Verification of seasonal forecasts from the ECOMS User Data Gateway: a worked example

Joaquín Bedia[1] and Jonas Bhend[2]

[1]Santander Met Group. University of Cantabria - CSIC (Spain)
[2]Federal Office of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland

2015-02-20

Abstract

This document provides a worked example on how to download forecast and validation data from the ECOMS-User Data Gateway and perform forecast validation using the tools integrated in the R package `easyVerification`. All the steps are undertaken within the R environment. Access to ECOMS-UDG is done via the R package `ecomsUDG.Raccess`, serving as a user-friendly interface to the ECOMS-UDG by exploiting OpenDAP and other remote service protocols. It is not the aim of this tutorial to provide a comprehensive description of the `easyVerification` or the `ecomsUDG.Raccess` packages. More detailed information on the latter can be obtained at the ECOMS-UDG wiki.

Contents

1 Obtaining data from the ECOMS-UDG

1.1 Access authorization

As different terms of use and policies apply to the different datasets stored in the ECOMS User Data Gateway, a fine-grained user authorization scheme has been implemented using the THREDDS Administration Panel (TAP), which allows user registration and data access authorization for the ECOMS partners (EUPORIAS, SPECS and NACLIM projects), for which this document is initially conceived.

In all cases, dataset access is conditioned to the acceptance of the particular usage terms and conditions. Further instructions on TAP registration are provided here.

1.2 Installing the `ecomsUDG.Raccess` package

Package `ecomsUDG.Raccess` is available in a public GitHub repository. The recommended installation procedure is to use the `install_github` command from the `devtools` R package. More details on the package installation process, as well as some advice regarding the most common installation problems can be found at the ECOMS-UDG wiki.

```
devtools::install_github(c("SantanderMetGroup/downscaleR.java@stable",
                           "SantanderMetGroup/downscaleR@stable",
                           "SantanderMetGroup/ecomsUDG.Raccess@stable"))
```

Note that apart from the `ecomsUDG.Raccess` package, two more dependencies are installed. `downscaleR.java` is internally used, containing the netCDF Java API used by the other packages, but the user never calls explicitly to any of its functions. Furthermore, the package `downscaleR` has some utilities internally used by the `ecomsUDG.Raccess` package, plus some other utilities that we will explicitly use during the next examples (plotting, interpolation...).

1.3 Accessing data

Once a valid username and a password are obtained from the TAP, providing authorization for the required datasets, login can be performed from R using the `loginECOMS_UDG` function (see the `help(loginECOMS_UDG)` for details in case of proxy connections).

```
library(ecomsUDG.Raccess)
```

```
## Loading required package: downscaleR.java
## Loading required package: rJava
## NetCDF Java Library v4.3.22 (27 May 2014) loaded and ready
## Loading required package: downscaleR
## Loading required package: maps
## downscaleR version 0.5-2 (22-Jan-2015) is loaded
## ecomsUDG.Raccess version 2.2-5 (06 Sep 2014) is loaded
## WARNING: Your current version of ecomsUDG.Raccess is not up-to-date
## Get the latest stable version 2.2.6 at http://meteo.unican.es/ecoms-udg/RPackage#Versions
```

```
loginECOMS_UDG(username = "myUser", password = "999a")
```

Data download is straightforward using the `loadECOMS` function. Argument names are intuitive, and the function has been conceived thinking in the needs of the downscaling and impact research communities, being simple to obtain spatio-temporal slices for particular spatial domains, forecast times, seasons and years.

Suppose we are interested obtaining data from the ECMWF's System4 seasonal forecasting model of 15 members[1] (`dataset = "System4_seasonal_15"`). In particular, in this example we will retrieve maximum daily surface temperature (`var = "tasmax"`) for boreal summer (JJA, `season = 6:8`), for a rectangular domain centered on the Iberian Peninsula and France (`lonLim = c(-10,15)` and `latLim = c(35,50)`), for the period 1981-2000 (`years = 1981:2000`), and considering the first 9 ensemble members (`members = 1:9`) and a lead-month 2 forecast[2] (`leadMonth = 2`). We will illustrate the verification of these data against the observational gridded datasets WATCH Forcing Dataset-ERA-Interim (WFDEI, `dataset = "WFDEI"`), also available via the ECOMS-UDG.

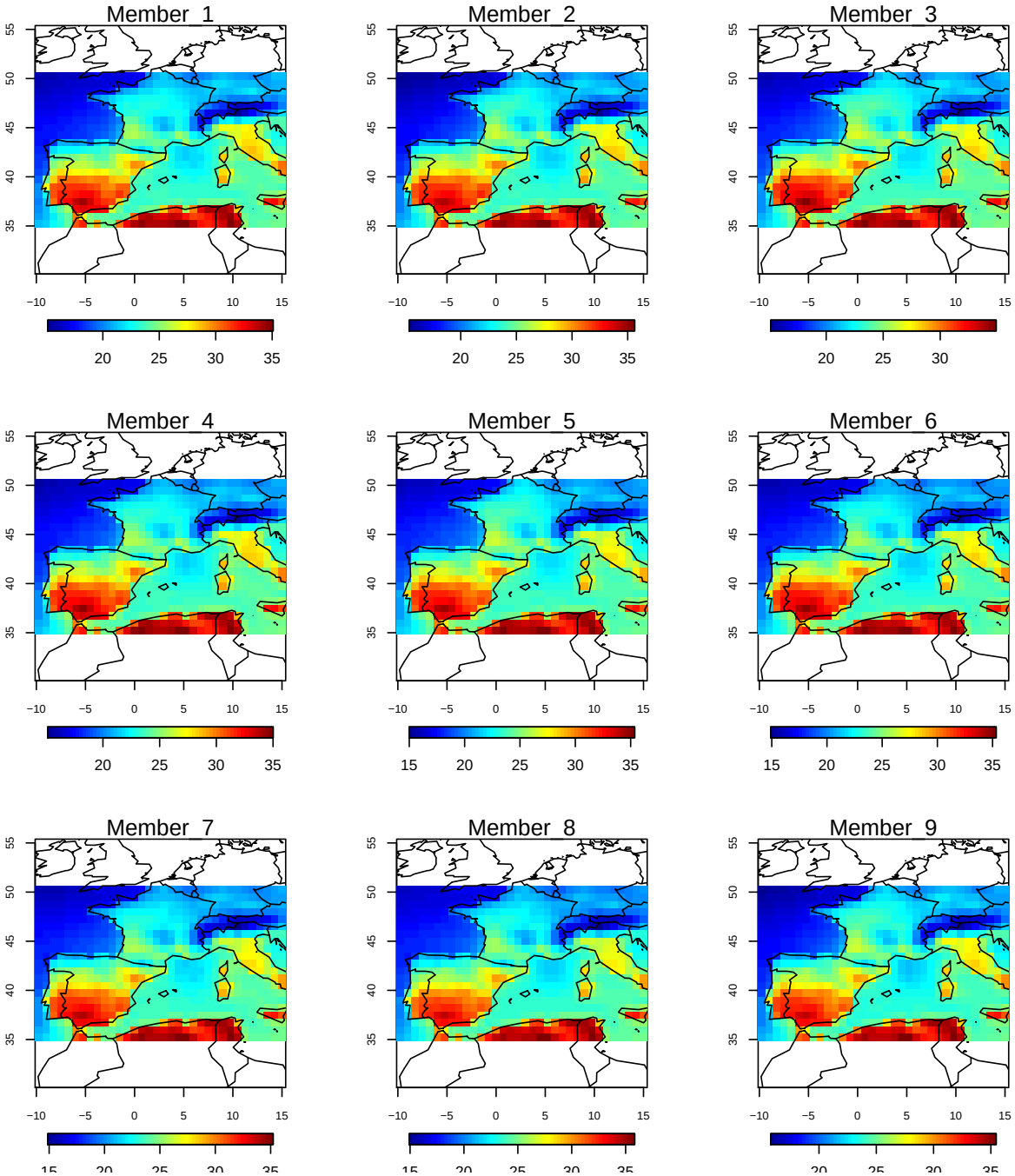
```
tx.forecast <- loadECOMS(dataset = "System4_seasonal_15",
                         var = "tasmax",
                         members = 1:9,
                         lonLim = c(-10 ,15),
                         latLim = c(35, 50),
                         season = 6:8,
                         years = 1991:2000,
                         leadMonth = 2)

## [2015-01-19 14:54:01] Defining homogeneization parameters for variable "tasmax"
## [2015-01-19 14:54:01] Defining geo-location parameters
## [2015-01-19 14:54:01] Defining initialization time parameters
## [2015-01-19 14:54:05] Retrieving data subset ...
## [2015-01-19 14:55:09] Done
```

The data loaded can be quickly inspected using the `plotMeanField` tool[3]. We can plot a map of the temporal mean (mean maximum temperature summer 1991-2000) either as a multi-member mean (argument `multimember = FALSE`), or considering each member sepparately (`multimember = TRUE`), as in this example:

```
plotMeanField(tx.forecast, multi.member = TRUE)
```

[1] More specific worked examples for different cases are presented in the examples section of the ECOMS wiki.
[2] See this link details on the configuration of this particular dataset.
[3] Loading times may differ significantly depending on various factors. More details here



Similarly, we next load the reference observation data from the gridded dataset WFDEI. Note that in this case, the argument referring to the forecast time (`leadMonth`), is omitted, as the data loaded is not a forecast.

```
tx.obs <- loadECOMS(dataset = "WFDEI",
                    var = "tasmax",
                    lonLim = c(-10 ,15),
                    latLim = c(35, 50),
```

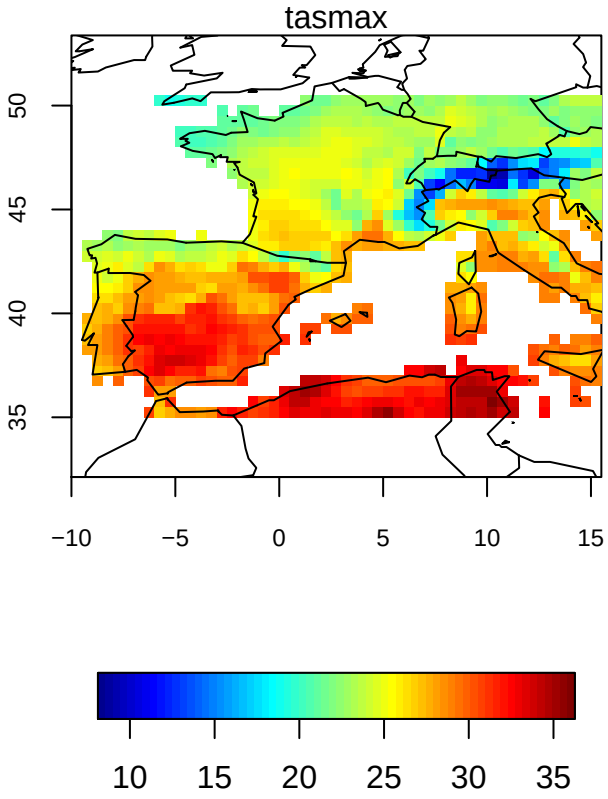


Figure 1: Summer (JJA) mean daily maximum temperature from the WFDEI data set for 1991-2000

```
                    season = 6:8,
                    years = 1991:2000)

## [2015-01-19 14:59:03] Defining homogeneization parameters for variable "tasmax"
## [2015-01-19 14:59:03] Defining geo-location parameters
## [2015-01-19 14:59:03] Defining time selection parameters
## [2015-01-19 14:59:04] Retrieving data subset ...
## [2015-01-19 14:59:05] Done
```

This is the map (temporal mean) of the observed reference data we want to use for validation:

```
plotMeanField(tx.obs)
```

2 Forecast validation

2.1 Installing the easyVerification package

Similar to installing the ecomsUDG.Raccess, you can install the easyVerification package providing verification scores and the functionality to compute these scores on large datasets. To install easyVerification please install the SpecsVerification package from CRAN first to make use of the validation metrics supplied with SpecsVerification.

```
install.packages("SpecsVerification")
devtools::install_github("MeteoSwiss/easyVerification", build_vignettes=TRUE)
```

To find out more on the functionality in the easyVerification package, please read the help to the functions, the vignette with vignette("easyVerification") or the README on GitHub.

2.2 Mean bias

After installing and loading the easyVerification package, we compute a few forecast scores and skill scores to illustrate how easyVerification can be used with seasonal forecast data downloaded from the ECOMS-UDG. To be able to validate the forecasts, we first have to interpolate either the forecasts or the observations to have the data on a common grid. We interpolate the observations to the grid of the forecasts using bilinear interpolation.

```
tx.obsintp <- interpGridData(tx.obs,
                             new.grid=getGrid(tx.forecast),
                             method="bilinear")
```

```
## [2015-02-20 11:20:24] Performing bilinear interpolation... may take a while
## [2015-02-20 11:20:39] Done
```

Before we go on, we compute 3-monthly averages of the forecasts and observations for validation of seasonal average daily maximum temperature.

```
annualmean <- function(x){
  xout <- x
  is.time <- which(attr(x$Data, 'dimensions') == 'time')
  isnot.time <- which(attr(x$Data, 'dimensions') != 'time')
  years <- getYearsAsINDEX(x)
  xout$Data <- aperm(apply(x$Data, isnot.time, tapply, years, mean),
                     seq(along=dim(x$Data))[c(is.time, isnot.time)])
  xout$Dates$start <- tapply(x$Dates$start, years, min)
  xout$Dates$end <- tapply(x$Dates$end, years, max)
  return(xout)
}

mn.tx.forecast <- annualmean(tx.forecast)
mn.tx.obsintp <- annualmean(tx.obsintp)
```

Now we're ready to compute validation scores on the 3-monthly mean daily maximum temperature forecasts. First we start by analysing the mean bias of the forecasts.

```
suppressPackageStartupMessages({
  library(easyVerification)
  library(fields)
})

bias <- veriApply("EnsMe",
                  fcst=mn.tx.forecast$Data,
                  obs=mn.tx.obsintp$Data,
                  ensdim=1, tdim=2)

bias.breaks <- pretty(c(bias, -bias), 50)
ncols <- length(bias.breaks) - 1
bias.col <- c(hcl(240, l=seq(20,99,length=ncols/2), c=seq(70,30, length=ncols/2)),
              hcl(10, l=seq(99,20,length=ncols/2), c=seq(30,70,length=ncols/2)))
```

```
image.plot(mn.tx.obsintp$xyCoords$x,
           mn.tx.obsintp$xyCoords$y,
           t(bias),
           breaks=bias.breaks, col=bias.col,
           xlab='longitude', ylab='latitude', las=1)
map(add=T, lwd=2, col='darkgrey')
```

2.3 Correlation

Next, we compute the correlation between the ensemble mean and verifying observations.

```
corr <- veriApply("EnsCorr",
                  fcst=mn.tx.forecast$Data,
                  obs=mn.tx.obsintp$Data,
                  ensdim=1, tdim=2)

corr.breaks <- seq(-1,1,0.05)
ncols <- length(corr.breaks) - 1
corr.col <- c(hcl(240, l=seq(20,99,length=ncols/2), c=seq(70,30, length=ncols/2)),
              hcl(10, l=seq(99,20,length=ncols/2), c=seq(30,70,length=ncols/2)))
```

```
image.plot(mn.tx.obsintp$xyCoords$x,
           mn.tx.obsintp$xyCoords$y,
           t(corr),
           breaks=corr.breaks, col=corr.col,
           xlab='longitude', ylab='latitude', las=1)
map(add=T, lwd=2, col='darkgrey')
box()
```

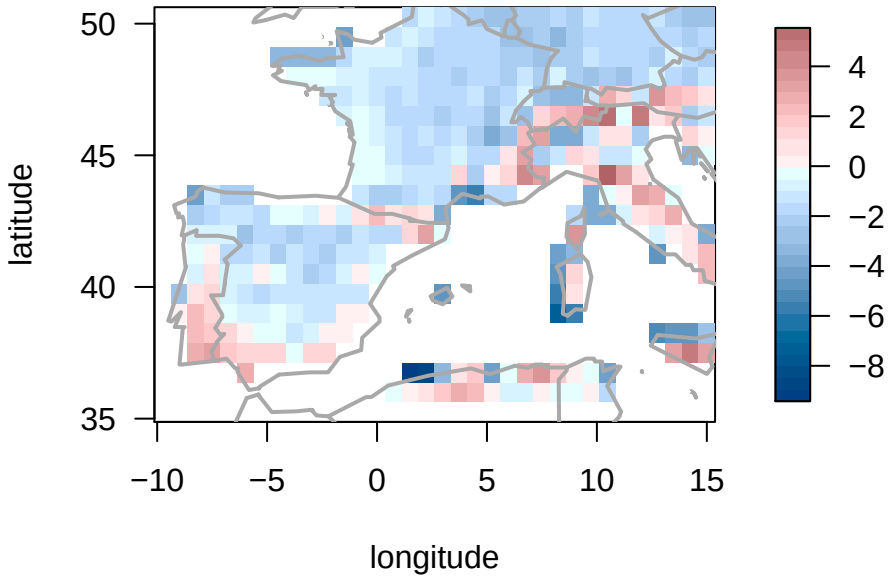


Figure 2: Mean bias of the summer (JJA) forecasts from ECMWF System4 initialized in May against the verifying observations from WFDEI for 1991-2000.

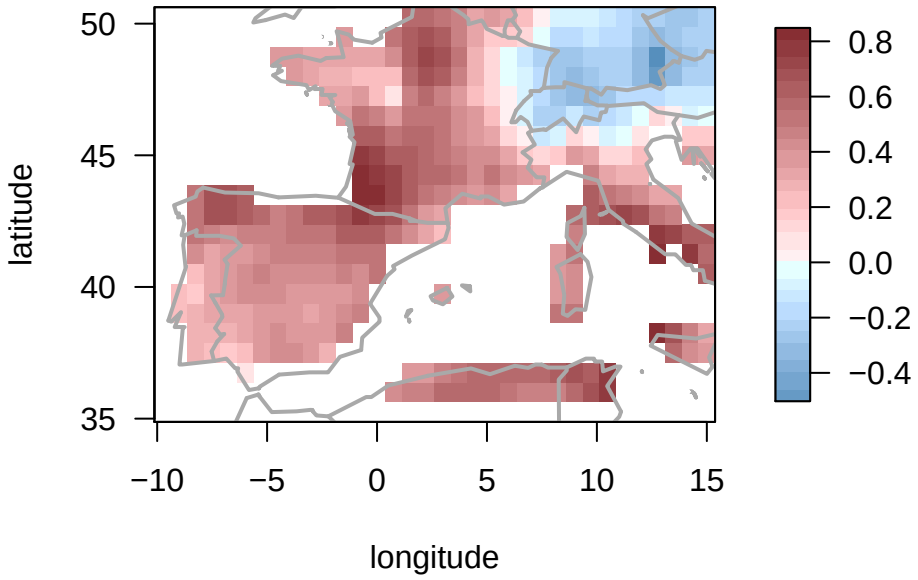


Figure 3: Corelation of the summer (JJA) ensemble mean forecast from ECMWF System4 initialized in May against the verifying observations from WFDEI for 1991-2000.

We find that the ensemble mean summer forecasts for 1991-2000 correlate well with the veryfing observations over most of the western Mediterranean, but forecasts do not skillfully represent year-to-year variability over Switzerland, Austria and southern Germany.

2.4 Generalized discrimination score

We continue with a probabilistic forecast verification metric that is a measure of how well the forecasts are able to discriminate between varying observations. This generalized discrimination score (also referred to as the two alternatives forced choice score) has been introduced by Mason and Weigel (2009). In `easyVerification` the generalized discrimination score for continuous ensemble forecasts is included as described in Weigel and Mason (2011).

```
gds <- veriApply("Ens2AFC",
                 fcst=mn.tx.forecast$Data,
                 obs=mn.tx.obsintp$Data,
                 ensdim=1, tdim=2)
```

This score ranges from 0 to 1, where 1 denotes perfect resolution, and 0.5 is no resolution.

```
image.plot(mn.tx.obsintp$xyCoords$x,
           mn.tx.obsintp$xyCoords$y,
           t(gds),
           breaks=corr.breaks/2+0.5, col=corr.col,
           xlab='longitude', ylab='latitude', las=1)
map(add=T, lwd=2, col='darkgrey')
box()
```

Again, we find that the System4 forecasts for summer maximum temperature are more skillful in the south-western part of the domain and one is better of using climatological forecasts rather than the System4 forecasts in the north-eastern part of the domain.

2.5 Ranked probability score

The final skill score we analyse is the ranked probability skill score (RPSS). Here we use the ranked probability skill score for tercile forecasts, that is probability forecasts for the three categories colder than average, average, and warmer than average. In order to convert observations and forecast in probabilities for the three categories, we have to add an additional argument `prob` to the `veriApply` function with the quantile boundaries for the categories as shown below.

```
rpss <- veriApply("EnsRpss",
                  fcst=mn.tx.forecast$Data,
                  obs=mn.tx.obsintp$Data,
                  prob=c(1/3,2/3),
                  ensdim=1, tdim=2)
```

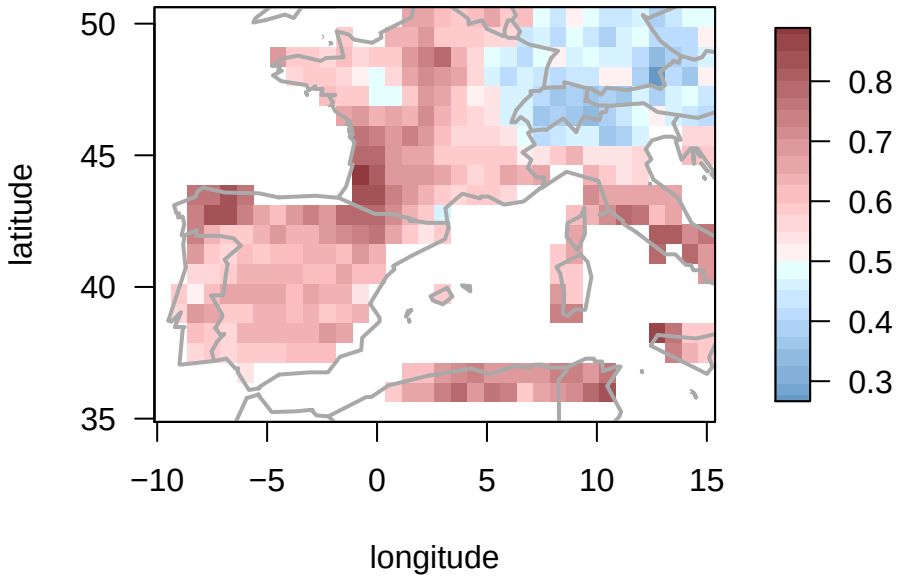


Figure 4: The generalized discrimination score for summer (JJA) ECMWF System4 forecasts initialized in May against the verifying observations from WFDEI for 1991-2000.

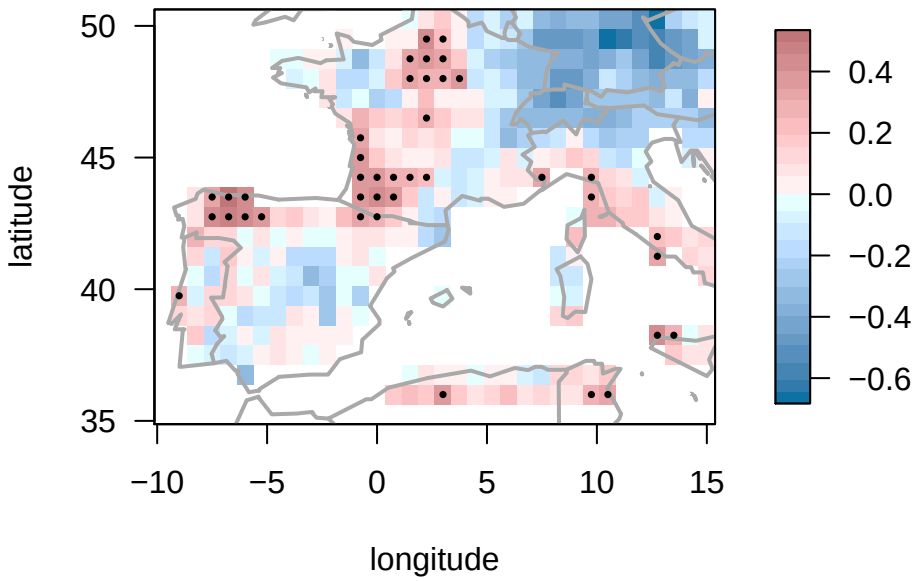


Figure 5: The RPSS for summer (JJA) ECMWF System4 forecasts initialized in May against the verifying observations from WFDEI for 1991-2000. Stippling indicates locations where the RPSS is significantly larger than zero at the 5 percent level.

Please note that along with the RPSS, the function `EnsRpss` (and thus `veriApply` as well) also returns the standard error of the RPSS. This allows us to mark where the forecasts have significant skill.

```
image.plot(mn.tx.obsintp$xyCoords$x,
           mn.tx.obsintp$xyCoords$y,
           t(rpss$rpss),
           breaks=corr.breaks, col=corr.col,
           xlab='longitude', ylab='latitude', las=1)
map(add=T, lwd=2, col='darkgrey')
sig.i <- rpss$rpss > rpss$rpss.sigma*qnorm(0.95)
lons <- rep(mn.tx.obsintp$xyCoords$x, each=length(mn.tx.obsintp$xyCoords$y))
lats <- rep(mn.tx.obsintp$xyCoords$y, length(mn.tx.obsintp$xyCoords$x))
points(lons[sig.i], lats[sig.i], pch=16, cex=0.5)
box()
```

3 Summary

This vignette illustrates how to download seasonal forecast data from the ECOMS User Data Gateway using the `ecomsUDG.Raccess` R package and how to validate such forecasts using verification scores

from the `easyVerfication` R package. More information on both packages can be found in the respective GitHub repositories (`ecomsUDG.Raccess` and `easyVerification`) and on the ECOMS UDG wiki.

References

Mason, Simon J., and Andreas P. Weigel. 2009. "A Generic Forecast Verification Framework for Administrative Purposes." *Monthly Weather Review* 137 (1): 331–49.

Weigel, Andreas P., and Simon J. Mason. 2011. "The Generalized Discrimination Score for Ensemble Forecasts." *Monthly Weather Review* 139 (9): 3069–74.