Seasonal Predictions of Fire Weather Index: Paving the Way for their operational applicability in Mediterranean Europe

Companion R examples of the paper published in Climate Services (2017, DOI:10.1016/j.cliser.2017.04.001)

J. Bedia & M. Iturbide

2017-04-26

Abstract

This is a worked example that reproduces the steps followed in Bedia et al. 2017 using the R-package `fireDanger`, that contains functions for computing the different components of the Fire Weather Index System (FWI, van Wagner 1987) from station and numerical climate model data.

Packages `loadeR.ECOMS`, `downscaleR` and `easyVerification` are also used for climate data loading, manipulation, bias correction and verification. Furthermore, in the last section of this tutorial we also illustrate the use of different functionalities from the `visualizeR` package for seasonal forecast verification and visualization.

The code provided in this example allows the reproducibility of all the analyses presented in the paper. However, for the sake of openness, data from the CFSv2 seasonal model (Saha et al 2010) are used instead of System4 (see the paper), the first being publicly available without restrictions. The reference observations dataset (WFDEI, Weedon et al 2014) is also publicly available.

Installing the packages

The R packages used in this example are available through their respective public GitHub repositories. In order to avoid problems derived from version changes (all packages are currently under current development and some of the features can be expected to change in the near future), it is recommended to install the "tagged" versions used to run these examples.

To this aim, the usage of function `install_github` from the `devtools` package is recommended. The specific package versions can be installed using the `@` symbol followed by the version tag label. Note that the most recent stable releases of the different packages are available through the master branch. In this case, the `@master` suffix should be used.

Installing the R Climate Processing Bundle from the Santander MetGroup

This is a bundle of packages developed by the Santander Meteorology Group (University of Cantabria, Spain), that loads climate datasets from local or remote (e.g. OPeNDAP) data servers and performs climate data transformation and post-processing, including statistical downscaling, bias correction, visualization and verification. The emphasis has been done on simplicity from the user perspective, while many advanced features are available for advanced users. Other key features inspiring its design have been extensibility (it is easy to link other packages for other climate-realted analysis, for instance FWI calculation using `fireDanger`) and efficiency (e.g., parallelization options are available in many functions). The following packages compose the bundle:

- `loadeR`: tools for climate data loading from local or remote locations
- `loadeR.ECOMS`: a loadeR extension to access the Santander MetGroup User Data Gateway for the ECOMS Project
- `transformeR`: climate data analysis and transformation

- `downscaleR`: statistical downscaling and bias correction methods
- `visualizeR`: tools for seasonal forecast verification and visualization
- Other packages. These are not used in this example, but other utilities exist like for instance `loadeR.2nc` for exporting R climate data structures into CF-compliant netCDF4 files.

In this example, we will first illustrate the use of `loadeR.ECOMS`, required for accessing the OPeNDAP server from the Santander MetGroup , storing a number of seasonal forecast datasets and variables in the so called User Data Gateway, described in the next sections. A number of dependencies are installed, although only `loadeR.ECOMS` is explicitly loaded by the user.

```
devtools::install_github(c("SantanderMetGroup/loadeR.java@v1.1-0",
                  "SantanderMetGroup/loadeR@v1.0-7",
                  "SantanderMetGroup/loadeR.ECOMS@v1.2.3"))
```

In case any problems arise with the installation at this point, see this help guide at the loadeR's wiki page.

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

The package `transformeR` allows performing climate data transformations, including Principal Component/EOF analysis, subsetting, aggregation, regridding/interpolation... In this vignette, the subsetting, aggregation, detrending and visualization capabilitites of `transformeR` will be used.

```
devtools::install_github("SantanderMetGroup/transformeR@v0.0.8")
```

```
library(transformeR)
```

The package `downscaleR` implements several methods of bias correction and perfect-prog downscaling. It will be used for the bias-correction experiment illustrated in the article.

```
devtools::install_github("SantanderMetGroup/downscaleR@v2.0-1")
```

```
library(downscaleR)
```

Finally, `visualizeR` will be used to produced visualization plots for verification and communication of the undertainty/past performance of the forecasting system.

```
devtools::install_github("SantanderMetGroup/visualizeR@v0.2-0")
```

```
library(visualizeR)
```

Installing fireDanger

The package `fireDanger` is used for the calculation of the FWI System from the climate data structures provided by the packages above. It is thus seamlessly integrated with the Santander MetGroup R Climate Processing Bundle.

```
devtools::install_github("SantanderMetGroup/fireDanger@v1.0.2")
```

```
library(fireDanger)
```

Other packages

Package `easyVerification` (MeteoSwiss 2017) provides convenient functions for the verification of large ensemble forecast datasets, as the ones used in this paper. We use this package to calculate some of the verification measures used in the article (ROC area, ensemble correlation etc.). `easyVerification` is available on CRAN.

```
install.packages("easyVerification")
```

```
library(easyVerification)
```

Similarly, the `RColorBrewer` library is required in order to customize the FWI map palette according to the color scheme presented in Bedia et al (2015) and in the present paper.

```
install.packages("RColorBrewer")
```

```
library(RColorBrewer)
```

Accessing the User Data Gateway (UDG)

The User Data Gateway is the one stop shop for climate data access maintained by the Santander MetGroup. The UDG builds on the THREDDS Access Portal (UDG-TAP) which is the entry point for authentication and data access (information for registration). Authorization is organized in thematic groups, according to the data policies of different projects and international initiatives. There is also public data available that is open for everyone, for instance, the CFSv2 seasonal forecast and the WFDEI observational data used in this example. After registration, a username and password is provided to sign in.

Loading data using the R interface

`loadeR.ECOMS` is the R-package providing transparent access to different seasonal forecast products available through the UDG (see the complete Table of available datasets and variables).

Prior to accessing the data at UDG, the credentials need to be set. This can be done within R using the function `loginUDG`:

```
loginUDG(username = "myuser", password = "mypassword")
```

Note: loading seasonal forecast data at the daily temporal resolution required for FWI calculation is a time-consuming step. Loading times may vary greatly depending on several factors. This step may take several hours of data download. On the contrary, loading the observations (WFDEI) is very fast, due to the much greater simplicity of the dataset.

Loading seasonal forecast data

Next we are loading the variables needed for computing the FWI with function `loadECOMS`. For brevity, we will consider a smaller spatio-temporal domain studied in the paper. In this case, we will develop the example using the NCEP's CFS-v2 hindcast instead of ECMWF System4, as the former is a public product, while System4 has restricted access. A 27-year hindcast period for CFSv2 is considered (1983-2009).

A few arguments are used to unequivocally define the seasonal forecast request (type `?load.ECOMS` for details):

```
dataset <- "CFSv2_seasonal"
season <- 5:9
leadMonth <- 0
years <- 1983:2009
latLim <- c(35, 47)
lonLim <- c(-10, 30)
members <- 1:15
```

Note that in order to reproduce the paper results (based on System 4), the argument dataset should be set to `dataset = "System4_seasonal_15"`, leaving the rest of commands and instructions unchanged. Thus, the `loadeR.ECOMS` package has been conceived as "user-transparent", in the sense that the users do not need to worry about the technical complexities regarding member definition (see e.g. how CFS-v2 members have been defined) and/or the different variable naming and units. In addition, the `aggr.d` argument allows to perform time aggregation of sub-daily variables on-the-fly, as required for instance to calculate daily accumulated precipitation from the 6-hourly flux (CFSv2 model). For instantaneous variables, the argument `time` would be set to the corresponding verification time (e.g. `time="12"` for 12 UTC verifying records). This ensures the maximum reproducibility of the results, that may be altered to some extent depending on the various aggregation options and/or unit transformations.

In this example, we will use aggregated data to illustrate this capability. This would be the proxy-based approach for the calculation of FWI used for the generation of the state-of-the-art future FWI projections for Europe (Bedia et al 2014).

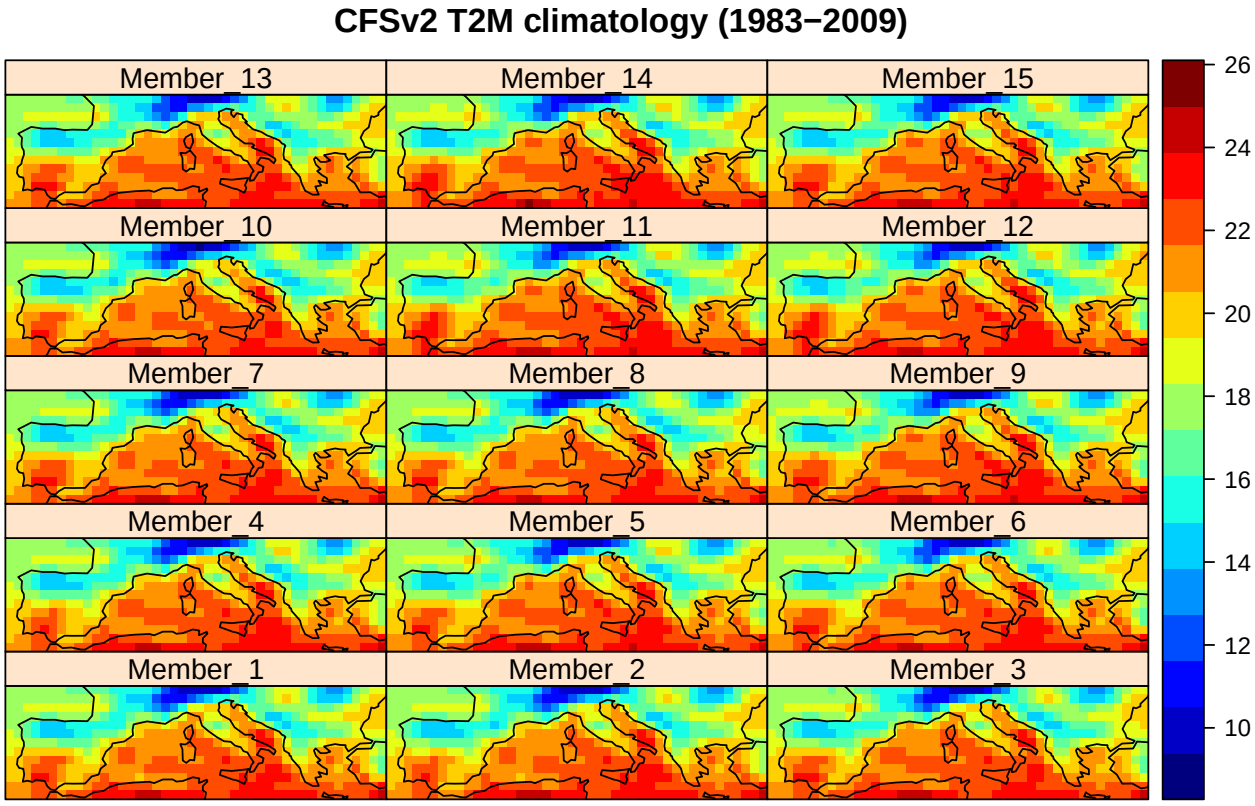
The argument `leadMonth` indicates the initialization time of the predictions. This argument and the argument `season` (in months, from 1 to 12), unequivocally define the initialization time and the corresponding verification times to be chosen. In addition, the argument `years` indicates the whole analysis period. So, in this particular case, we are loading the May initialization (`leadMonth = 0` and `season = 5:9`), since the starting day (1 May) until the end of the target season (30 September). Note that the fire season considered in the paper is June-September (JJAS), but the predictions for May are also considered in order to compute FWI a few weeks before the start of the season as a spin-up period for FWI stabilization (see Sec. 2.1 in the paper).

Finally, the arguments `lonLim` and `latLim` provide the longitudinal and latitudinal boundaries of the spatial domain, while the argument `members` indicates the number of ensemble members to be considered.

```
## Load temperature
Tm <- loadECOMS(dataset, var = "tas", members = members,
                latLim = latLim, lonLim = lonLim,
                season = season, years = years, leadMonth = leadMonth,
                time = "DD", aggr.d = "mean")
## Load relative humidity
H <- loadECOMS(dataset, var = "hurs", members = members,
               latLim = latLim, lonLim = lonLim,
               season = season, years = years, leadMonth = leadMonth,
               time = "DD", aggr.d = "min")
## Load precipitation
r <- loadECOMS(dataset, var = "tp", members = members,
               latLim = latLim,  lonLim = lonLim,
               season = season, years = years, leadMonth = leadMonth,
               time = "DD", aggr.d = "sum")
## Load wind speed
W <- loadECOMS(dataset, var = "wss", members = members,
               latLim = latLim, lonLim = lonLim,
               season = season, years = years, leadMonth = leadMonth,
               time = "DD", aggr.d = "mean")
```

The climatologies can be inspected with function `plotClimatology` and `climatology`. For example:

```
plotClimatology(climatology(Tm), backdrop.theme = "coastline",
                main = "CFSv2 T2M climatology (1983-2009)")
```



Note that neither relative humidity (`hurs`), nor wind speed (`wss`) are original variables produced by the CFSv2 model (see the table of available variables). These variables are computed on-the-fly by the `loadeR.ECOMS` R interface, greatly facilitating data processing to the user.

Also note that the standard units of `wss` are m/s. Thus, an intermediate step is needed to convert to Km/h, as required for FWI calculation. After this operation, the four input variables are "stacked" in a single `multigrid`, which is an special object containing several variables as an extra-dimension, that ensures their spatio-temporal consistency, either for downscaling, EOF/PCA analyses or, as in this case, to construct a FWI grid with `fireDanger`. We use to this aim the multigrid constructor `makeMultiGrid`:

```
## Convert wss units from m/s to km/h
W$Data <- W$Data*3.6
## Update "units" attribute
attr(W$Variable, "units") <- "km.h-1"
## make multigrid
multigrid_hind <- makeMultiGrid(Tm, H, r, W)
```

Loading Observations

The same operation is repeated to get the multigrid of the observations, in this case we will use the Watch Forcing Dataset based on ERA-Interim, as described in the paper (dataset = `WFDEI`). Note that in this case the arguments `leadMonth` and `members`, particular for predictions, do not apply. Also, because the WFDEI variables currently available are already daily aggregated, there is no need to specify particular time aggregation options, as the default will load the maximum temporal resolution.

As `loadeR.ECOMS` performs dataset homogenization, there is no need to change the variable nomenclature. In the case of relative humidity, we directly request minimum relative humidity (`var = "hursmin"`). Similarly, the variables loaded will have exactly the same units as with the CFSv2 dataset, and thus the conversion from m/s to km/h is needed again:

```
## Define the target dataset
dataset <- "WFDEI"

## Load temperature
Tm.obs <- loadECOMS(dataset = dataset, var = "tas",
                latLim = latLim, lonLim = lonLim,
                season = season, years = years)
## Load relative minimum humidity
H.obs <- loadECOMS(dataset = dataset, var = "hursmin",
               latLim = latLim, lonLim = lonLim,
               season = season, years = years)
## Load precipitation
r.obs <- loadECOMS(dataset = dataset, var = "tp",
               latLim = latLim,  lonLim = lonLim,
               season = season, years = years)
## Load wind speed
W.obs <- loadECOMS(dataset = dataset, var = "wss",
               latLim = latLim, lonLim = lonLim,
               season = season, years = years)
```

Conversion of standard units to km/h for windspeed (and metadata update):

```
W.obs$Data <- W.obs$Data*3.6
attr(W.obs$Variable, "units") <- "km.h-1"
```

And multigrid creation:

```
multigrid_obs <- makeMultiGrid(Tm.obs, H.obs, r.obs, W.obs)
```

Direct load of the climate data objects above (shortcut)

In order to facilitate data exploration, the seasonal forecast and observations data (`multrigrid_hind` and `multigrid_obs`) can be directly loaded into the R session through the following links:

```
load(url("http://www.meteo.unican.es/work/fireDanger/wiki/data/multigrid_hind.rda"))
load(url("http://www.meteo.unican.es/work/fireDanger/wiki/data/multigrid_obs.rda"))
```

FWI calculation

The function `fwiGrid` in package `fireDanger` is a wrapper of function `fwi` in the same package, that handles in a convenient way the model data structures ("grids") provided by the `loadeR` and `loadeR.ECOMS` packages. It operates on either ordinary grids (e.g. gridded observations, reanalysis...) or multi-member grids, as it is the case of seasonal forecast datasets. In the latter, FWI is computed member by member independently. The output is a grid containing the target FWI component, as well as all the metadata (dates, attributes etc.).

There are several arguments controlling the different options and output values. If these are not provided, the default FWI is computed. One optional (yet important) argument is the argument `mask`. It is a land mask that avoids FWI calculation on the sea. In this case, WFDEI is a land dataset, and therefore the argument can be omitted (default to `NULL`). However, in the case of CFSv2, it is highly advisable to use the model land-mask.

Calculating observed (WFDEI) FWI

```
obs <- fwiGrid(multigrid = multigrid_obs)
```

Seasonal Forecast Hindcast

In this example, the CFSv2 model mask is used to avoid sea surface when computing the FWI. The mask is loaded by requesting the variable "lm" (land mask) in function `loadECOMS`. To ensure spatial domain consistence between object `multigrid_hind` and the mask we interpolate `cfs_mask` to the `multigrid_hind` grid with functions `interpGrid` and `getGrid`.

```
cfs_mask <- loadECOMS(dataset, var = "lm", latLim = latLim, lonLim = lonLim)
mask <- interpGrid(cfs_mask, getGrid(multigrid_hind))
```

The object `mask` is passed to the argument of the same name:

```
hindcast <- fwiGrid(multigrid = multigrid_hind, mask = mask)
```

Adjusting the fire season

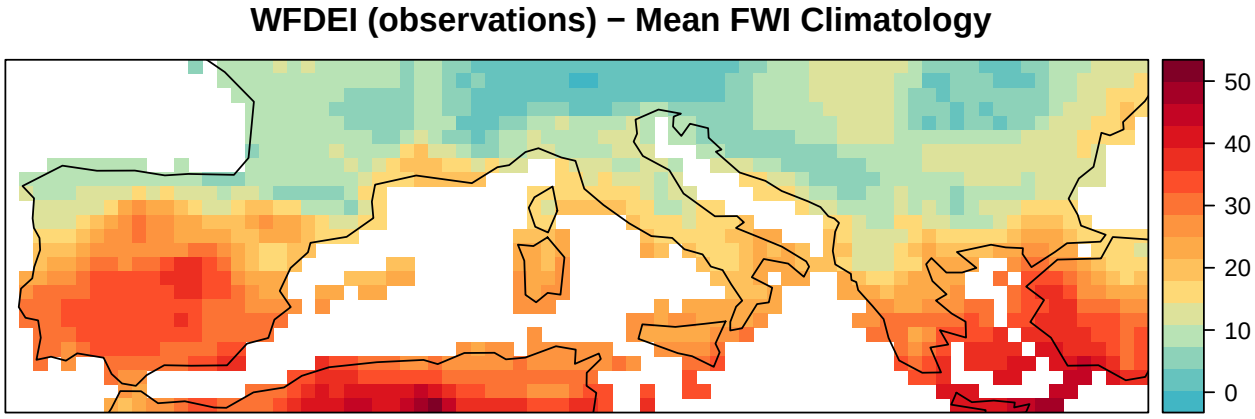
As indicated in the paper, a spin-up period of 1 month is used to calculate FWI. Now, it is time to remove this first month (May), to retain the data for the fire season (JJAS). This can be done using the `subsetGrid` function from `downscaleR`, allowing flexible grid subsetting along selected dimensions:

```
hindcastJJAS <- subsetGrid(hindcast, season = 6:9)
obsJJAS <- subsetGrid(obs, season = 6:9)
```

The mean FWI climatology (1983-2009) is mapped next. Note the use of the library `RColorBrewer` for the definition of a suitable color palette for FWI.

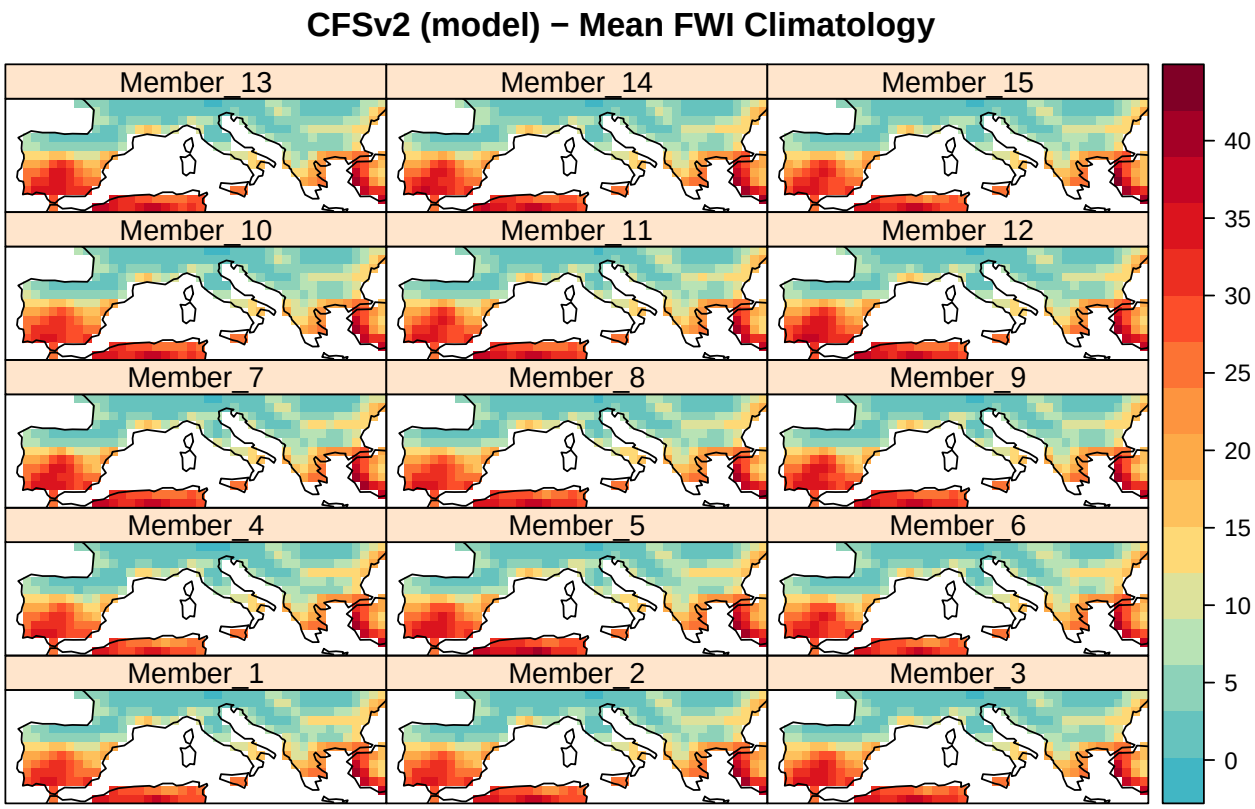
```
fwi.colors <- colorRampPalette(c(rev(brewer.pal(9, "YlGnBu")[3:5]),
                                 brewer.pal(9, "YlOrRd")[3:9]))

plotClimatology(climatology(obsJJAS),
                backdrop.theme = "coastline",
                col.regions = fwi.colors,
                main = "WFDEI (observations) - Mean FWI Climatology")
```



and the forecast data.

```
plotClimatology(climatology(hindcastJJAS),
                backdrop.theme = "coastline",
                col.regions = fwi.colors,
                main = "CFSv2 (model) - Mean FWI Climatology")
```

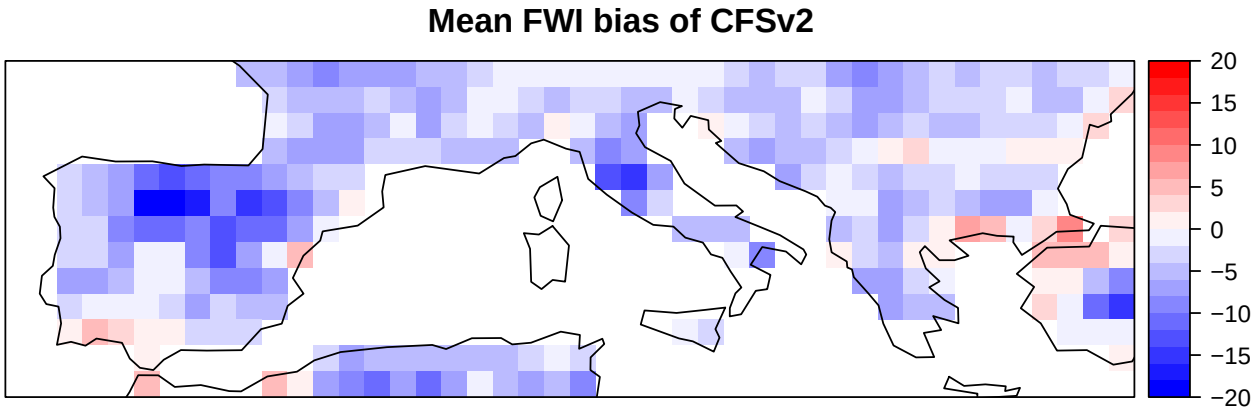


To calculate the mean CFS bias with respect to WFDEI we will perform the multi-member mean of the forecast with function `aggregateGrid`. Then, we will interpolate the observed FWI to the CFS's grid, to operate on the same spatial data, enabling the bias calculation. This is easily done with functions `interpGrid` and `getGrid`. The bias is assigned to a new grid object ("bias").

```
hindcast_aggr <- aggregateGrid(hindcastJJAS, aggr.mem = list(FUN = mean))
bias <- hindcast_aggr
obsJJAS_int <- interpGrid(grid = obsJJAS, new.coordinates = getGrid(hindcast_aggr))
bias$Data <- hindcast_aggr$Data - obsJJAS_int$Data
```

The mean bias can be visualized applying funcions `plotClimatology` and `climatology`. The last will compute the mean of the time-period by default.

```
bias.colors <- colorRampPalette(c("blue", "white", "red"))
plotClimatology(climatology(bias),
                backdrop.theme = "coastline",
                col.regions = bias.colors,
                at = seq(-20,20,2),
                main = "Mean FWI bias of CFSv2")
```



Bias Correction

In the `downscaleR` package, the user can find the standard bias correction techniques used in the literature (scaling factors and quantile mapping) as well as other recently published extensions of these techniques (e.g. the multi-variable bias-correction ISI-MIP method, Hempel et al. 2013). The bias correction methods implemented in `downscaleR` are included in the functions `biasCorrection` and `isimip`. For more information, visit the wiki of downscaleR.

Empirical quantile mapping approach:

```
hindcast_bc <- biasCorrection(y = obsJJAS, x = hindcastJJAS,
                              newdata = hindcastJJAS, method = "eqm")
```

Calculating FWI climatologies

Different commonplace seasonal FWI-derived indices (see e.g. Bedia et al 2014 for a description of the FWI-derived indices used in this article and others) can be flexibly calculated using `aggregateGrid`. Here, data are annually aggregated, using any user-defined function. `aggregateGrid` is a versatile function that can perform aggregations on any of the dimensions of the data array (members, longitude and/or latitude or different time intervals). In this case, to specify aggregations that are performed on an annual basis, the argument `aggr.y` is used. Some examples of user-defined aggregation functions are given in the next subsections:

Mean seasonal FWI

The mean seasonal FWI value is directly calculated using the R base function `mean`. Functions are passed to `aggregateGrid` using a named list. The firs element (`FUN`) is the name of the function, and then further arguments passed to the aggregation function can be indicated.

```
fwimean_obs <- aggregateGrid(obsJJAS, aggr.y = list(FUN = "mean", na.rm = TRUE))
```

The same procedure is applied to the hindcast:

```
# raw data
fwimean_hind_raw <- aggregateGrid(hindcast, aggr.y = list(FUN = "mean", na.rm = TRUE))
```

```
# bias-corrected data
fwimean_hind <- aggregateGrid(hindcast_bc, aggr.y = list(FUN = "mean", na.rm = TRUE))
```

FWI90

FWI 90 is defined as the seasonal 90th percentile of FWI. In this case, percentiles are directly computed using the R base funtion `quantile`:

```
fwi90_obs <- aggregateGrid(obs, aggr.y = list(FUN = "quantile", probs = .9, na.rm = TRUE))
```

FOT30

Another frequently used index in the Mediterranean is FOT30, defined as the frequency of days with a FWI above 30. In this case, the user can first define the function to compute FOT30, which is then passed to `aggregateGrid`:

```
fot30 <- function(x) {
      sum(x > 30, na.rm = TRUE) / length(na.omit(x))
}

fwi30_obs <- aggregateGrid(obs, aggr.y = list(FUN = "fot30"))
```

In the following we will use only the `fwimean` climatology.

Forecast verification

Before calculating the skill, we correct the trend in the data with function `detrendGrid`, as indicated in the paper.

```
fwimean_obs_detrend <- detrendGrid(fwimean_obs)
fwimean_hind_detrend <- detrendGrid(fwimean_hind)
```

We use function `veriApply` of package `easyVerification` to compute the different verification measures. In this example we will compute the Area under the ROC Curve (`verifun = "EnsRoca"`). In the manuscript, the bias-corrected and the raw FWI of the forecasts are validated (objects `fwimean_hind` and `fwimean_hind_raw` in this tutorial). For brevity, we will only consider the bias-corrected data in the following, this is, object `fwimean_hind`.

In this example we will extract the Area under the ROC Curve correspoding to the upper-tercile predictions (predicted above-normal seasonal FWI), for which the threshold is set in 2/3 (argument `prob`). This way two FWI categories are obained (i.e. above-normal and others). Further details can be found in the helps of the function `veriApply`. The data arrays named `"Data"` within the hindcast and observation objects can be directly passedd to veriApply. It is only necessary to specify what is the index position of ensemble members (dimension 1 of the forecast array) and time (dimension 2 of the forecast array):

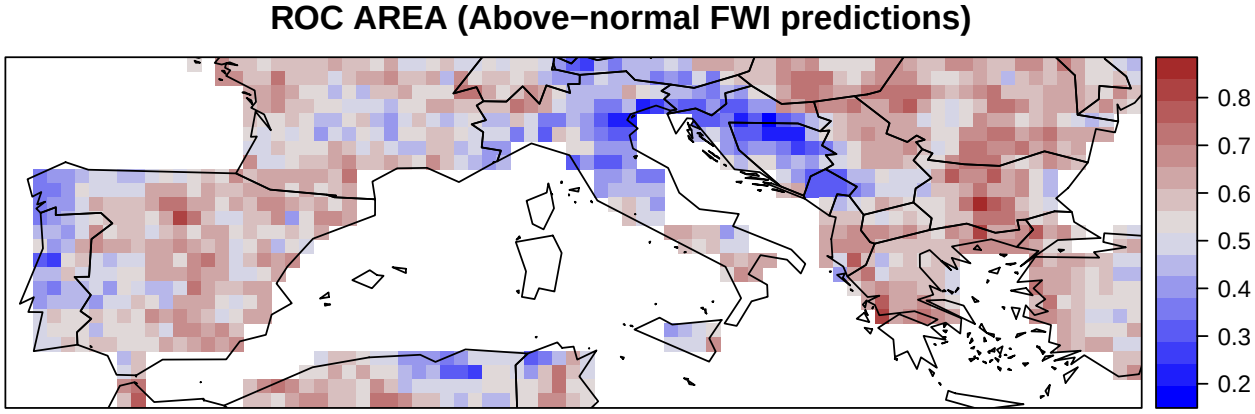
```
skill <- veriApply(verifun = "EnsRoca",
                   fcst = fwimean_hind_detrend$Data,
                   obs = fwimean_obs_detrend$Data,
                   prob = 2/3,
                   ensdim = 1,
                   tdim = 2)
```

The verification output is then converted to a suitable format, compatible with the R climate postprocessing bundle, using this `transformeR` function:

```
upper.tercile <- easyVeri2grid(easyVeri.mat = skill$cat2,
                               obs.grid = fwimean_obs_detrend,
                               verifun = "EnsRoca")
```

Note that `skill$cat2` corresponds to the second category, this is the upper-tercile (above-normal). The ROC Area map, similar to figure 3 in the manuscript, is obtained using the `plotClimatology` function. Note that `plotClimatology` is just a wrapper for the powerful `spplot` function from the `sp` package, and it is therefore possible to customize the plots. Also note the introduction of a red-grey-blue palette.

```
ms.colors <- colorRampPalette(c("blue", "grey90", "brown"))
plotClimatology(upper.tercile,
                backdrop.theme = "countries",
                main = "ROC AREA (Above-normal FWI predictions)",
                col.regions = ms.colors)
```



Visualization

The `visualizeR` package is aimed at the visual communication of the uncertainty in seasonal forecasts, providing tools that incorporate different verification methods. In the following sections, we illustrate two complementary visualization examples involving functions `tercilePlot` and `bublePlot`. Further worked examples on the use and installation of `visualizeR` can be found in the corresponding gitHub repository: https://github.com/SantanderMetGroup/visualizeR

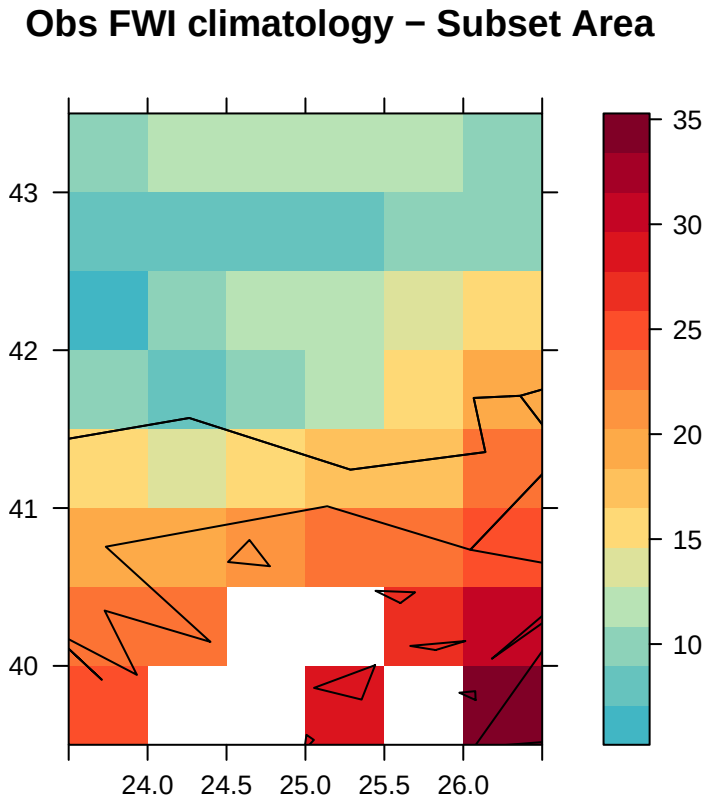
```
library(visualizeR)
```

Tercile validation

In order to gain a better insight into the areas of high skill (sec. 3.3 in the manuscript), next we will spatially subset the data with function `subsetGrid`. The extracted region encompasses grid boxes over Greece and Bulgaria.

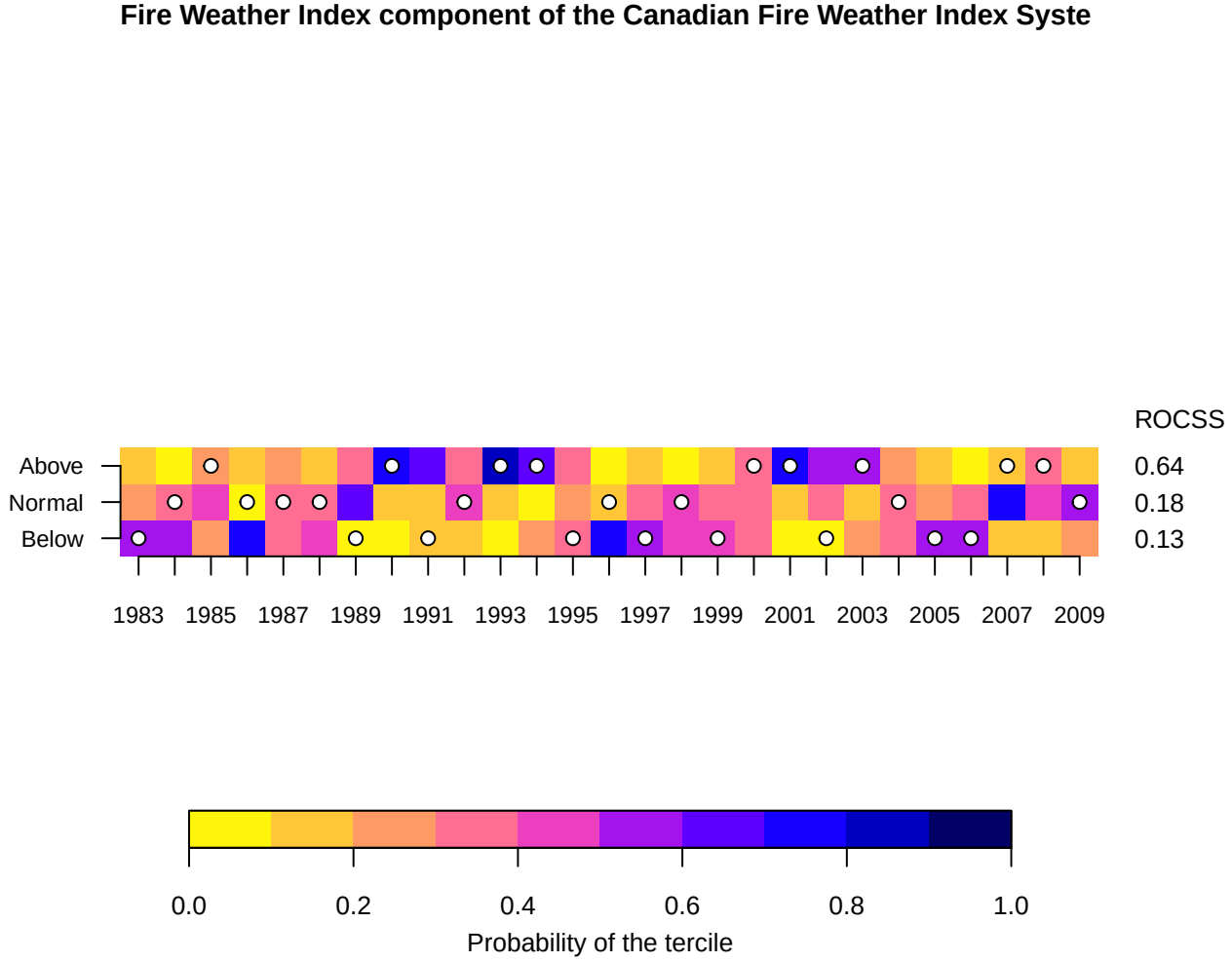
```
fwimean_hind_sub <- subsetGrid(fwimean_hind_detrend,
                               latLim = c(40,43.5), lonLim = c(24,26.5))
fwimean_obs_sub <- subsetGrid(fwimean_obs_detrend,
                              latLim = c(40,43.5), lonLim = c(24,26.5))
plotClimatology(climatology(fwimean_obs_sub),
```

```
                backdrop.theme = "countries",
                col.regions = fwi.colors,
                scales = list(draw = TRUE, x = list(alternating = FALSE, tick.number = 6)),
                main = "Obs FWI climatology - Subset Area")
```



The tercile plot is obtained by function `tercileValidation`. Terciles are arranged by rows. Further details for graph interpretation are given in Sec. 2.5.2. of the manuscript.

```
tercilePlot(fwimean_hind_sub, obs = fwimean_obs_sub, color.pal = "ypb")
```

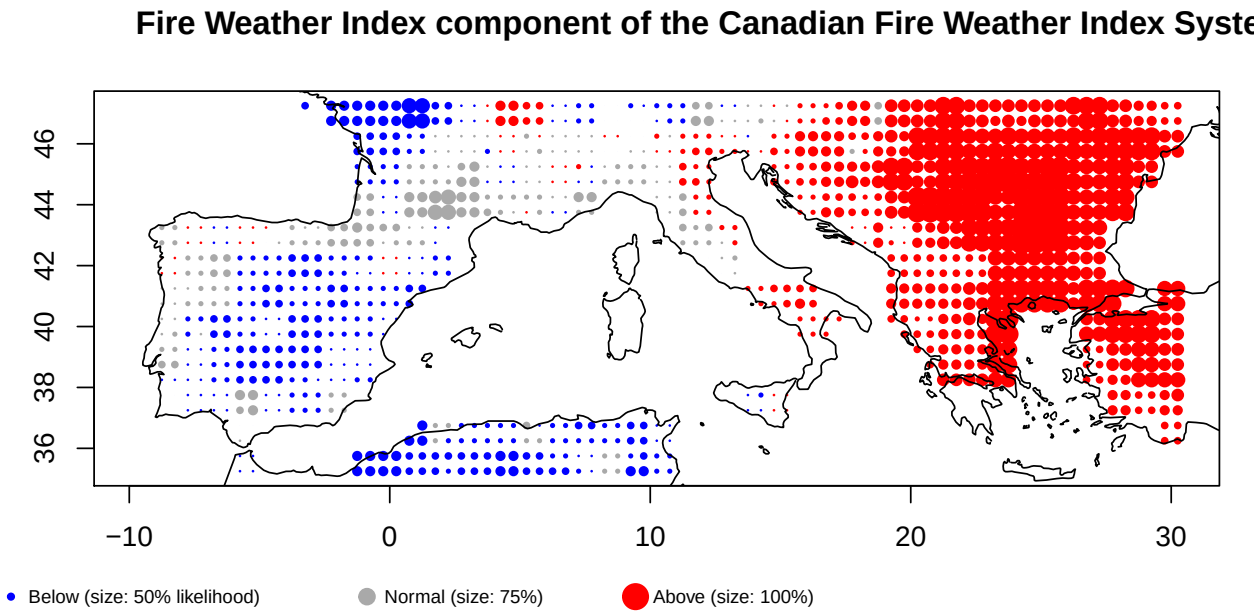


Bubble map

In this example, we present a bubble map, showing in the same plot the forecast prediction for a particular year and the overall performance of the forecasting system. In this example, we choose the year 2001, a year of above-normal FWI conditions in SE Europe that were notably well predicted by the model (see the tercile plot above).

Bubble maps are implemented by the function `bubblePlot`, that combines in a geographic map several aspects of the seasonal forecast system using three properties of the bubbles: colour, size and transparency. Type `?bubblePlot` for details.

```
bubblePlot(hindcast = fwimean_hind, obs = fwimean_obs,
           year.target = 2001,
           bubble.size = 3,
           size.as.probability = TRUE,
           score = FALSE)
```



References

- Bedia, J., Herrera, S., Camia, A., Moreno, J.M., Gutierrez, J.M., 2014. Forest Fire Danger Projections in the Mediterranean using ENSEMBLES Regional Climate Change Scenarios. Clim Chang 122, 185–199. doi:10.1007/s10584-013-1005-z
- Bedia, J., Herrera, S., Gutierrez, J.M., Benali, A., Brands, S., Mota, B., Moreno, J.M., 2015. Global patterns in the sensitivity of burned area to fire-weather: implications for climate change. Agr. Forest Meteorol. 214–215, 369–379. doi:10.1016/j.agrformet.2015.09.002
- Saha, S., Moorthi, S., Wu, X., Wang, J., Nadiga, S., Tripp, P., Behringer, D., Hou, Y.-T., Chuang, H., Iredell, M., Ek, M., Meng, J., Yang, R., Peña Mendez, M., van den Dool, H., Zhang, Q., Wang, W., Chen, M., Becker, E., 2013. The NCEP Climate Forecast System Version 2. J Clim 130925135638001. doi:10.1175/JCLI-D-12-00823.1
- van Wagner, C.E., 1987. Development and structure of the Canadian Forest Fire Weather Index (Forestry Tech. Rep. No. 35). Canadian Forestry Service, Ottawa, Canada.
- Weedon, G.P., Balsamo, G., Bellouin, N., Gomes, S., Best, M.J., Viterbo, P., 2014. The WFDEI meteorological forcing data set: WATCH Forcing Data methodology applied to ERA-Interim reanalysis data. Water Resour. Res. 50, 7505–7514. doi:10.1002/2014WR015638

Session Information

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/lapack/liblapack.so.3.0
##
```

```
## 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:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2      easyVerification_0.4.1
##  [3] SpecsVerification_0.5-2 fireDanger_1.0.2
##  [5] visualizeR_0.2-0        sm_2.2-5.4
##  [7] downscaleR_2.0-1        transformeR_0.0.8
##  [9] loadeR.ECOMS_1.2.3      loadeR_1.0.9
## [11] loadeR.java_1.1-0       rJava_0.9-8
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.10        plyr_1.8.4          compiler_3.4.0
##  [4] bitops_1.0-6        tools_3.4.0         vioplot_0.2
##  [7] boot_1.3-19         digest_0.6.12       evd_2.3-2
## [10] evaluate_0.10       lattice_0.20-35     Matrix_1.2-8
## [13] yaml_2.1.14         parallel_3.4.0      spam_1.4-0
## [16] akima_0.6-2         stringr_1.2.0       knitr_1.15.1
## [19] mapplots_1.5        fields_8.10         maps_3.1.1
## [22] rprojroot_1.2       grid_3.4.0          dtw_1.18-1
## [25] pbapply_1.3-2       rmarkdown_1.5       sp_1.2-4
## [28] magrittr_1.5        scales_0.4.1        backports_1.0.5
## [31] htmltools_0.3.5     CircStats_0.2-4     MASS_7.3-47
## [34] abind_1.4-5         colorspace_1.3-2    proxy_0.4-17
## [37] stringi_1.1.5       munsell_0.4.3       RCurl_1.95-4.8
## [40] verification_1.42   RcppEigen_0.3.2.9.1
```