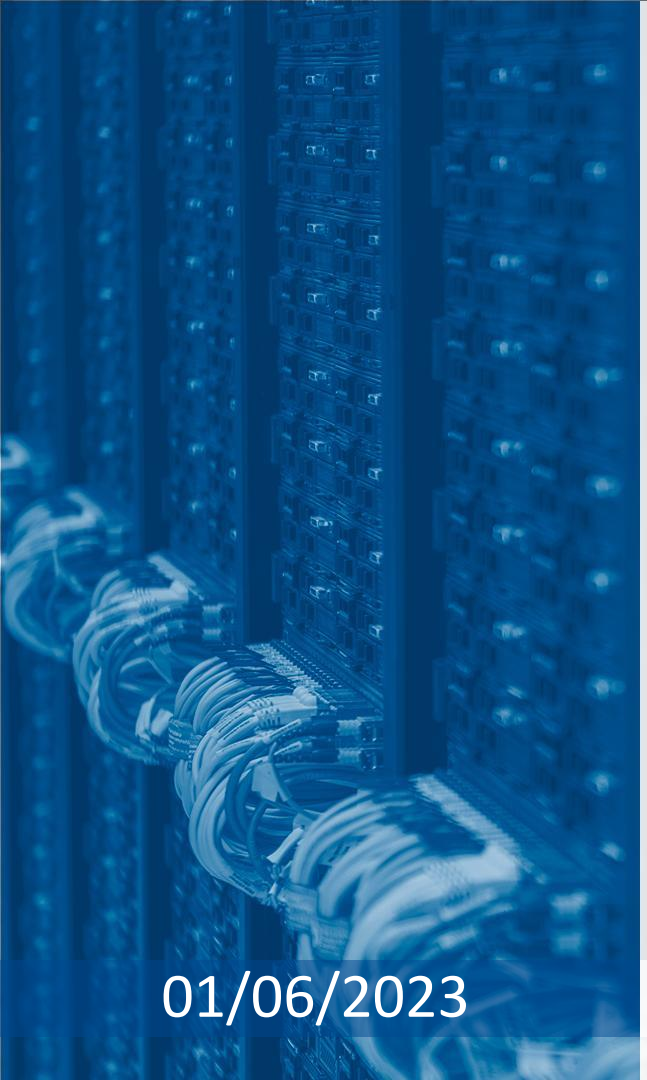


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

R user meeting

An-Chi Ho, Eva Rifà, Victòria Agudetse

contributor: Eva

01/06/2023

Agenda

1. Ice-breaker: Introduction to S3 and S4
2. News
    - General R
    - s2dv
    - startR
    - multiApply
    - CSTools
    - CSIndicators
    - SUNSET
3. Q&A

Ice-breaker


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Functional vs Object-Oriented Programming

Functional Programming (FP):

- A functional language is centred on functions
- Pure functions are when the output only depends on the inputs and without side-effects

Generally in R, FP is much more important than OOP. We typically solve complex problems by decomposing them into simple functions, not simple objects.

Object-Oriented Programming (OOP):

- Programming paradigm that organizes code around objects
- It enables you to define classes, create objects, implement methods

There are multiple OOP systems to choose from. The most importance are: S3, R6, and S4. S3 and S4 are provided by base R.

Reference: https://adv-r.hadley.nz/oo.html

Base type

Definition

- In R, everything is an object, not everything is object-oriented. We need to distinguish between **base objects** and **OO objects**
- The difference is that only OO objects have a “class” attribute, every object has a base type
- There are 25 base types; vectors: NULL, logical, integer, double, complex, character, list, and raw

Functions

- The function **class()** is safe to apply to S3 and S4 objects, but it returns misleading results when applied to base objects
- The function **typeof()** determines the base type of any object

```
attr(1:10, "class")
#> NULL
```

```
typeof(1:10)
#> [1] "integer"
```

```
class(CSTools::lonlat_prec)
#> [1] "matrix" "array"
```

Some definitions of S3

S3 is R's first and simplest OO system: we can't take away any part of it and still have a useful OO system. It is used in the base and stats packages, also most used in CRAN packages and R tools.

- **Class (S3)**: Defined by having attribute called "class" in an object. It determines the behavior and method dispatch for the object.
- **Generic (S3)**: It refers to a function that has multiple methods associated with it. The generic function acts as a placeholder or template for the methods. When a generic is called, the method dispatch mechanism identifies the appropriate method for that specific object's class.
- **Method (S3)**: A method is a specific implementation of a generic function for a particular class of objects. The method's name includes the name of the generic function, followed by a dot and the name of the class.

```
f <- factor(c("a", "b", "c"))
typeof(f)
#> [1] "integer"
attributes(f)
#> $levels
#> [1] "a" "b" "c"
#> $class
#> [1] "factor"
```

```
> mean
function (x, ...)
UseMethod("mean")
<bytecode: 0x562580f176f0>
<environment:
namespace:base>
```

S4 class

S4 provides a formal approach to functional OOP. The underlying ideas are similar to S3 but implementation is much stricter and makes use of specialised functions for creating classes:

- **Class (S4)**: You define an S4 class by calling setClass() with the class name and a definition of its slots, and the names and classes of the class data.
- **Generic (S4)**: Perform method dispatch, i.e. find the specific implementation for the combination of classes passed to the generic.
- **Method (S4)**: And then defining methods with setMethod().

```
# Define a class
setClass("Person",
  slots = c(
    name = "character",
    age = "numeric"
  )
)
```

```
# Check class
is(john)
#> [1] "Person"

# Access slot
john@name
#> [1] "John Smith"
```

```
# Create a generic
setGeneric("age", function(x)
           standardGeneric("age"))
# defining methods
setMethod("age", "Person",
           function(x) x@age)
```

```
# Construct new objects
john <- new("Person", name = "John Smith", age = NA_real_)
```

General R

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Package removed from CRAN

The following packages: **ClimProjDiags**, **CSTools**, **CSIndicators**, **startR**, **s2dv**, **easyNCDF** were removed from CRAN on 15th May due to dependency issue. The root reason is that the external package “climdex.pcic”, on which ClimProjDiags depends, was removed from CRAN. The other in-house packages have dependency on ClimProjDiags, so they were removed too as the chain effect.

- CSIndicators is on CRAN again
- We’re working on the other packages
  - ClimProjDiags: Temporarily remove the dependency on climdex.pcic
  - easyNCDF and CSTools will not depend on ClimProjDiags anymore
  - s2dv still needs this dependency, so as startR (chain effect)

shp_mask() developments

New developments in the function shp_mask() that now is in ‘s2dv’ GitLab. The new features and tests are being reported in Issue #75.

- Write documentation
- Added GADM (Database of Global Administrative Areas) database in the accepted Shapefile System Database.

Next steps:

- Allow to save mask array to NetCDF
- Add features if needed while being tested

Status: in branch s2dv::develop-shp_mask

s2dv

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PlotRobinson() changes

- Documentation added.
- Sanity checks improved.
- Correct argument “var_limits”, “title_size”.
- Polygon can actually be plotted in workstation.
- Remove ggplot2 background grid and axis label.
- An intermediate question before plotting global map with polygon (slow and incorrect)
- The default of argument “target_proj” is changed to “54030”, which is “ESRI:54030”. The argument must be a valid CRS string, e.g., proj4 (“+proj=robin”), EPSG, or ESRI code.

Status: in branch develop-PlotRobinson

Issue: https://earth.bsc.es/gitlab/es/s2dv/-/issues/95

CDORemap(): ncores

New argument ‘ncores’ to use multiple thread with cdo.

Use module CDO/1.9.8,

- On Nord3v2 compute node (interactive session) 1 thread: 33s / 8 threads: 16s
- On workstation, no difference is shown.

Status: in master

Issue: https://earth.bsc.es/gitlab/es/s2dv/-/issues/97

startR

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Bugfix when interpolating lon = 'all'

- Error: Bugfix when interpolating with requesting all val lon = 'all' One point missing. It was due to the target grid crop domain selection.
- Solution: extend the crop domain to all longitudes range.

```
Start(dat = path, var = variables,
      lon = "all",
      lat = "all",
      [...]
      transform = CDORemapper,
      transform_params = list(grid = 'r128x64', method = 'con'),
      transform_vars = c('lon','lat'),
      [...]
      retrieve = FALSE)
```

```
* Detected dimension sizes:
*      dat:   1
*      var:   1
*      lon: 127
*      lat:  64
*    sdate:   1
```

Status: in branch master

multiApply

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Discussion: Should Apply() return attributes?

Apply() doesn't have attributes returned, even when the input data has attributes and parameter “use_attributes” is used.

If `use_attributes = NULL`, the attributes are not taken by Apply() so it makes sense to not have attributes along with the returned array. However, with use_attributes defined, the attributes are still lost.

There are three types of attributes:
(1) All the attributes of the input data
(2) The ones in “use_attributes”
(3) The ones returned by “fun”

Issue: https://earth.bsc.es/gitlab/ces/multiApply/-/issues/15

CSTools

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CST_SaveExp new parameter startdates

The parameter 'startdates' has been added to the function CST_SaveExp as proposed in the last R user meeting. This parameter is used for naming the output files directly (without taking the start dates from Dates).

Parameter startdates

- Vector of dates used for the filenames when saving the data in multiple files.
- Same length as the start date dimension of data (if there is).
- Class Dates, 'POSIXct' or character with lenghts between 1 and 10.

```
CST_SaveExp(lonlat_prec, destination = "./dev-startdates/",
            single_file = FALSE, dat_dim = 'dataset', memb_dim = 'member',
            ftime_dim = 'ftime', var_dim = NULL,
            startdates = c('1', '2', '3'))
```

Status: in branch develop-CST_SaveExp-startdates

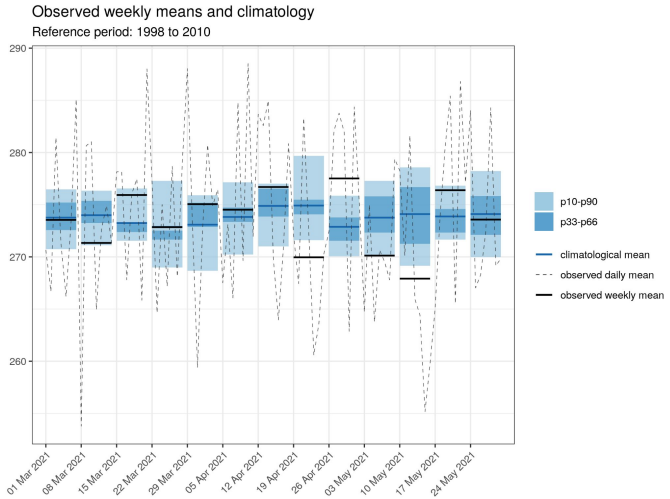
PlotWeeklyClim years outside ref period

- New development in order that it is possible to plot years outside the reference period.
- To do it, it's needed to specify the whole years of data in a new parameter called data_years.

Changes

- data_years: Vector with the complete years of the data. Ex: data_years = 2007:2021.
- ref_period: Vector with all the years of the reference period. Ex: ref_period = 2007:2010.
- last_date: Optional, to indicate the last date of the timeseries.
- subtitle: Optionla, used to set a subtitle.
- ytitle: Optional, it stands for the y-axis title.
- legend: Optional, whether to plot the legend or not.

```
PlotWeeklyClim(data = data, first_date = '2021-03-01',
               data_years = 1993:2021,
               ref_period = 1998:2010,
               title = "Observed weekly means and climatology",
               subtitle = "Reference period: 1998 to 2010")
```



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Status: in branch [develop-allow_plot_outside_reference]

CSIndicators

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

New release 1.0.1

NEWS:

- Add EnergyIndicators vignette figures
- Remove **ClimProjDiags** dependency
- Remove **s2dv** dependency

The **s2dv** dependency has been substituted with the following:

- Function s2dv::InsertDim has been substituted with an auxiliary function .insertdim() in here. Only in MergeRefToExp.
- Function s2dv::Reorder has been substituted with match() and aperm() from base R. Only in SelectPeriodOnDates.R

The **ClimProjDiags** dependency has been substituted with the following:

- Function ClimProjDiags::Subset has been substituted with an auxiliary function .arraysubset() in zzz.
- In: R/QThreshold.R and /SelectPeriodOnData.R

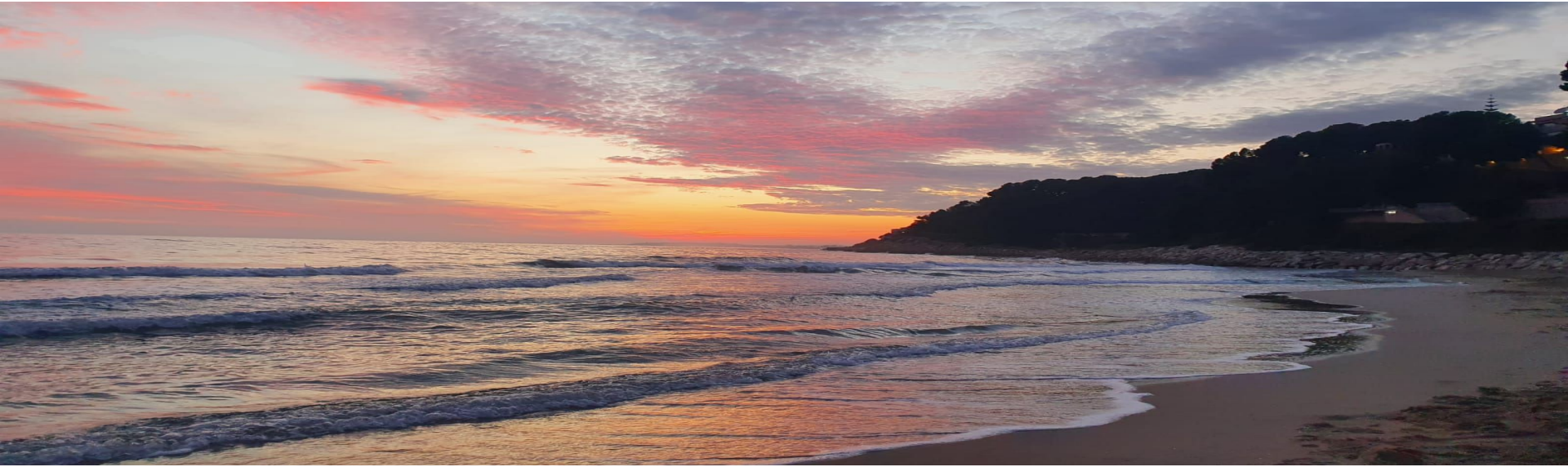
SUNSET

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

New name!

The ESS Verification Suite is now named **SUNSET**

“SUbseasoNal to decadal climate forecast post-processIng and asSEmenT suite”



Multiple variables in one atomic recipe

It is now possible **to load and process multiple variables in a single atomic recipe**, in the same s2dv_cube object (‘var’ dimension). This development is a first step to being able to compute indicators.

```
  Variables:
    name: tas, prlr
    freq: monthly_mean


> data$hcst$dims
      dat       var      sday     sweek     syear      time  latitude longitude
        1         2         1         1        12         2        21        21
 ensemble
       25
```

status: in master

Visualization module: New plotting options

Issue: https://earth.bsc.es/gitlab/es/auto-s2s/-/issues/72

Some new options have been added to the visualization module:

- The user can now specify if they want the plots for all forecast times to be in the same file (‘multi-panel’) or one file per forecast time (‘single-panel’).
- For the single-panel option, the user can also choose their preferred projection.

```
Visualization:

  plots: skill_metrics, forecast_ensemble_mean, most_likely_terciles

  multi_panel: no

  projection: robinson
```

status: in branch dev-Visualization-PlotRobinson

Visualization module: Choosing a projection

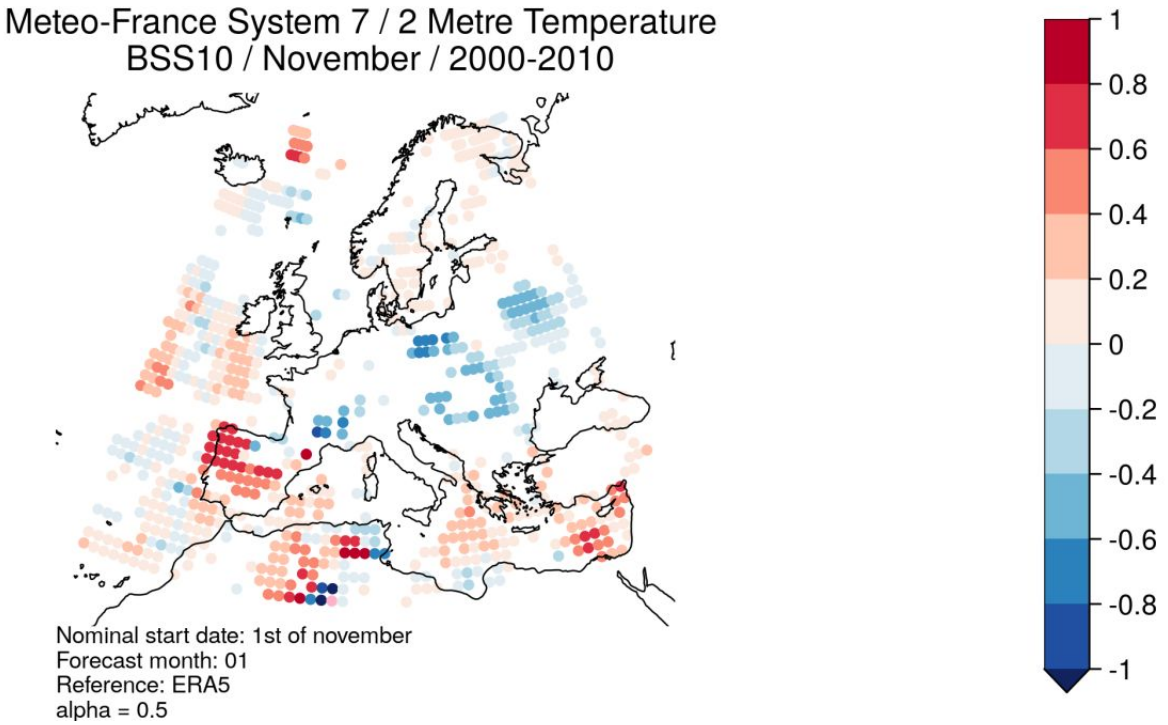
There are two possibilities for the ‘projection’ parameter:

1. Using the built-in projection options: 'cylindrical_equidistant', 'robinson', 'lambert_europe', and 'stereographic'.
2. Specifying a custom CRS string, EPSG or ESRI code for the projection you want to use. Be aware that different machines (e.g. WS vs Nord3v2) might require different codes for the same projection. See: PlotRobinson() Visualization Vignette

status: in branch dev-Visualization-PlotRobinson

Visualization module: Single-panel plots

Example: Single-panel plot using the Lambert Europe projection, including an individual title and a caption.



User presentation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

SPEI Indicator

The Standardised Precipitation-Evapotranspiration Index (SPEI) is a multiscalar drought index based on climatic data. It can be used for determining the onset, duration and magnitude of drought conditions with respect to normal conditions in a variety of natural and managed systems such as crops, ecosystems, rivers, water resources, etc. The SPI is calculated using monthly (or weekly) precipitation as the input data. The SPEI uses the monthly (or weekly) difference between precipitation and PET. This represents a simple climatic water balance which is calculated at different time scales to obtain the SPEI

Development

- Code was provided from Alba and Bala
- The SPEI Indicator will be included in the module Indicators within SUNSET that will call a CSIndicators function called CST_PeriodSPEI.

More information:

See issue #27: https://earth.bsc.es/gitlab/es/csindicators/-/issues/27

status: in branch develop-SPEI

Module Indicator workflow

SUNSET::IndicatorsModule

CSIndicators::PeriodSPEI

Step0: variables (e.g. tasmin, tasmax, prlr) are loaded in the same s2dv_cube

Step1: units transformation

	original	transformed
Temperature	K	C
Precipitation	mm/day	mm/month

⇨ s2dv_cube subset with 3 independent variables data

Step2: checks

- data units
- provided data vs evapotranspiration estimation method

Step3: estimate evapotranspiration

package SPEI	variables
hargreaves	**tasmin**, **tasmax**, *lat*
hargreaves modified*	**tasmin, tasmax, prlr**, *lat*
thornthwaite*	**tas** (mean), *lat*

Step4: accumulation

- from matrix to vector + accumulation with `rollapply(data_vector, accum_scale, sum)`
- vector of **complete years** to avoid accumulation with "previous" months which are not consecutive

Step5: standardization

- parametric (select distribution: hardcoded to "log-Logistic")
- non-parametric

parametric is the hardcoded option (when calling spX_crossvalid) and the only option that allows for cross_validation and handle_infinity

hcst, **fcst** *and* **obs** *are treated independently: loop* `for (dat in datasets){...}` *where* datasets *can be specified when calling the function and needs to be all or some of the* **names(data)**

Source: Alba Llabrés

Structure of Module Indicators

Step 1

```
c(data, recipe) %<-%
load_data_from_recipe(recipe_file)
```

```
> str(data)
List of 3
$ hcst: List of 4 -> ‘s2dv_cube’ --> exp
$ fcst: List of 4 -> ‘s2dv_cube’ --> exp_cor
$ obs: List of 4 -> ‘s2dv_cube’ --> obs
```

Step 2

```
# Subset and
transform units if
necessary:
```

```
obs <- list(tasmax = CST_Subset(), tasmin = CST_Subset(),prlr = CST_Subset())
obs <- transform_units(obs)
exp <- list(tasmax = CST_Subset(), tasmin = CST_Subset(), prlr = CST_Subset())
exp <- transform_units(exp)
exp_cor <- list(tasmax = CST_Subset(), tasmin = CST_Subset(), prlr = CST_Subset())
exp_cor <- transform_units(exp)
```

Step 3

```
# Set parameters
```

Step 4

```
# Call CST_PeriodSPEI
```

```
spei_obs <- CST_PeriodSPEI(exp = obs, exp_cor = NULL,...)

spei_exp_fcst <- CST_PeriodSPEI(exp = exp, exp_cor = exp_cor,...)
```

Structure of CST_PeriodSPEI

```
CST_PeriodSPEI <- function(exp, exp_cor = NULL, pet = NULL,
                           time_dim = 'syear', leadtime_dim = 'time', memb_dim =  'ensemble',
                           lon_dim = 'longitude', lat_dim = 'latitude',
                           accum = 1, start = NULL, end = NULL,
                           pet_method = NULL, standardization = TRUE,
                           params = NULL,   param_error = -9999,  handle_infinity = FALSE,     cross_validation = FALSE,
                           method = 'parametric', distribution = 'log-Logistic',
                           fit = 'ub-pwm',  n_procs = 4) {
 # dates
 dates <- exp[[1]]$attrs$Dates
 # lat
 lat <- exp[[1]]$coords[[lat_name]]
 # exp and exp_cor
 exp <- lapply(exp, function(x) x$data)
 exp_cor <- lapply(exp_cor, function(x) x$data)

 res <- PeriodSPEI(exp = exp, exp_cor = exp_cor, pet = NULL, dates = dates, lat = lat,
                   time_dim = 'syear', leadtime_dim = 'time', memb_dim = 'ensemble', lon_dim = 'longitude', lat_dim = 'latitude',
                   accum = 1, start = NULL, end = NULL, pet_method = NULL,standardization = TRUE,
                   params = NULL, param_error = -9999,      handle_infinity = FALSE,cross_validation = FALSE,
                   method = 'parametric', distribution = 'log-Logistic', fit = 'ub-pwm',
                   n_procs = 4)

}
```

Structure of PeriodSPEI

```
PeriodSPEI <- function(exp = exp, exp_cor = exp_cor, pet = NULL,dates = dates, lat = lat,
                   time_dim = 'syear', leadtime_dim = 'time', memb_dim = 'ensemble', lon_dim = 'longitude', lat_dim = 'latitude',
                   accum = 1, start = NULL, end = NULL, pet_method = NULL,standardization = TRUE,
                   params = NULL, param_error = -9999,      handle_infinity = FALSE,cross_validation = FALSE,
                   method = 'parametric', distribution = 'log-Logistic', fit = 'ub-pwm',
                   n_procs = 4) {
 # Part (1): Initial checks
  [...]
 # (2) Complete dates
 [...]
 # (2) Loop for evapotranspiration, accumulation, spei_standardization for exp and exp_cor
 for (data in .return2list(exp, exp_cor)) {
     pet <- evapotranspiration(data, dates_monthly, pet_method, time_dim, leadtime_dim, memb_dim, lon_dim, lat_dim, n_procs)
     data_accum <- accumulation(diff_P_PET, dates_monthly, accum, time_dim, leadtime_dim, memb_dim, n_procs)
     spei_dat <- spei_standardization(data_accum = data_accum, leadtime_dim = leadtime_dim, time_dim = time_dim, memb_dim = memb_dim,
                                      cross_validation = cross_validation, handle_infinity = handle_infinity,n_procs = n_procs,
                                    accum = accum,param_error = param_error,
                                    params = params, method = method, distribution = distribution, fit = fit)
     spei_res[[k]] <- spei_dat
 }
 return(spei_res)

}
```

Function PeriodSPEI

Step 1 Initial checks and prepare dates

Step 2 Compute the SPEI for exp (hcst) Get params

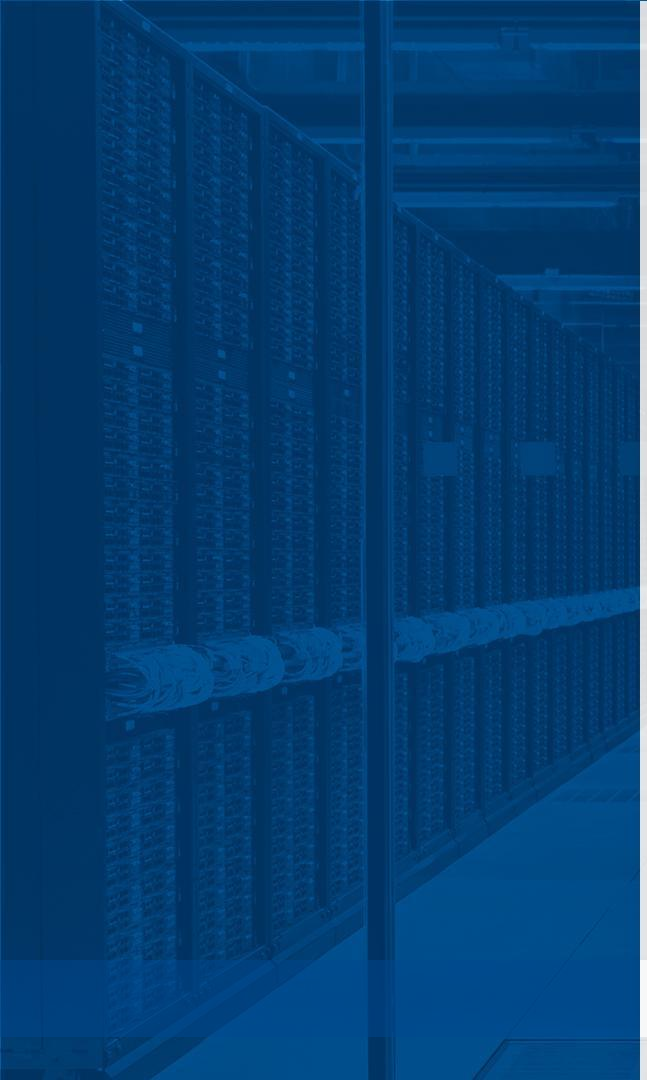
Step 3 Compute the SPEI for exp_cor (fcst) with params from exp

```
pet <- evapotranspiration(data = data,
                          dates_monthly = dates_monthly,
                            lat = lat,
                          pet_method = pet_method[k],
                            [...])
# Accumulation
diff_p_pet <- data$prlr - pet
data_accum <- accumulation(data = diff_p_pet,
                             dates_monthly = dates_monthly,
                           accum = accum,
                             [...])
# Standardization:
if (standardization) {
  spei_dat <- spei_standardization(data = data_accum,
                                   [...]
                                   ref_period,
                                   cross_validation,
                                   handle_infinity,
                                   accum,
                                   param_error,
                                   params, method
                                   distribution,
                                   fit)
  params <- spei_dat$params
}
```

Value:

```
> str(result)
List of 3
 $ exp    :  MD array with SPEI for exp data
 $ exp_cor: MD array with SPEI for exp_cor data
 $ params:  MD array with params from exp with same dimensions except memb_dim and coef dimension
```

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thanks for joining

Next meeting: 6th July 12h