Single point selections

First of all, the user needs to be registered in the ECOMS-UDG via the Thredds administration Panel (TAP) before using the package for accessing any dataset. Once registered, the user will have a valid username and password and can log-in using the `loginECOMS_UDG` function:

```
library(ecomsUDG.Raccess)
loginECOMS_UDG("user","password")
```

Now we are ready for accessing the ECOMS-UDG.

The following call to `loadECOMS` will load a time series of summer 2001 (JJA, `season = 6:8`, `years = 2001`) daily (`time = "DD"`) mean (`aggr.d = "mean"`) surface (2m) temperature (`var = "tas"`), for the coordinate -3.7E 40.4N, (`lonLim = -3.7`, `latLim = 40.4`) corresponding to the city of Madrid (Spain), as forecasted the previous March (`leadMonth = 2`) by the CFSv2 hindcast (`dataset = "CFSv2_seasonal"`). We will select the first 10 members (`members = 1:10`).

```
point.cfs <- loadECOMS(dataset = "CFSv2_seasonal",
                       var = "tas",
                       members = 1:10,
                       lonLim = -3.7,
                       latLim = 40.4,
                       season = 6:8,
                       years = 2001,
                       leadMonth = 2,
                       time = "DD",
                       aggr.d = "mean")
```

Note that a number of informative messages will appear on-screen:

```
[2015-05-15 11:46:49] Defining homogeneization parameters for variable "tas"
[2015-05-15 11:46:50] Defining geo-location parameters
[2015-05-15 11:46:50] Defining initialization time parameters
NOTE: Daily aggregation will be computed from 6-hourly data
[2015-05-15 11:46:54] Retrieving data subset ...
[2015-05-15 11:49:47] Done
```

The original variable is stored in the CFSv2 database as 6-hourly (see the available variables and their time resolution). Hence, we indicate to the function to perform a daily average using the argument `time = "DD"` via the `mean` function (`aggr.d = "mean"`).`

This is the size of the loaded object:

```
print(object.size(point.cfs)) # 20128 bytes
```

The returned object contains all the necessary information for data representation (geo-location, time, other metadata...). In the next lines we plot the loaded time series for each member. The element `Data` contains the data itself. In this case, it is a 2D array with the dimensions `member` (10 members selected) and `time` (92 days for June, July and August), as indicated by the `dimensions` attribute.

```
str(point.cfs$Data)
```

which returns information on the Data array structure:

```
 num [1:10, 1:92] 13.3 16.3 17.1 12.9 10.4 ...
- attr(*, "dimensions")= chr [1:2] "member" "time"
```

Note that, by convention, the dimensions of the Data array will be always ordered in the canonical form *member* > *time* > *lat* > *lon*. Several vertical levels are never loaded at the same time, so the dimension *level* will never appear. This will be indicated in the Variables element of the returned output. In this case it is NULL, because the variable loaded is a surface variable. Other useful metadada are included in the Variables element:

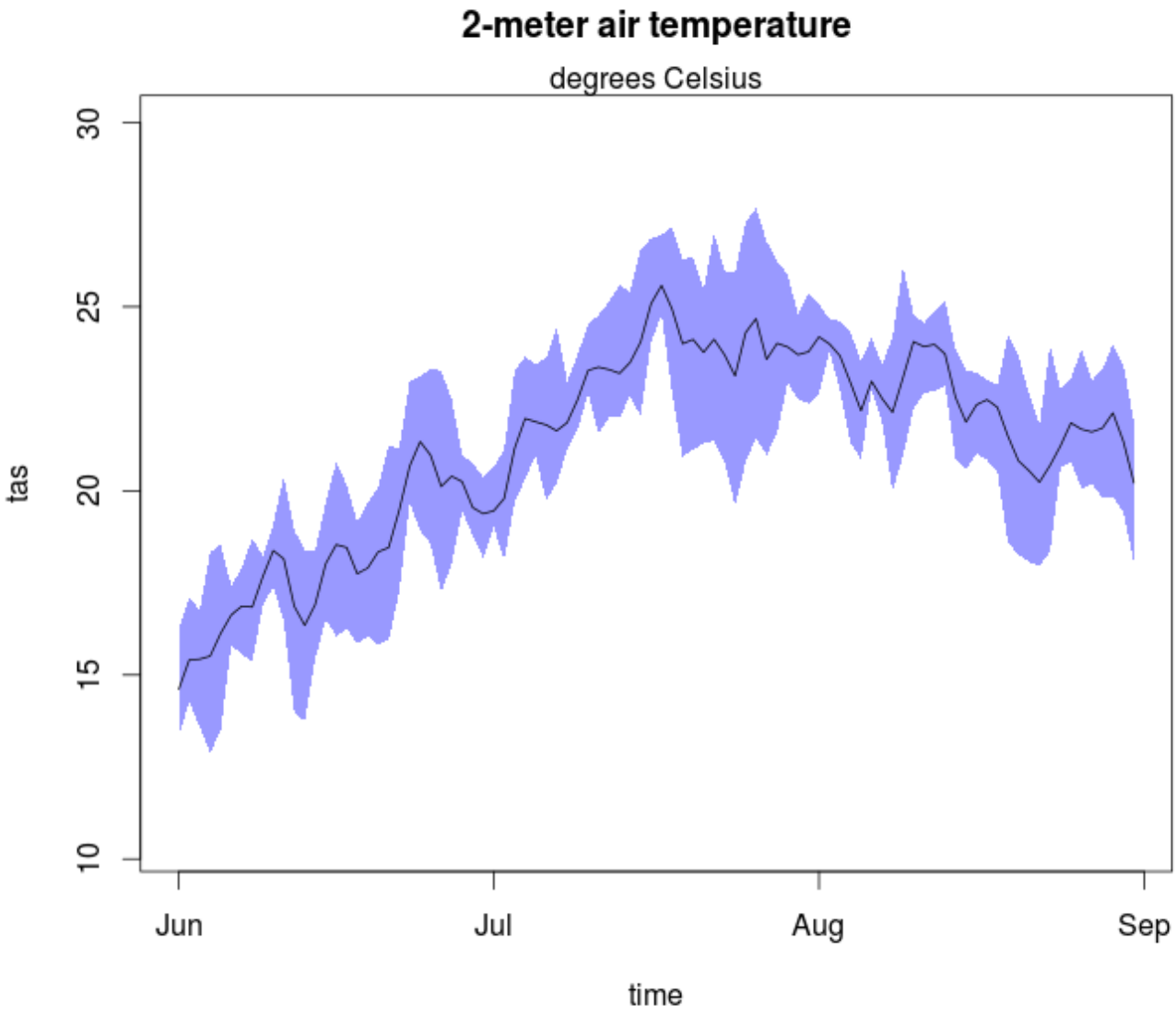
```
str(point.cfs$Variable)
```

returns the structure of the Variable element of the output:

```
List of 2
$ varName: chr "tas"
$ level  : NULL
- attr(*, "is_standard")= logi TRUE
- attr(*, "units")= chr "degrees Celsius"
- attr(*, "longname")= chr "2-meter air temperature"
- attr(*, "daily_agg_cellfun")= chr "mean"
- attr(*, "monthly_agg_cellfun")= chr "none"
- attr(*, "verification_time")= chr "DD"
```

In the following example we plot the time series with the multi-member mean and its dispersion (interquartilic range 25-75):

```
quartiles <- apply(point.cfs$Data, MARGIN = 2, FUN = quantile, probs = c(.25,.75))
ens.mean <- colMeans(point.cfs$Data)
dates <- as.POSIXlt(point.cfs$Dates$start, tz="GMT")
plot(dates, ens.mean, ylim = range(point.cfs$Data),
    type = 'n', ylab = point.cfs$Variable$varName, xlab = "time",
    main = attr(point.cfs$Variable, "longname"))
mtext(attr(point.cfs$Variable, "units"))
polygon(x = c(dates, rev(dates)),
       y = c(quartiles[1, ], rev(quartiles[2, ])),
       border = "transparent", col = rgb(0,0,1,.4))
lines(dates, ens.mean)
```


2-meter air temperature
degrees Celsius
tas
30
25
20
15
10
Jun
Jul
Aug
Sep
time