myDirectoryOfClimateChangeData/"
list.files(dirh, recursive = T)
```

```
##  [1] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20060101-20101231.nc"
##  [2] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20110101-20151231.nc"
##  [3] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20160101-20201231.nc"
##  [4] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20210101-20251231.nc"
##  [5] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20260101-20301231.nc"
##  [6] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20310101-20351231.nc"
##  [7] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20360101-20401231.nc"
##  [8] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20410101-20451231.nc"
##  [9] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20460101-20501231.nc"
## [10] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20510101-20551231.nc"
## [11] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20560101-20601231.nc"
## [12] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20610101-20651231.nc"
## [13] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20660101-20701231.nc"
## [14] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20710101-20751231.nc"
## [15] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20760101-20801231.nc"
## [16] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20810101-20851231.nc"
## [17] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20860101-20901231.nc"
## [18] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20910101-20951231.nc"
## [19] "tasmax_EUR-44_ICHEC-EC-EARTH_rcp85_r12i1p1_SMHI-RCA4_v1_day_20960101-21001231.nc"
```

Each file in the list contains data for a 5-year period of the same variable (tasmax). Therefore, we use a "catalog" (*.ncml file) to load data for the required period without worrying about the different files that need to be read and bound. Next we create two catalogs (for each scenario) with function `makeAggregateDataset` ("CDX_hist.ncml" and "CDX_rcp85.ncml"):

```r
makeAggregatedDataset(source.dir = dir, recursive = T, ncml.file = "CDX_hist.ncml")
makeAggregatedDataset(source.dir = dirf, recursive = T, ncml.file = "CDX_rcp85.ncml")
```

The created *.ncml files are then used as a single access point to load data and to do the data inventory as well:

```r
di <- dataInventory("CDX_hist.ncml")
str(di$tasmax)
```

```
## List of 4
##  $ Description: chr "Daily Maximum Near-Surface Air Temperature"
##  $ DataType   : chr "float"
##  $ Units      : chr "K"
##  $ Dimensions :List of 3
##   ..$ time:List of 4
##   .. ..$ Type      : chr "Time"
##   .. ..$ TimeStep  : chr "1.0 days"
##   .. ..$ Units     : chr "days since 1949-12-01 00:00:00"
##   .. ..$ Date_range: chr "1951-01-01T12:00:00Z - 2005-12-31T12:00:00Z"
##   ..$ lat :List of 3
##   .. ..$ Type  : chr "GeoY"
##   .. ..$ Units : chr "degrees"
##   .. ..$ Values: num [1:103] -23.2 -22.8 -22.3 -21.9 -21.4 ...
##   ..$ lon :List of 3
##   .. ..$ Type  : chr "GeoX"
##   .. ..$ Units : chr "degrees"
##   .. ..$ Values: num [1:106] -28.2 -27.8 -27.3 -26.9 -26.4 ...
```

Contrarily to the case of the E-OBS dataset, the variable name is standard, but not the units (K). Therefore we define the harmonization parameters in another dictionary file ("dicCDX.dic"), where the offset is -273.15 to convert the data to the standard units (ºC):

```r
file.create("dicCDX.dic")
writeLines(c("identifier,short_name,time_step,lower_time_bound,upper_time_bound,
             cell_method,offset,scale,deaccum,derived,interface",
             "tasmax,tasmax,24h,0,24,max,-273.15,1,0,0,"), "dicCDX.dic")
```

2.1.2.1 Historical data

Next, harmonized data is loaded for a single CORDEX model, for the historical scenario and the same reference period used to load E-OBS observational data (1971-2000):

```r
SUh <- loadGridData(dataset = "CDX_hist.ncml",
                    var = "tasmax",
                    season = 1:12,
                    lonLim = lon,
                    latLim = lat,
                    years = 1971:2000,
                    aggr.m = "sum",
                    threshold = 25,
                    condition = "GT",
                    dictionary = "dicCDX.dic")
```

The same operations of annual aggregation and visualization shown before are repeated next. As a result Figure 2 is obtained (Fig. 2b in the manuscript).

```r
SUh.annual <- aggregateGrid(SUh, aggr.y = list(FUN = "sum"))
```

```r
spatialPlot(climatology(SUh.annual), at = seq(0, 260, 10),
            col.regions = colorRampPalette(colsindex))
```

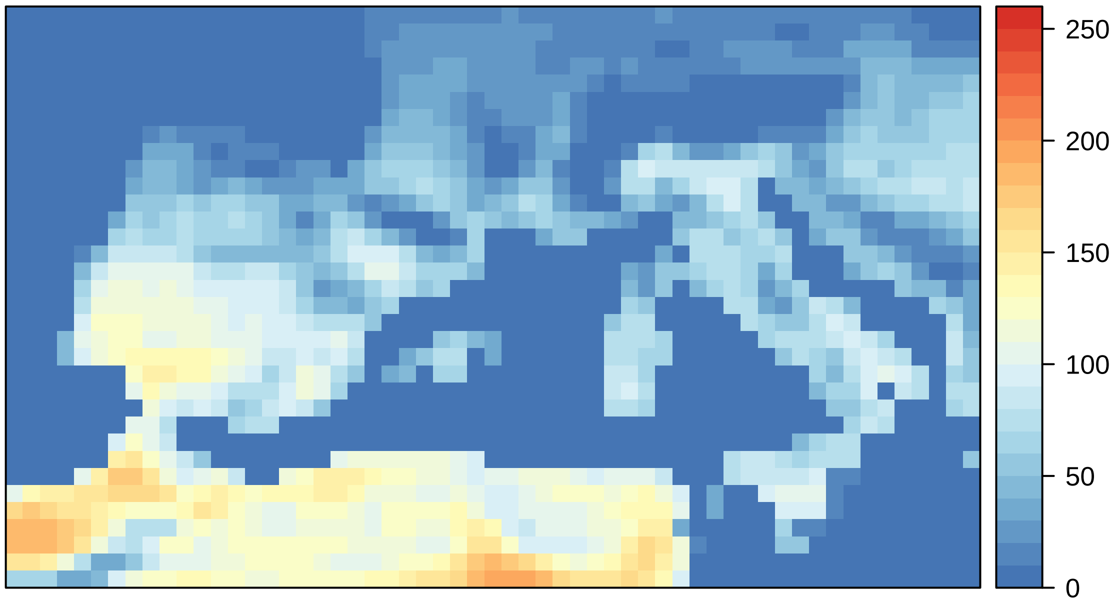

Figure 2: Southern Europe summer days for CORDEX and the historical period 1971-2000. Fig. 2b in the manuscript.

As can be noted in Figure 2, the spatial grid of CORDEX is different from E-OBS (Figure 1). We can use function `interpGrid` to interpolate CORDEX data to the E-OBS spatial grid, allowing the subsequent extraction of the SU bias in the reference period (1971-2000). This is done by subtracting the SU index of E-OBS (object `SU.annual`) to the SU index of historical CORDEX (object `SUh.interp`), for which function `gridArithmetics` is used.

Despite not being necessary, here we apply a land mask before calculating the bias in order to eliminate the values projected by the CORDEX model over the sea. To do so, `gridArithmetics` might be also used, first to create the mask and second to apply it.

```r
SUh.interp <- interpGrid(SUh.annual, getGrid(SU.annual))

eobs.mask <- gridArithmetics(SU.annual, 0, operator = "*")
SUh.interp <- gridArithmetics(SUh.interp, eobs.mask, operator = "+")
```

```
bias <- gridArithmetics(SUh.interp, SU.annual, operator = "-")
```

Next we plot the SU index for CORDEX (object `SUh.interp`) and its bias (object `bias`) to generate Figures 3 and 4 (Figs. 2c and 2d in the manuscript).

```
spatialPlot(climatology(SUh.interp), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

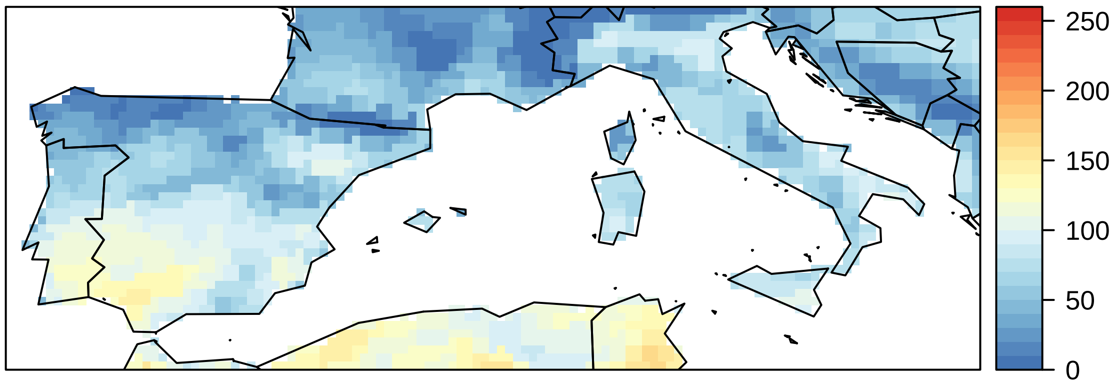

Figure 3: Southern Europe summer days for interpolated CORDEX and the historical period 1971-2000. Fig. 2c in the manuscript.

```
spatialPlot(climatology(bias), backdrop.theme = "countries",
            at = seq(-100, 100, 10), col.regions = colorRampPalette(colsbias))
```

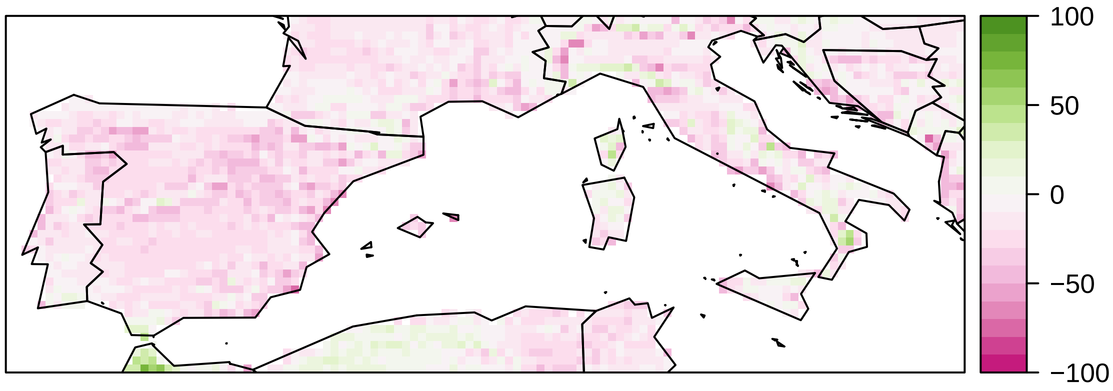

Figure 4: Southern Europe summer days bias for CORDEX and the historical period 1971-2000. Fig. 2d in the manuscript.

2.1.2.2 Future data

We repeat the same operations of data loading and transformation but for the RCP8.5 scenario and future period 2071-2100:

```
SUf <- loadGridData(dataset = "CDX_rcp85.ncml",
                    var = "tasmax",
                    season = 1:12,
                    lonLim = lon,
                    latLim = lat,
                    years = 2071:2100,
                    aggr.m = "sum",
                    threshold = 25,
                    condition = "GT",
                    dictionary = "dicCDX.dic")
```

```
SUf.annual <- aggregateGrid(SUf, aggr.y = list(FUN = "sum"))
```

Note that in this case the application of `gridArithmetics` gives the projected climate change signal (object `CCsignal`) w.r.t the historical period (object `SUh.interp`).

```
SUf.interp <- interpGrid(SUf.annual, getGrid(SU.annual))
SUf.interp <- gridArithmetics(SUf.interp, eobs.mask, operator = "+")

CCsignal <- gridArithmetics(SUf.interp,
                            SUh.interp,
                            operator = "-")
```

Figures 5 and 6 are generated next, which show the future SU index and the climate change signal (Figs. 3a and 3b in the manuscript).

```
spatialPlot(climatology(SUf.interp), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

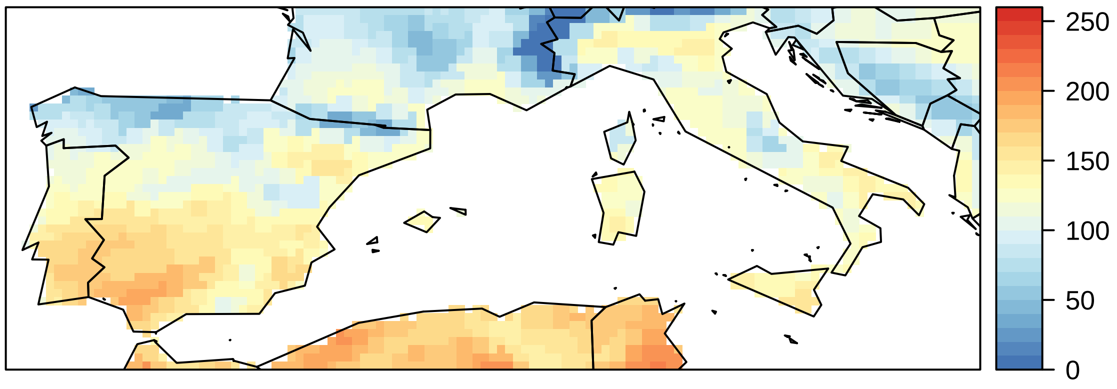

Figure 5: Southern Europe summer days for the interpolated EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100. Fig. 3a in the manuscript.

```
spatialPlot(climatology(CCsignal), backdrop.theme = "countries",
            at = seq(0, 80, 5), col.regions = colorRampPalette(colsdelta))
```

2.2 Post-processing: Bias Correction

Next the “additive” type of the “scaling” method is applied to bias correct future monthly CORDEX data (object `SUf`) by means of function `biasCorrection`. The output is annually aggregated (object `SUf.bc.annual`) and the climate change signal is again calculated from the bias corrected data (object `CCsignal.bc`).

```
SUf.bc <- biasCorrection(y = SU, x = SUh, newdata = SUf,
                         method = "scaling", scaling.type = "additive")
SUf.bc.annual <- aggregateGrid(SUf.bc, aggr.y = list(FUN = "sum"))

CCsignal.bc <- gridArithmetics(SUf.bc.annual,
                               SU.annual,
                               operator = "-")
```

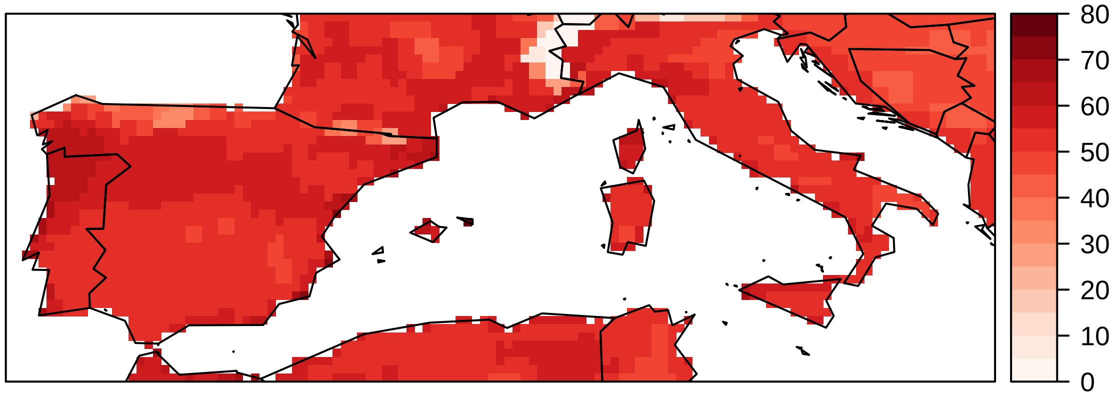

Figure 6: Southern Europe summer days ‘delta’ for the EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100. Fig. 3b in the manuscript.

By plotting the resulting objects we obtain Figures 7 (Fig. 3c in the manuscript) and 8 (not shown in the manuscript):

```r
spatialPlot(climatology(SUf.bc.annual), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

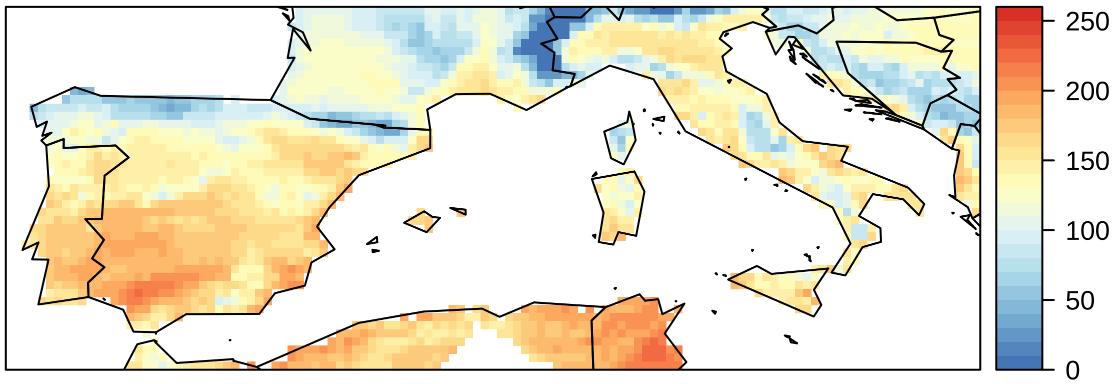

Figure 7: Southern Europe summer days for the bias corrected (additive scaling) EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100. Fig. 3c in the manuscript.

```r
spatialPlot(climatology(CCsignal.bc), backdrop.theme = "countries",
            at = seq(0, 80, 5), col.regions = colorRampPalette(colsdelta))
```

Other useful plotting function is `temporalPlot` that displays temporal series of multiple datasets and periods on the same plot. Here we plot the series corresponding to a single grid box (the one nearest to Zaragoza, Spain), therefore, a previous spatial subsetting is done with function `subsetGrid` (`latLim = 41.64` and `lonLim = -0.89`). As a result object `ts` is obtained, a list that contains the SU index for E-OBS and CORDEX. If several grid boxes are considered (e.g. the whole domain) `temporalPlot` performs the spatial (`lat` and `lon` dimensions) aggregation before plotting (the `mean` is computed by default, type `?temporalPlot`).

Note that function `temporalPlot` is based on `lattice` and arguments from function `xyplot` are optionally passed to argument `xyplot.custom`, allowing for a fine tuning of multiple graphical parameters. The next code chunk generates Figure 9 (Fig. 4 in the manuscript).

```r
ts <- lapply(list("E-OBS" = SU.annual, "CDX_hist" = SUh.interp,
                  "CDX_rcp85" = SUf.interp, "CDX_rcp85_corrected" = SUf.bc.annual),
            function(x)  subsetGrid(x, latLim = 41.64, lonLim = -0.89))
cols = c("black", "red", "red", "blue")
```

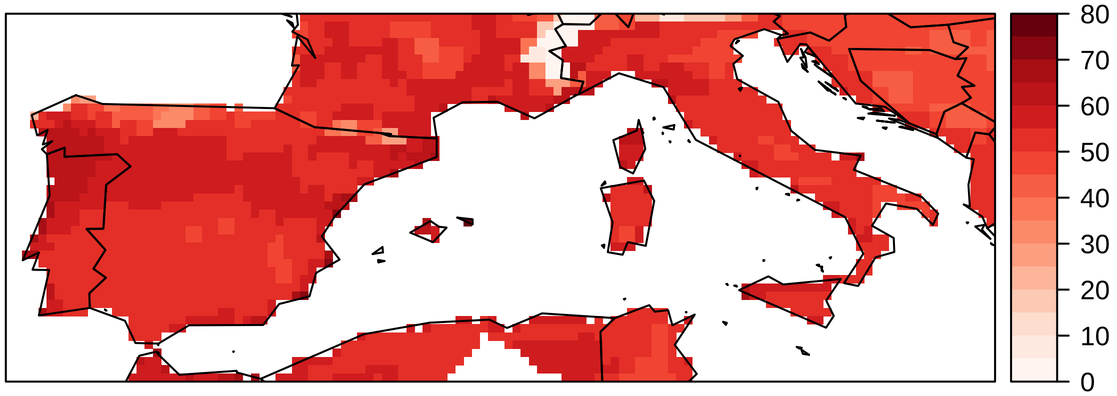

Figure 8: Southern Europe summer days 'delta' for the bias corrected (additive scaling) EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100. Not shown in the manuscript.

```r
temporalPlot(ts, cols = cols, lwd = 0.8, xyplot.custom = list(ylab = "", ylim = c(70, 220),
  key = list(space = "top", lines = list(pch = 15, col = cols, cex = .5),
             text = list(names(ts),
                         cex = .7), columns = 2)))
```

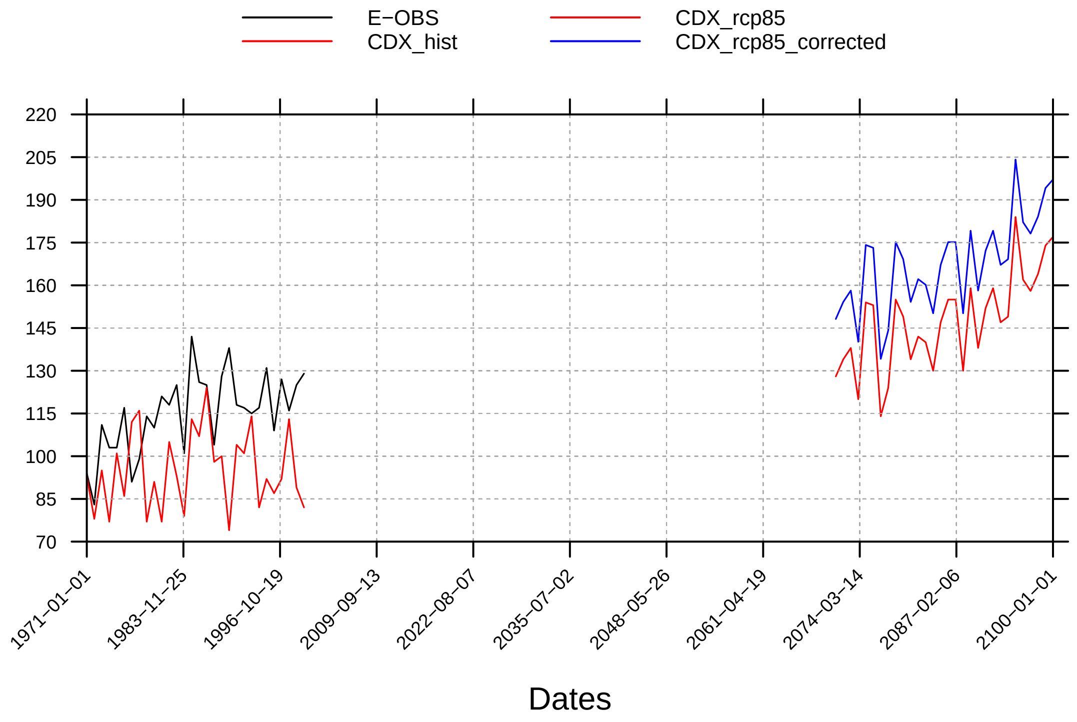

Figure 9: Annual summer days time series for a single gridbox (Zaragoza, Spain) for the observations (E-OBS) and the projection (original and bias corrected) in the historical and future periods. Fig. 4 in the manuscript.

2.3 Working with daily data

Alternatively, the SU index could be calculated using the `climate4R.climdex` package from the original variable (maximum temperature). To do so, we first load daily maximum temperature data by using the previously created dictionaries:

```
TX <- loadGridData(eobs,
                   var = "tasmax",
                     season = 1:12,
                         lonLim = lon,
                         latLim = lat,
                         years = 1971:2000,
                         dictionary = "dicEOBS.dic")
TXh <- loadGridData(dataset = "CDX_hist.ncml",
                     var = "tasmax",
                     season = 1:12,
                     lonLim = lon,
                     latLim = lat,
                     years = 1971:2000,
                     dictionary = "dicCDX.dic")
TXf <- loadGridData(dataset = "CDX_rcp85.ncml",
                     var = "tasmax",
                     season = 1:12,
                     lonLim = lon,
                     latLim = lat,
                     years = 2071:2100,
                     dictionary = "dicCDX.dic")
```

Since we are now working with daily data, we can use the EQM (Empirical Quantile Mapping) method to bias correct the original variable. As pointed in the previous section, CORDEX projections are built over rotated grids. Nevertheless, function `biasCorrection` performs data interpolation internally taking as spatial reference the grid of observation data (E-OBS, object `TX`). Therefore, it is not necessary to use `interpGrid` before applying `biasCorrection`.

```
TXf.bc <- biasCorrection(y = TX,
                         x = TXh,
                         newdata = TXf,
                         method = "eqm",
                         window = c(30, 7),
                         extrapolation = "constant")
```

Next, we calculate the annual SU index with function `climdexGrid` for future raw (object `SUf`) and bias corrected (object `SUf.bc`) CORDEX data:

```
SUf <- climdexGrid(tx = TXf, index.code = "SU")
SUf.bc <- climdexGrid(tx = TXf.bc, index.code = "SU")
```

To obtain comparable maps and/or perform further operations between the obtained results (e.g. using `gridArithmetics`), we can interpolate the raw SU index (function `interpGrid`) to the E-OBS spatial grid and apply the land-sea mask (function `gridArithmetics`) in the manner previously shown:

```
SUf.interp <- interpGrid(SUf, getGrid(TX))
SUf.interp <- gridArithmetics(SUf.interp, eobs.mask, operator = "+")
```

Finally, the maps of raw (Fig. 10) and bias corrected (Fig. 11) SU index for the EC-EARTH driven, RCP8.5 scenario (period 2071-2100) are plotted resulting in Figures 10 (Not shown in the manuscript) and 11 (Fig. 5 in the manuscript).

```
spatialPlot(climatology(SUf.interp), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

```
spatialPlot(climatology(SUf.bc), backdrop.theme = "countries",
            at = seq(0, 260, 10), col.regions = colorRampPalette(colsindex))
```

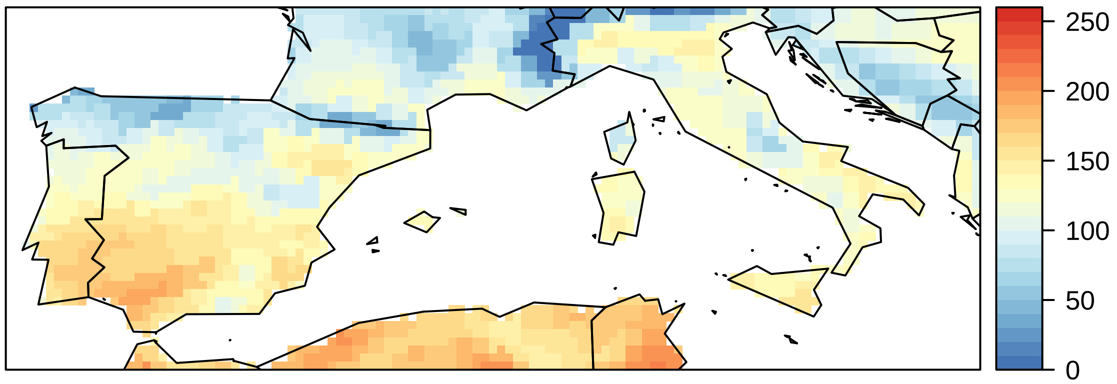

Figure 10: Southern Europe summer days for the EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100 (calculated with package climate4R.climdex from daily data). Not shown in the manuscript.

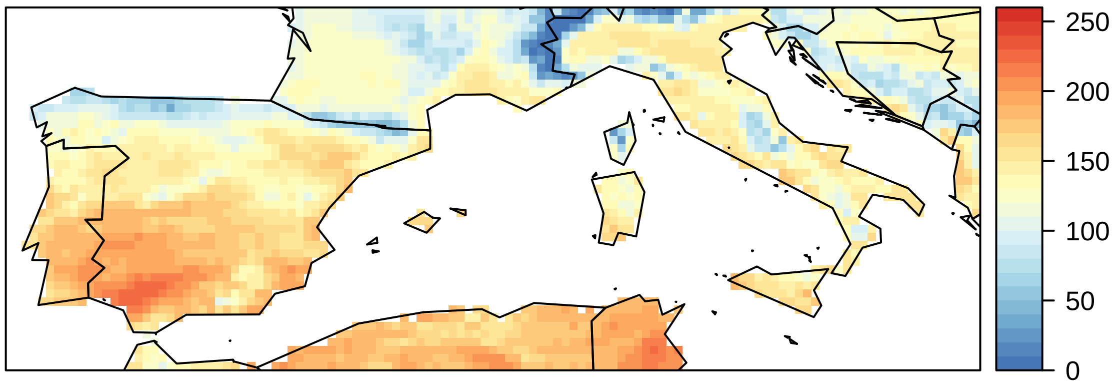

Figure 11: Southern Europe summer days for the bias corrected (emipirical quantile mapping) EC-EARTH driven, RCP8.5 scenario in the future period 2071-2100 (calculated with package climate4R.climdex from daily data). Fig. 5 in the manuscript.

3 Other available material

- 2018_climate4R_example2.pdf contains the full code for **Example 2** of the paper 'climate4R: An Ecosystem of R packages for Climate Data Access, Post-processing and Bias Correction'.
- Find more worked examples on the utilization of climate4R packages in their respective GitHub **wiki**-s at https://github.com/SantanderMetGroup:
  - loadeR: https://github.com/SantanderMetGroup/loadeR/wiki
  - transformeR: https://github.com/SantanderMetGroup/transformeR/wiki
  - downscaleR: https://github.com/SantanderMetGroup/downscaleR/wiki
  - visualizeR: https://github.com/SantanderMetGroup/visualizeR/wiki