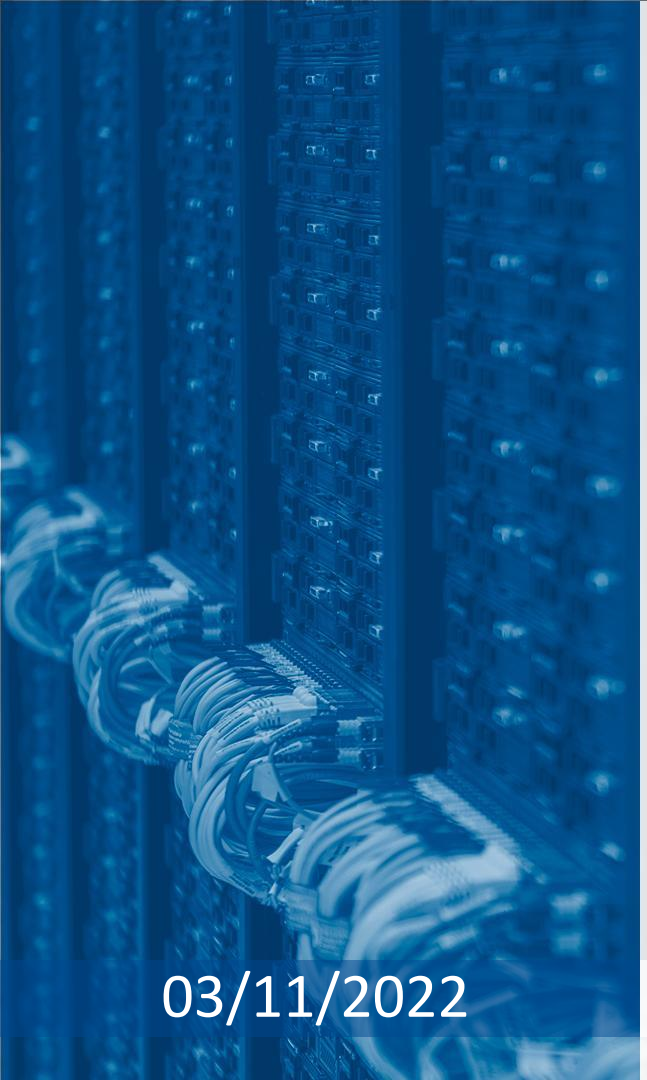

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

R user meeting

An-Chi Ho and Eva Rifà

contributor: Victòria

03/11/2022

Agenda

1. Ice-breaker: package “clock”
2. News
    - General R
    - s2dv
    - startR
    - CSIndicators
    - CSTools
3. User presentation: log4r [Victòria]
4. Q&A
    - Does Apply() use multiple nodes? [An-Chi]
    -

Ice-breaker

Package “clock”

Package description: *“Provides a comprehensive library for date-time manipulations using a new family of orthogonal date-time classes (durations, time points, zoned-times, and calendars) that partition responsibilities so that the complexities of time zones are only considered when they are really needed. Capabilities include: date-time parsing, formatting, arithmetic, extraction and updating of components, and rounding.”*

CRAN: https://cloud.r-project.org/web/packages/clock/index.html
Introduction: https://clock.r-lib.org/
Function reference: https://clock.r-lib.org/reference/index.html

date-time manipulation tools

- **base**: class “Date”, “POSIXct”
- **lubridate**: https://lubridate.tidyverse.org/ Deal with R’s native date (Date) and date-time (POSIXct) classes
- **clock**: Deal with R’s two date classes plus provide entirely new date-time types
- **PCICt**: Deal with different calendar types (e.g., 365 days, 360 days)

Some examples:
https://earth.bsc.es/gitlab/aho/aho-testtest/-/blob/master/helper_script/test_clock.R

General R

Maintainers' job distribution

In theory,

- **Eva**: CSTools, CSIndicators
- **An-Chi**: startR, s2dv, ClimProjDiags, and the rest of the packages.
- We still share the workload and know each other's tasks

→ If you want to ask questions/discuss things outside of GitLab or user meeting, contact the corresponded maintainer first.

→ If you open an issue or merge request on GitLab, feel free to tag both of us (at least the key maintainer must be tagged.)

Open RStudio on Nord3v2

Use **`--exclusive`** to prevent instability.

```
#!/bin/bash
#SBATCH --cpus-per-task 16
#SBATCH --ntasks 1
#SBATCH --time 10:00:00
#SBATCH --exclusive
#SBATCH --job-name rstudio-server
#SBATCH --output rstudio-server-%J.out
#SBATCH --error  rstudio-server-%J.err
```

- Try not to ask for highmem at the same time. You can use medmem: `#SBATCH --constraint=medmem`
- Don't use multiple ntasks (--n); use multiple cpus (--c) instead: ~~`#SBATCH --ntasks 8`~~

First time and need template? https://earth.bsc.es/wiki/doku.php?id=tools:Rtools&s[]=Rtools#rstudio-server

Slurm documentation: https://slurm.schedmd.com/sbatch.html

PATC 2022: R tools hands-on

We're going to give the annual PATC tutorial next Thursday (10th November). If you're learning our tools (startR, CSTools, CSIndicators, s2dv), you can check the materials we prepared. We will put the slides and hands-on on Earth wiki and GitLab.

First peek:
https://earth.bsc.es/gitlab/es/startR/-/tree/develop-PATC2022/inst/doc/tutorial/PATC2022

s2dv

New release 1.3.0

s2dv 1.3.0 (Release date: 2022-10-17)

- New functions: Bias, AbsBiasSS, CRPS, CRPSS
- split RPSS parameter 'weights' into 'weights_exp' and 'weights_ref'
- The warning message format is consistent; use internal function .warning() for all the cases
- PlotEquiMap() bugfixes when lon vector is not continuous
- PlotEquiMap() parameter "dots", "varu", "varv", and "contours" array latitude and longitude dimension order is flexible
- PlotLayout(): Add parameter to change subplot title size
- PlotLayout works with CSTools::PlotMostLikelyQuantileMap()
- Parameter "dat_dim" can be NULL in all functions
- Add "dat_dim" in RPS and RPSS to allow multiple datasets to be calculated
- DiffCorr: Add two-sided significance test. New param "test.type" to specify the one- or two-sided significance test.

Next step: plotting function improvement

Bug: AbsBiasSS() significance test

status Fixed and in master branch

Due to a bug in the function, the significance test (RandomWalkTest) input had only one value, so the result was always insignificance.

Source the function from gitlab master branch if you need to use this function.

```
source("https://earth.bsc.es/gitlab/es/s2dv/-/raw/master/R/AbsBiasSS.R")
```

Season() poor sanity checks

The function Season() does not return any error or warning when out-of-range parameters are provided. Check issue https://earth.bsc.es/gitlab/es/s2dv/-/issues/81

status Not improved yet

Unify the structure of functions with significance test

[Corr() & other appliable functions]

- Parameter ***alpha*** is numeric (0.05 by default), replacing *conf.lev*
- Outputs are *corr*, *p.val*, *conf.lower*, *conf.upper*, *sign*
- Flag parameters *pval = TRUE, conf = TRUE, sign = FALSE* --> If TRUE, return the corresponding value.

[DiffCorr() / ResidualCorr()]

- Parameter alpha is numeric (0.05 by default); *cannot be NULL*
- Outputs are *diff.corr/res.corr, p.val, sign*
- Flag parameters *pval = TRUE, sign = FALSE* --> If TRUE, return the corresponding value.

status Not improved yet

See issue https://earth.bsc.es/gitlab/es/s2dv/-/issues/79

CSIndicators

New release 0.0.2

- **Changes:**
  - Revise examples using s2dv::InsertDim in MergeRefToExp().
  - Sanity check correction in CST_ functions.
  - Corrected documentation.
  - Changed examples to avoid import data from CSTools: *lonlat_prec* and *lonlat_temp*.

→ **Future development:**

- Add real sample data
- New vignettes
- Threshold functions to allow between thresholds or equal
- Correct figures of EnergyIndicators vignette
- s2dv_cube object development for CST_ functions

Sanity check correction in CST_ functions

- In CSIndicators: ‘start’ and ‘end’ parameters are used to subset the data with specific dates range.

```
CST_PeriodAccumulation <- function(data, start = NULL, end = NULL,
                                   time_dim = 'ftime', na.rm = FALSE,
                                   ncores = NULL)
```

→ Requirement to subset: **dates** array dimensions have to match with **data** array.

→ If dimensions don’t match all **data** is used and a warning appears:

```
warning("Dimensions in 'data' element 'Dates$start' are missed and ",
        "all data would be used.")
```

Note: All functions work better if dimensions and dimension names are provided for all objects (data, exp, obs, dates…).

Corrected documentation

Format requirements:

- Documentation format with roxygen2: write .Rd files in the man/ directory with devtools::document().
- Line breaks at 80 characters
- Elements: description, inputs, param, examples in order

Examples requirements:

- Useful use cases that work perfectly
- Fast run
- Avoid using data from other functions

```
#'Period Accumulation on 's2dv_cube' objects
#'
#'Period Accumulation computes the sum (accumulation) of a given variable in a
#'period. Providing precipitation data, two agriculture indices can be obtained
#'by using this function:
#'\itemize{
#'  \item\code{SprR}{Spring Total Precipitation: The total precipitation from
#'                   April 21th to June 21st}
#'  \item\code{HarR}{Harvest Total Precipitation: The total precipitation from
#'                   August 21st to October 21st}
#'}
#'
#'@param data An 's2dv_cube' object as provided function \code{CST_Load} in
#'  package CSTools.
#'@param start An optional parameter to defined the initial date of the period
#'  to select from the data by providing a list of two elements: the initial
#'  date of the period and the initial month of the period. By default it is set
#'  to NULL and the indicator is computed using all the data provided in
#'  \code{data}.
#'@param end An optional parameter to defined the final date of the period to
#'  select from the data by providing a list of two elements: the final day of
#'  the period and the final month of the period. By default it is set to NULL
#'  and the indicator is computed using all the data provided in \code{data}.
#'@param time_dim A character string indicating the name of the dimension to
#'  compute the indicator. By default, it is set to 'ftime'. More than one
#'  dimension name matching the dimensions provided in the object
#'  \code{data$data} can be specified.
```

CSTools

New release 4.1.0

New features:

- Dependency on package 's2dverification' is changed to 's2dv'
- CST_BiasCorrection new parameters 'memb_dim', 'sdate_dim', 'ncores'
- CST_Calibration is able to calibrate forecast with new parameter 'exp_cor'
- CST_QuantileMapping uses cross-validation and provides option to remove NAs; new parameters 'memb_dim', 'sdate_dim', 'window_dim' and 'na.rm'; 'sample_dim' and 'sample_length' are removed
- s2dv_cube() new parameter 'time_dim'

Fixes:

- as.s2dv_cube() detects latitude and longitude structure in startR_array object
- Data correction: 'lonlat_data' is renamed to 'lonlat_temp'; 'lonlat_prec' is corrected by one-day shift
- Typo and parameter correction in vignette 'MostLikelyTercile_vignette'
- Figure and result correction in vignette 'RainFARM_vignette'
- PlotMostLikelyQuantileMap() works with s2dv::PlotLayout

Improvements

- **CST_BiasCorrection()**: memb_dim does not function well
  - Corrected ‘memb_dim’ parameter.

    `if (dim(obs$data)['member'] != 1)` → `(dim(obs)[memb_dim] != 1)`
  - Added initial checks for BiasCorrection().

- **CST_QuantileMapping**: New function
  - New parameters:
    - sdate_dim (default 'sdate')
    - memb_dim (default 'member')
    - window_dim (default NULL)
    - na.rm (default FALSE)
  - New development:
    - Cross-validation
    - NA functionality
    - Work with dimension names

Bug in CST_Analogs

When parameter “obsVar” is used, The array is not subsetted properly to the best analogs due to a bug in Subset() usage.

dim(res)

nAnalogs	lat	lon
10	3	3

“nAnalogs” is the same as input “time” dimension.

status Not solved yet

Check issue: https://earth.bsc.es/gitlab/external/cstools/-/issues/105

User presentation:
Logging in R: The log4r package [Victòria]

Logging in R

A **log** is a recording (generally stored in a file) of display messages that describe events that occur when a software runs.
Logs are helpful for software development, debugging, or simply to keep information about a particular run or experiment that we might want to recover later.

One way to achieve this in an R script is to simply print the terminal output to a file with a function like cat(), sink() or capture.output().
But these have their limitations, and can be dangerous sometimes.

The log4r package

Luckily, there are “proper” ways to do this. Several packages in R provide specific functions to create and manage logs.

For example, let’s take a look at the ‘log4r’ package:
CRAN: https://CRAN.R-project.org/package=log4r

From the log4r documentation:
“The log4r package is meant to provide a fast, lightweight, object-oriented approach to logging in R based on the widely-emulated 'log4j' system and etymology.”

Logging with log4r

The basics of generating a log with log4r are quite simple: We need to create a **logger object**, to which we can assign the path to the log file and a priority threshold, among other things.

The places where we want the messages to be logged is assigned through the 'appenders'. E.g. console_appender() for the console, or file_appender() for a file.

There are 5 priority levels:

- DEBUG (1), INFO (2), WARN (3), ERROR (4), and FATAL (5).

Each level corresponds to a different type of message. When we assign a threshold to the 'logger' object, only the messages that are **the same level or higher** will be printed. The default is "INFO".

Logging with log4r: A simple example

```
> library(log4r)
# We create a logger object with the logger() function.
> logger <- logger(threshold = 'INFO', appenders = file_appender("test.log"))

# Now let's log some messages
> debug(logger, "Stuff only developers care about")
> info(logger, "I'm printing to a log file!")
> warn(logger, "Are you sure you wanna do this...?")
> error(logger, "Something went wrong.")
> fatal(logger, "Self-destructing in 3, 2, 1...")

# We set the log level to 'INFO': only messages above this threshold will be printed.
# If we check the log file, we should not see the DEBUG message:
> readLines("test.log")
[1] "INFO  [2022-10-28 11:44:03] I'm printing to a log file!"
[2] "WARN  [2022-10-28 11:44:07] Are you sure you wanna do this...?"
[3] "ERROR [2022-10-28 11:44:12] Something went wrong."
[4] "FATAL [2022-10-28 11:44:16] Self-destructing in 3, 2, 1..."
```

Logging with log4r: Another simple example

```
# We can also create a logger object that appends to a file and to the console
> logger <- logger(threshold = "INFO",
>                  appenders = list(file_appender("test.log"),
>                                   console_appender()))
> info(logger, "Messages will now be appended to the console")
INFO  [2022-11-02 11:40:41] Messages will now be appended to the console

# And we can use the level() function to change the log threshold on the fly:
> level(logger) <- "DEBUG"
> debug(logger, "Another message only devs want to read.")
DEBUG [2022-11-02 11:43:40] Another message only devs want to read.
```

The pros and cons of log4r

Log4r pros:

- Good performance [See: Comparison with other similar packages]
- Relies on an object of class ‘logger’, which can be modified, passed on and returned from functions, etc.
- Provides some more ‘fancy’ built-in layout formats (JSON, logfmt) [See: Structured Logging] and appenders to write to other connections [See: Logging Beyond Local Files]
- Users can make custom layouts and appenders

Cons:

- No easy way (that I have found…) to properly format complex multi-line outputs and evaluate expressions [Possible work-arounds: combining log4r with sink() or the ‘evaluate’ package]

The pros and cons of log4r

```
# Example: Logging the summary() of an array
> my_array <- rnorm(300)
> dim(my_array) <- c(time = 3, latitude = 10, longitude = 10)
> summary(my_array)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-4.107477 -0.768419 -0.005368 -0.048431  0.585346  2.324948

# Trying to log the output of summary(my_array) results in a disaster
> info(logger, summary(my_array))
 INFO  [2022-11-02 12:08:58]
 -4.10747739703713-0.768419321756904-0.00536806011248123-0.04843144461472020.585346303843
 5652.32494802664473
```

Trying the same with a different package

```
# Other packages make this easier
> library(logger) # Not installed in WS or HPC machines
> log_appender(appender_file("logger.log"))
> log_eval(summary(my_array), multiline = TRUE, level = INFO)
> readLines("logger.log")
[1] "INFO [2022-11-02 12:46:57] Running expression: ====================="
[2] "INFO [2022-11-02 12:46:57] summary(my_array)"
[3] "INFO [2022-11-02 12:46:57] Results: ================================="
[4] "INFO [2022-11-02 12:46:57]     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. "
[5] "INFO [2022-11-02 12:46:57] -4.10747 -0.76841 -0.00536 -0.04843  0.58534  2.32494 "
[6] "INFO [2022-11-02 12:46:57] Elapsed time: 0 sec"
```

Q & A

Does Apply() use multiple nodes?

Anyone has experience to share? When submitting a job with multiple nodes and use `ncores > 1` in functions using Apply(), do you notice the better efficiency?

```
#!/bin/bash
#SBATCH -N 2
#SBATCH -c 16
#SBATCH -n 2
```

Thanks for joining

Next meeting: 1st Dec. 2022 (4 pm)