Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Training: Introduction to the Department R Tools

Núria Pérez-Zanón and An-Chi Ho

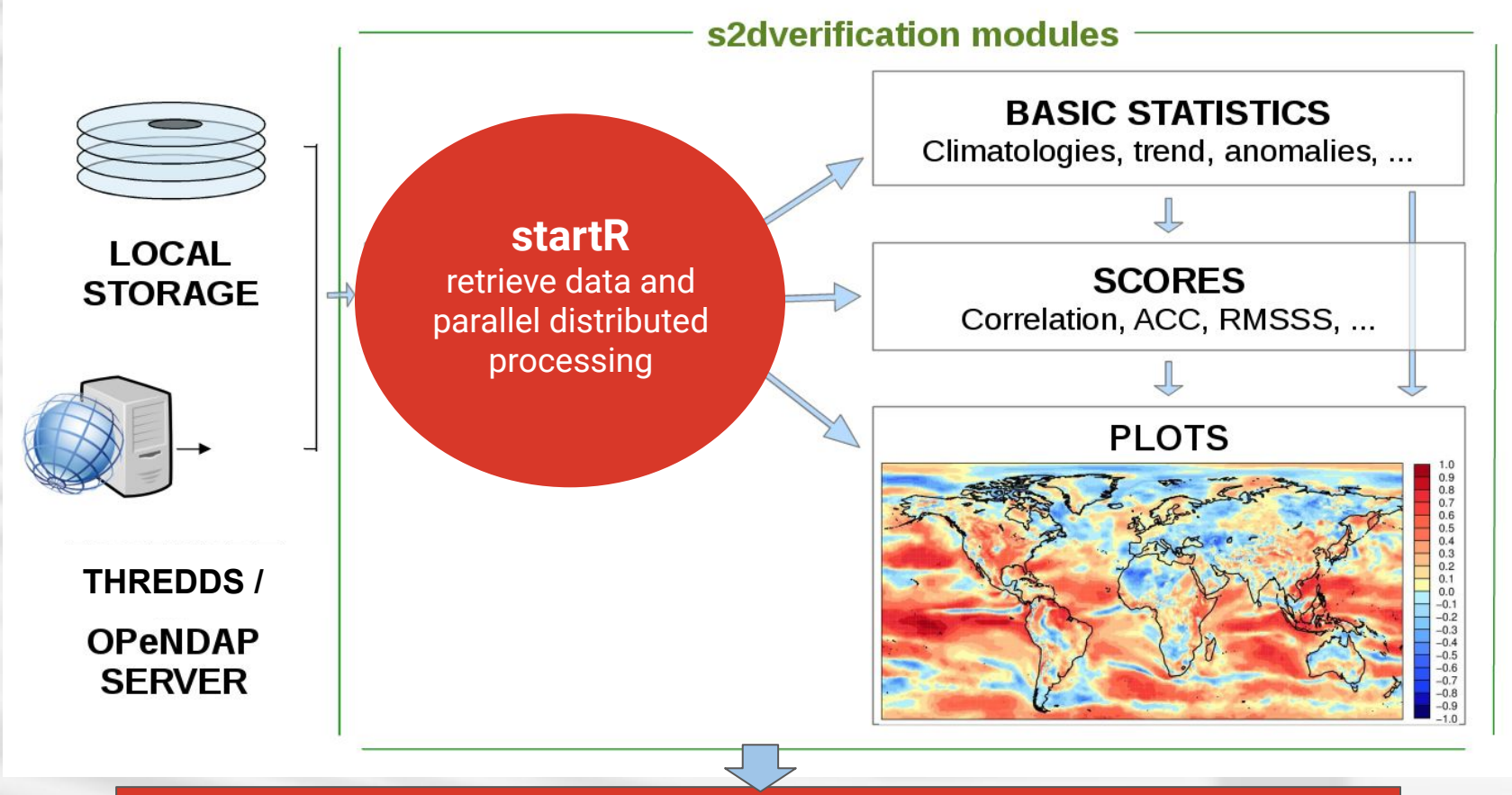
January 2020

Index

Overview of the department R tools

Department R Tools scheme

multiApply paradigm



CSTools

Climate Services Tools - MEDSCOPE Toolbox

MEDiterranean Services Chain based On Climate PrEdictions

multiApply paradigm

multiApply paradigm
1. What is an array object in R?

What is an array object in R?

- *An array is data object which can store data with **any number of dimensions**.*
- *Each dimension can have its own **name***
- ***array() function** allows to create them or you can add dimensions to an existing vector.*

N-dimensional array with named dimensions

Synthetic samples in array of 4 dimensions

```
data <- array(rnorm(1800), dim = c(var = 2, level = 10, lat = 90, lon = 1))

# variable in position 1 along 'var' dimension is saved in a new object:
profile_NO <- data[1, , , ]

# Selection of levels using a function
Low_levels <- s2dverification::Subset(data, along = 'level', indices = 1:3)
```

multiApply paradigm
2. How multiApply works?

How multiApply works?

Basic Example: Compute the ensemble mean of difference between two levels on the ratio of two variables

Data dimension (~24GB)

var	level	ensemble	time	latitude	longitude
2	2	51	120	256	512

Error: cannot allocate vector of size 23.9 Gb
That's why we also need startR (see later)

Using for loops

```
data <- array(rnorm(4 * 10 * 120 * 20 * 40),
                    c(var = 2, level = 2, ensemble = 10, time = 120, lat = 20, lon = 40))
Output <- array(NA, dim = c( time = 120, latitude = 20, longitud = 40))
Tmp_out <- NULL

for (k in 1:120) { # time
   for (n in 1:20){ # lat
      for(m in 1:40) { # lon
         for(j in 1:10) { # ens
               Out <- data[1,2,j, k, n, m] / data[2,2,j,k,n,m] - data[1,1,j,k,n,m]/data[2,1,j,k,n,m]
               Tmp_out <- c(Tmp_out, Out)
         }
         Output[k,n,m] <- mean(Tmp_out)
         Tmp_out <- NULL
      }
   }
}
```

- Foresee the final ouput
- Loops on dimensions
- Calculation definition
- Clean objects

How multiApply works?

Basic Example: Compute the ensemble mean of difference between two levels on the ratio of two variables

Data dimension (~24GB)

var	level	ensemble	time	latitude	longitude
2	2	51	120	256	512

Error: cannot allocate vector of size 23.9 Gb
That's why we also need startR (see later)

Using multiApply

```
data <- array(rnorm(4 * 10 * 120 * 20 * 40),
              c(var = 2, level = 2, ensemble = 10, time = 120, lat = 20, lon = 40))

library(multiApply)

result <- Apply(list(data), target_dims = c('var', 'level', 'ensemble'), fun = function(x) {
                  mean(apply(x, 3, function(y) {
                    y[1,2]/y[2,2] - y[1,1]/y[2,1]
                    }))
                  }, ncores = 4)$output1
```

x Foresee the final ouput

x Loops on dimensions

- Calculation definition

x Clean objects

Note: The number of code lines is reduced from 11 to 5 → this translates into easier maintenance

How multiApply works?

multiApply is an R package with a single function Apply.

Parameters

- **data =** a list of array object(s)
- **target_dims =** a list of dimension names to be used for the analysis
- **fun =** a function describing the analysis
- ….
- **ncores =** an integer indicating the number of cores to use in parallel computation

Information needed to use multiApply
1) Which are the dimensions of my data?
2') What function I want to apply?
2'') Over which dimensions?
3) Which are the output dimensions?

Note:
- At workstations do you have 4 or 8 cores
- The parallel computation can also be used on HPC and the 'ncores' parameter depends on the number of cores you request.

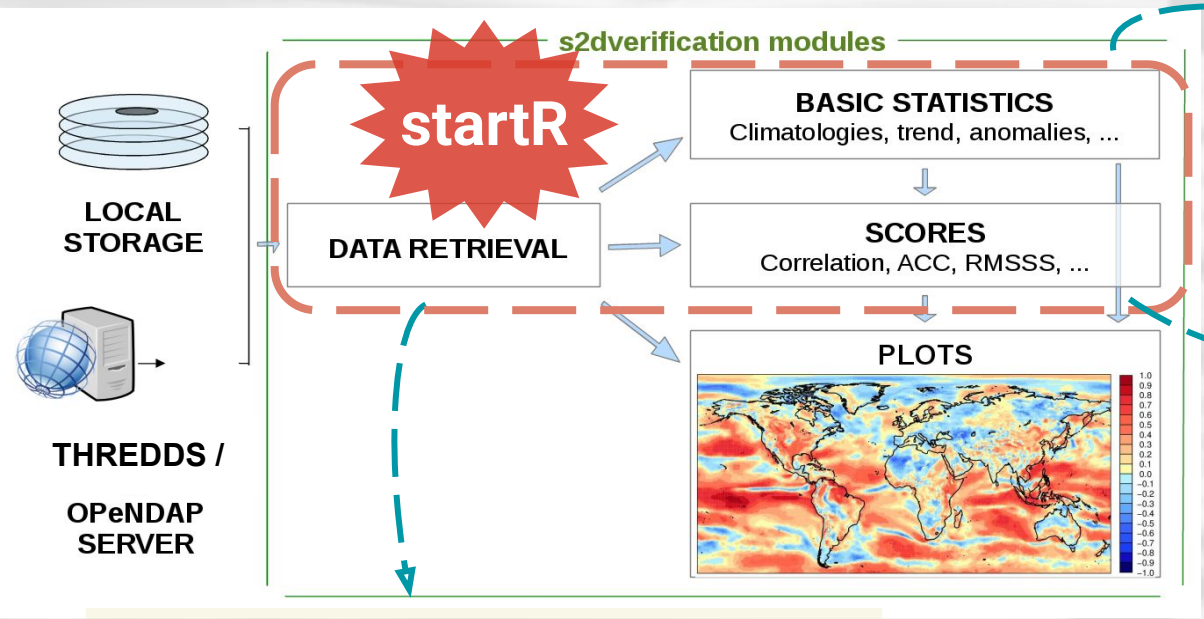
multiApply paradigm
3. Which are the benefits?

Which are the benefits of multiApply?

- The computation can be parallelized
- The number of code lines is reduced and it is easier to maintain
- The same function can be applied on different dimensions
- The user doesn't need to initialize the outputs and foresee the dimensions
- Multiple inputs and outputs are allowed
- It can be easier ingested by operationals (e.g.: S2S4E H2020 project)
- By using multiApply, you have half of the work done to publish your own function in the department R packages

startR introduction

How was startR born?



Big data

- Higher resolution in all dimensions
- Various kinds of data required

Complex analysis

- bootstrapping
- machine learning model
- etc.

Issues

- Longer data loading and processing time
- Limited memory space in local machine

startR feature

- ★ An R package tailored for big multi-dimensional data retrieval and processing
- ★ Apply multiApply paradigm
- ★ Automatic chunking of data set and parallel distributed data-processing on HPCs
- ★ Highly flexible according to the data structure and users' needs
- ★ Pre-processing: data transformation or reordering/merging/splitting dimension before performing analysis
- ★ Easy to reuse scripts due to the clear workflow
- ★ Use ecFlow workflow manager for job distribution and monitoring on HPC
- ★ Acceptable data format: netCDF for now, but may be available for others in the future

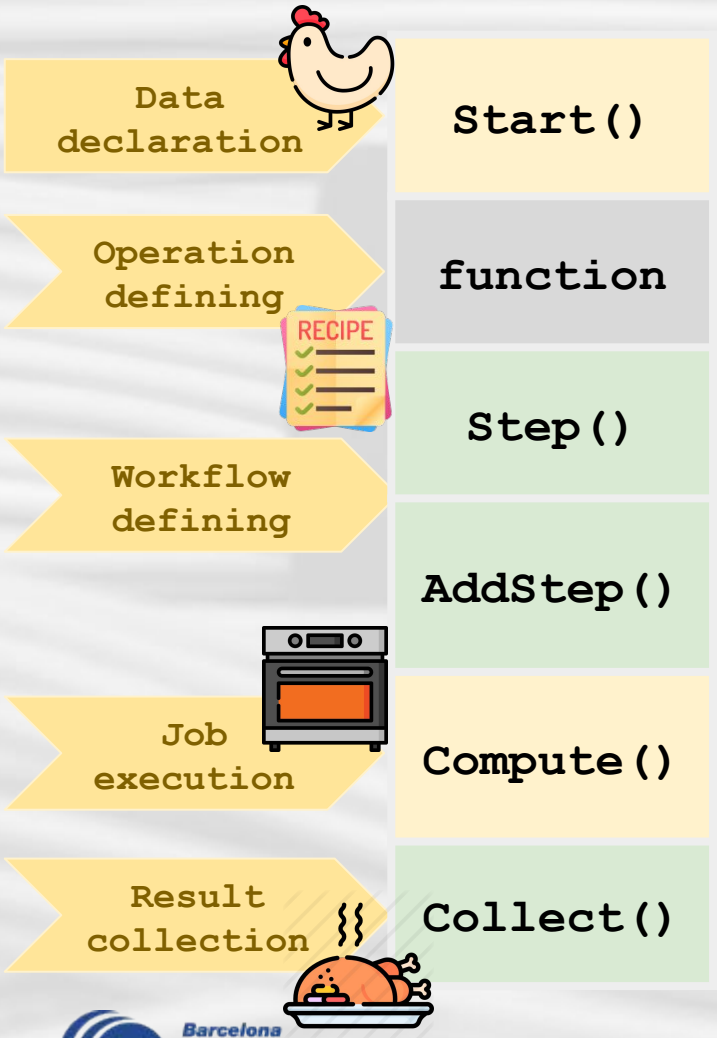
startR functions

`Start()`	Declare the data to be processed
`Step()`	Specify the operation to be applied to the data
`AddStep()`	
`Compute()`	Do chunking, specify the machine and its configuration for job employment, and trigger the execution
`Collect()`	Collect the results of background execution

And other helper functions.

startR workflow

Following the startR framework, users can create an analysis in a concise script with all the information needed, including:



1. Declare the data sources and the required file/inner dimensions
2. Define the data processing operation to be applied
3. Combine the elements from the previous steps to define the workflow
4. Trigger the job execution and set up the configuration for the machine used for data processing
5. Collect the results when the execution is done

startR workflow

Data declaration | Operation defining | Workflow defining | Job execution | Result collection

```
repos <-
'/esarchive/exp/ecmwf/system5_m1/monthly_mean/$var$_f6h/$var$_$sdate$.nc'

data <- Start(dat = repos,
              var = 'tas',
              sdate =  c('20170101', '20170201'),
              ensemble = indices(1:50),
              time = 'all',
              latitude = values(list(lat.min, lat.max)),
              longitude = values(list(lon.min, lon.max)),
              …,
              retrieve = FALSE)
```

→ data source

file dimension

inner dimension

Parameters for pre-processing, metadata, and definition etc.

→ *Create a pointer to data repository*

retrieve = TRUE

retrieve = FALSE

startR workflow

Data declaration → Operation defining → Workflow defining → Job execution → Result collection

Start() parameters

[define dimension]
pattern_dims
metadata_dims
path_glob_permissive
return_vars
synonims
*_depends
*_across
*_var

[reshape]
merge_across_dims
merge_across_dims_narm
split_multiselected_dims

[interpolate]
transform
transform_params
transform_vars
transform_extra_cells
apply_indices_after_transform

[helper function]
(no need to change in theory)
file_opener
file_var_reader
file_dim_reader
file_data_reader
file_closer
selector_checker

[operation]
num_procs
silent
debug

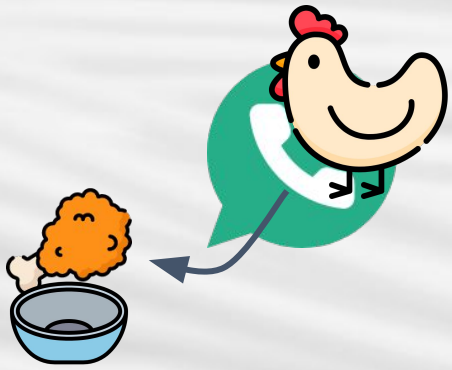
startR workflow

Data declaration | Operation defining | Workflow defining | Job execution | Result collection

- Define the operation in the **R function form**.
- The operation is only for **essential dimension** not the whole data, which is the same concept as multiApply.
- The output size should be small enough to fit in the workstation.

It can be as simple as one function:

```
fun <- function(x) {
    a <- apply(x, 2, mean)
    dim(a) <- c(time = length(a))
    return(a)
}
```



workstation

Or a complicated user-defined operation:

```
stratify_atomic <- function(field, MJO, season = c("JFM", "OND"), lag = 0, ampl = 2,
relative = TRUE, signif = 0.05) {
      # Arrange wind in form (days) to match MJO
      nmonths <- dim(field)[3]
      field <- aperm(field, c(1, 2, 4, 3))
      dim(field) <- c(31 * nmonths)
      if(season == "JFM") {
          daysok <- rep(c(rep(TRUE, 31), rep(TRUE, 28),
                          rep(FALSE, 3), rep(TRUE, 31)), nmonths / 3)
      } else if (season == "OND") {
          daysok <- rep(c(rep(TRUE, 31), rep(TRUE, 30),
                          rep(FALSE, 1), rep(TRUE, 31)), nmonths / 3)
      }
      field <- field[daysok]
      dim(field) <- c(days = length(field))

      if(dim(field)[1] != dim(MJO)[1]) {
          stop("MJO indices and wind data have different number of days")
      }

      idx <- function(MJO, phase, ampl, lag){
          if(lag == 0) {
              return(MJO$phase == phase & MJO$amplitude > ampl)
          }
          if(lag > 0) {
              return(dplyr::lag(MJO$phase == phase & MJO$amplitude > ampl,
                                lag, default = FALSE))
          }
          if(lag < 0) {
              return(dplyr::lead(MJO$phase == phase & MJO$amplitude > ampl,
                                 - 1 * lag, default = FALSE))
          }
      }
      strat <- plyr::laply(1:8, function(i) {
                                idx2 <- idx(MJO, i, ampl, lag)
                                if (relative) {
                                      return(mean(field[idx2]) / mean(field) - 1)
                                } else {
                                      return(mean(field[idx2]) - mean(field))
```

(Created by Llorenç)

startR workflow

Data declaration | Operation defining | **Workflow defining** | Job execution | Result collection

```
step <- Step(fun = fun,
             target_dims = c('ensemble'),
             output_dims = NULL)

wf <- AddStep(data, step, ...)
```

Additional parameters for the previous defined function

Which dimensions the operation performs on?

Which dimensions of output are expected?

Data dimension

dat	var	sdate	ensemble	time	latitude	longitude
1	1	1	51	7	256	512

```
fun <- function(x) {
  mean(x)
}
```

startR workflow

Data declaration | Operation defining | Workflow defining | **Job execution** | Result collection

```
res <- Compute(wf,
               chunks = list(latitude = 2,
                             longitude = 2),
               threads_load = 2,
               threads_compute = 4,
               cluster = list(queue_host = 'nord3',
                              queue_type = 'lsf',
                              …),
               ecflow_suite_dir = '/home/Earth/user_id/startR_local/',
               wait = TRUE
               )
```

The workflow defined at previous steps

Define chunking. Ensure each chunk size fits in the RAM memory module of HPC

Only needed for remote execution

startR workflow

Data declaration | Operation defining | Workflow defining | Job execution | Result collection

Use Collect() to collect and combine the results in the workstation if Compute() is on HPCs and its parameter 'wait = FALSE'.

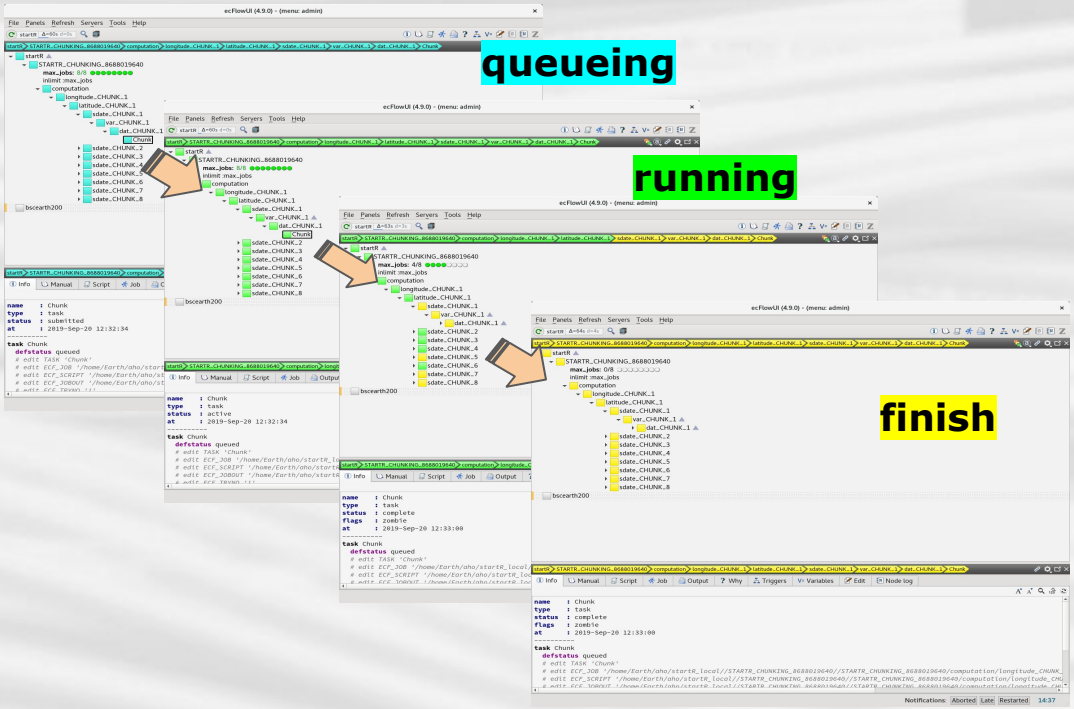
```
res <- Compute(wf,
               chunks = list(latitude = 2,
                             longitude = 2),
               cluster = list(queue_host = 'nord3',
                              queue_type = 'lsf',
                              …),
               ecflow_suite_dir = '/home/Earth/user_id/startR_local/',
               wait = FALSE
               )

saveRDS(res, file = 'test_collect.Rds')
**Now you can close the R console and come back later**

collect_info <- readRDS('test_collect.Rds')
result <- Collect(collect_info, wait = TRUE)
```

Store the descriptor of the execution

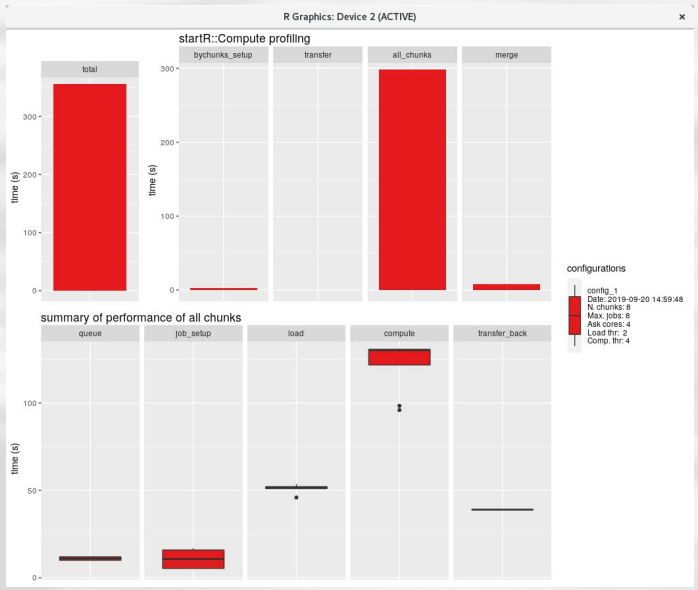
startR workflow (bonus)

Data declaration → Operation defining → Workflow defining → Job execution → Result collection

Monitoring the execution on **ecFlow UI**



Profiling the execution



Other packages overview

Department R packages

S2dverification/ s2dv	For 'seasonal to decadal' (s2d) climate forecast verification, but it can also be used in general climate analysis.
ClimProjDiags	A set of tools to compute metrics and indices for climate analysis. Also some formatting functions to facilitate the analysis.
CSTools	Forecast calibration, bias correction, statistical and stochastic downscaling, optimal forecast combination and multivariate verification.

↑ All of them apply the multiApply paradigm. → Easy to work with startR!

easyNCDF	A set of wrappers for the ncdf4 R package to simplify and extend its capabilities into/from multi-dimensional R arrays.

[Examples]

- ClimProjDiags::**WeightedMean**
  Calculate spatial area-weighted average of multidimensional arrays.
- CSTools::**CST_RainFARM**
  RainFARM stochastic precipitation downscaling of a CSTools object.
- s2dv::**PlotEquiMap**
  Plot a two-dimensional variable on A cylindrical equidistant projection.

s2dv

s2dv is the advanced version of package 's2dverification'. It is intended for 'seasonal to decadal' (s2d) climate forecast verification, but it can also be used in other kinds of forecasts or general climate analysis.

Data retrieval and formatting

- Load
- Reorder
- InsertDim
- LeapYear
- ToyModel

Basic statistics

- Clim
- Eno
- MeanDims
- Season
- Trend
- Ano
- Smoothing
- Composite

Skill score

- Corr
- Regression
- RMS
- RMSSS
- RandomWalkTest
- Persistance

Configuration

- ConfigApplyMatchingEntries
- ConfigEditDefinition
- ConfigEditEntry
- ConfigFileOpen
- ConfigShowSimilarEntries
- ConfigShowTable

Indices

- AMV
- GMST
- GSAT
- SPOD
- TPI

Plotting

- AnimateMap
- ColorBar
- PlotClim
- PlotEquiMap
- PlotAno
- PlotLayout
- PlotMatrix
- PlotSection
- PlotStereoMap

More functions will be transferred from s2dverification to s2dv.

ClimProjDiags

The package contains a set of tools to compute metrics and indices for climate analysis.

Formatting functions

DailyAno(): Daily anomalies
SeasonSelect(): Selects a season from daily data for multidimensional arrays
SelBox(): Selects spatial region
Subset(): Subsets an N-dimensional array (along, indices, drop parameters)
WeightedMean(): Calculate spatial area-weighted average

Vignettes

- Anomaly agreement
- Diurnal temperature range indicator
- Extreme indices t90p, t10n, rx5days, cdd, wx
- Heat and coldwaves duration

Computing Indicators

AnoAgree(): Percentage of anomalies which agrees with the sign of the mean anomaly
Climdex(): Wrapper to compute ETCCDI* climate change indices
CombineIndices(): Combine weighted indices
DTRIndicator(): Diurnal temperature range indicator
DTRRef(): Diurnal temperature range
Extremes(): Sum of spell lengths exceeding daily threshold
Threshold(): Daily thresholds based on quantiles
WaveDuration(): Heat and cold waves duration

*Expert Team on Climate Change Detection and Indices

CSTools

The package contains process-based methods for forecast calibration, bias correction, statistical and stochastic downscaling, optimal forecast combination and multivariate verification, as well as basic and advanced tools to obtain tailored products. This package was developed in the context of the ERA4CS project MEDSCOPE and the H2020 S2S4E project.

Basic functions

- CST_Load
- CST_Anomaly
- CST_SaveExp
- CST_SplitDim
- CST_MergeDims
- s2dv_cube
- as.s2dv_cube

Correction

- CST_BiasCorrection
- CST_Calibration
- CST_QuantileMapping
- CST_BEI_Weighting
- BEI_PDFBest
- CST_CategoricalForecast
- CST_DynamicalBC

Downscaling

- CST_Analogs
- CST_RFTemp
- CST_RainFARM
- CST_RFSlope
- CST_RFWeights
- CST_ADAMONT
- CST_AnalogsPredictors

Evaluation

- CST_MultivarRMSE
- CST_MultiMetric

Plotting functions

PlotMostLikelyQuantileMap
PlotForecastPDF PlotPDFsOLE
PlotCombinedMap PlotTriangles4Categories

Classification

CST_WeatherRegimes WeatherRegimes
CST_RegimeAssign RegimeAssign
CST_EnsClustering CST_MultiEOF

Shiny app - Introduction

Shiny is an R package that makes it easy to build interactive web apps directly from R.

The benefits of shiny app:

- Examine the figures quickly and easily.
- Display the figures together in one web page. Easy for discussion and sharing.
- Organize the figures in a constructive and tidy way.
- Detect the errors (of data, figures, and also storage) more easily.
- Highly interactive with users. Full control of what you want to see.
- Highly flexible to individual's needs. I.e., layout design, selection panel, tabs, etc.
- Incorporate CSS and HTML code which facilitate customization.

Shiny app - cp_shiny-figure

cp_shiny-figure

This Shiny app is created for visualization and monitoring the BSC-ES experimental/diagnostic data. It demonstrates the figures of the diagnostics generated by ESMValTool and mapgenerator.

URL (internal; need to use VPN)
https://earth.bsc.es/shiny/cp_shiny-figure/

Figure storage structure

/esarchive/exp/ecearth/a32p/plots/... **Exp ID**

Table
- atmos
- land
- ocean
- ocnBgchem
- sealce

Variable
- heatc0-300m
- heatcsum0-300m
- maxmoc26N
- maxmoc45N
- mlotst
- msftbarot
- msftyz
- sos
- tos
- zos

Plot type
- climantarctic_tos_xxx.png
- climarctic_tos_xxx.png
- climglobal_tos_xxx.png

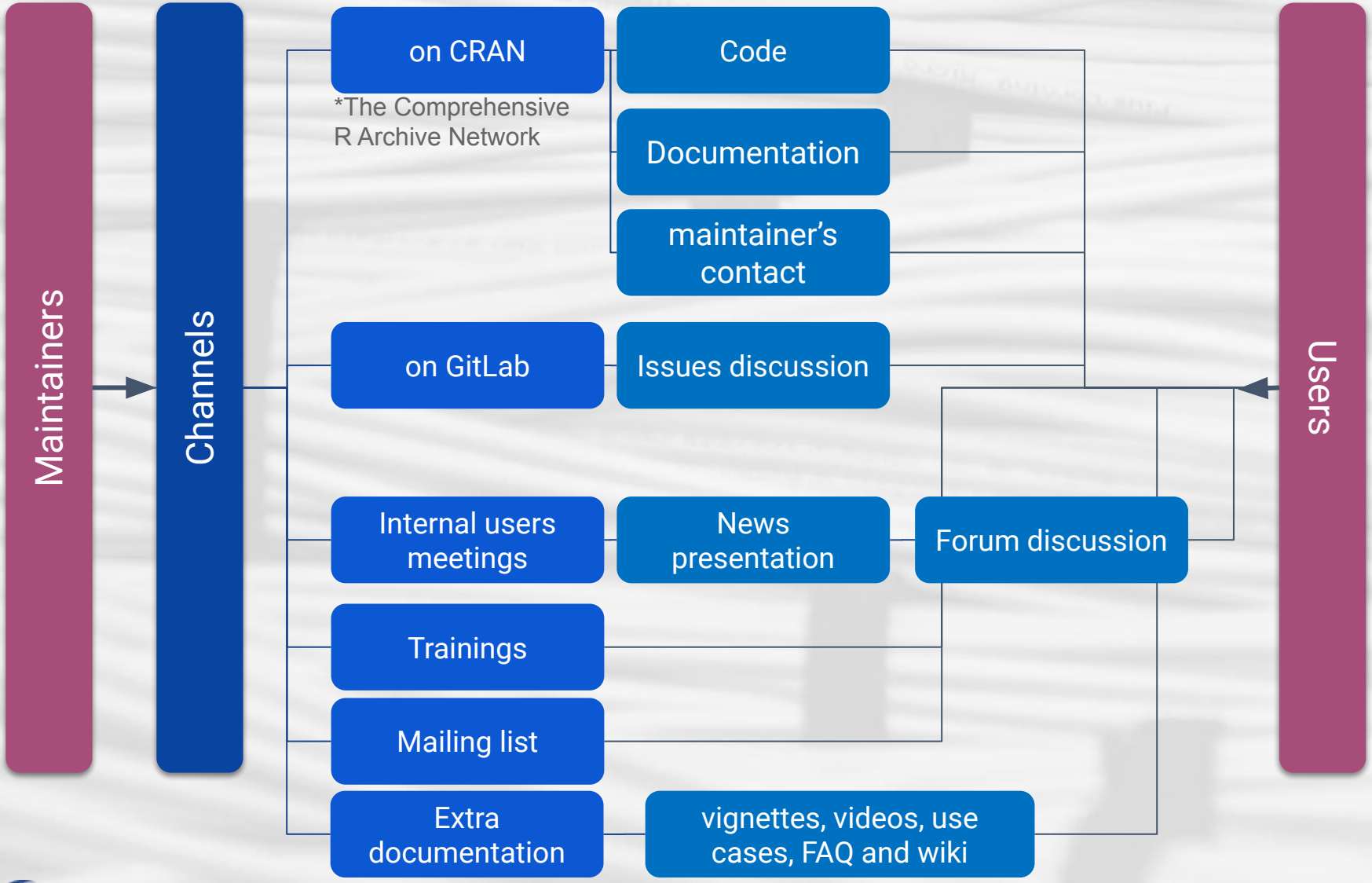
Other example

https://earth.bsc.es/shiny/C3S_34c/ (Created by Carlos Delgado)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Documentation & Useful links

Where to find documentation and answers?



Documentation & Useful links

multiApply	**CRAN:** https://CRAN.R-project.org/package=multiApply **GitLab:** https://earth.bsc.es/gitlab/ces/multiApply
startR	**CRAN:** https://CRAN.R-project.org/package=startR **GitLab:** https://earth.bsc.es/gitlab/es/startR

Other tools and channels

- **Department ES-BSC wiki: (Find information of other packages)** https://earth.bsc.es/wiki/doku.php?id=tools:Rtools&s[]=Rtools
- **Find all the functions here: Department R tool function list**
- **cp_shiny Gitlab:** https://earth.bsc.es/gitlab/es/cp_shiny
- **Mailing list: http://mailman3.bsc.es/postorius/lists/earth-rtools.bsc.es/ or contact us to join.**
  - **R-related news and announcements**
  - **Be aware of the R user meetings**

Examples

Examples
1.multiApply in CALIOPE-Urban

multiApply in CALIOPE-urban

1. Running CALIOPE-Urban took too long to become an operational
2. One bottleneck was found in a module coded in R: it took **more than 1 hour**
3. A data sample to run the problematic module was provided to CES
4. We did a profile of the code to detect the most problematic points

Profile of Urban background sheme step in CALIOPE-urban:
https://earth.bsc.es/gitlab/ac/caliope-urban/uploads/df563c53d82d35dccd5c131e7606dfef/profile.html

CALIOPE-Urban to become an Operational

multiApply in CALIOPE-urban

1. Running CALIOPE-Urban took too long to become an operational
2. One bottleneck was found in a module coded in R: it took **more than 1 hour**
3. A data sample to run the problematic module was provided to CES
4. We did a profile of the code to detect the most problematic points

Solution: Substitute the for loops by Apply by doing a wrapper function:

```
bg <- Apply(list(met, hours), margins = 'time',
            fun = estimate_bg_levels, ncores = 4)$output1
```

CALIOPE-Urban to become an Operational

multiApply in CALIOPE-urban

1. Running CALIOPE-Urban took too long to become an operational

2. One bottleneck was found in a module coded in R: it took **more than 1 hour**

3. A data sample to run the problematic module was provided to CES

4. We did a profile of the code to detect the most problematic points

   - The code speeds-up **53 % when running for 1 cell.**
   - For multiple cells, the new version takes about **1/3 of the time** than the original version
   - The entire domain (22x20 cells = 440 cells) using the **4 cores** in the WS and it took about **14 minutes**
   - In Nord3 using **16 cores**: **8 min 30 sec**

5. Look for the next bottleneck!

CALIOPE-Urban to become an Operational

Examples
2.startR and Monarch DUSTCIim simulations

startR and Monarch DUSTClim simulations

How to read the dataset?

https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/usecase/ex1_12_rotated_coordinates.R

Using startR

```
library(startR)

path <-
'/esarchive/oper/thredds-dust/monarch-dustclim/3hourly/$var$-av_an/$var$_$date$03_av_an.nc'

date <- c('20131229', '20131230')
# two temporal dimensions: one for days and another four hours
data_split <- Start(dataset = path,
            var = 'od550du',
            lev = 'all',
            date = date,
            time = 'all',
            rlat = 'all',
            rlon = 'all',
            return_vars = list(lev = NULL, time = NULL,
                               rlat = NULL, rlon = NULL,
                               lat = NULL, lon = NULL),
            retrieve = TRUE, num_procs = 1)
```

1) Create the path with labels between $

2) Define the values that the labels can take

3) Define the rest of the variables

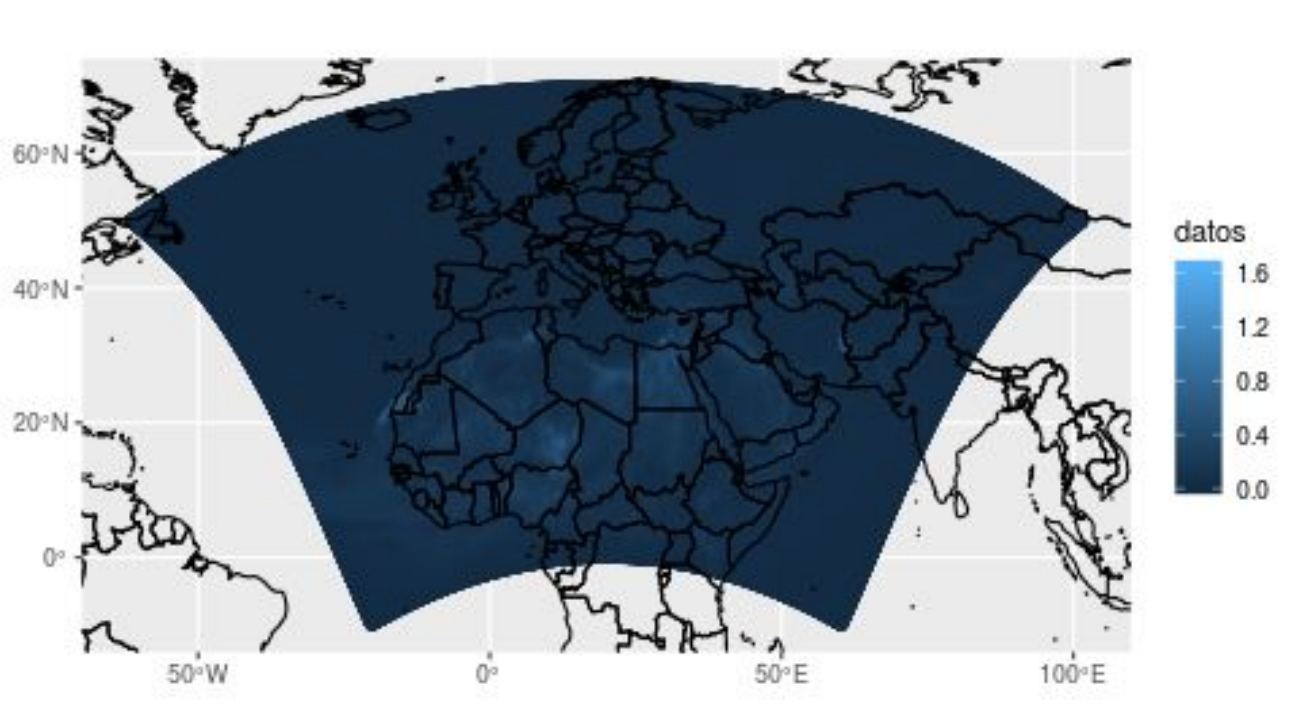
4) retrieve the data?

Note: We get an array object

```
> dim(data_split)
dataset  var    lev  date  time   rlat   rlon
      1    1      1     2     8    701   1021
```

startR and Monarch DUSTClim simulations

How to read the dataset?
https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/usecase/ex1_12_rotated_coordinates.R



Note: Using ggplot2 R package, plotting rotated coordinates is possible and customizable.

The **data size** is ~87.4 Mb for 2 days in 3 hourly frequency.

Multiplying

One year is ~2 GB.

Muyltiplying

20 years is ~ 40 GB.

Solution:
To perform analysis on the full period, we can split our data in pieces → **Chunking**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

startR and Monarch DUSTClim simulations

1. Define the pointer to the data source

```
library(startR)

path <-
'/esarchive/oper/thredds-dust/monarch-dustclim/3hourly/$var$-av_an/$var$_$year$$month$
$day$03_av_an.nc'

data <- Start(dataset = path,
              var = 'od550du',
              year = as.character(2007:2016),
              month = sprintf('%02d', 1:2),
              day = sprintf('%02d', 1:2),
              time = 'all',
              rlat = indices(100:200),
              rlon = indices(200:300),
              return_vars = list(time = NULL,
                                 rlat = NULL, rlon = NULL,
                                 lat = NULL, lon = NULL),
              retrieve = FALSE, num_procs = 1)
```

1) Create the path with labels between $

2) Define the variables in labels and inner dimensions

3) retrieve the data?

Note: We get a pointer

```
attributes(data_split)$Dimensions
dataset  var   year  month   day  time   rlat   rlon
      1    1     10     12    31     8    701   1021
```

startR and Monarch DUSTClim simulations

2. Analysis (function) and workflow

```
Statistics <- function(data, probs = c(0.25, 0.95)) {
  prom <- mean(data, na.rm = TRUE)
  perc <- quantile(data, probs = probs)
  sd <- sd(data, na.rm = TRUE)
  res <- array(c(prom, perc, sd), c(stats = 4))
  return(res)
}
## Test the function with Apply:
#res <- Apply(list(data), target_dims = c('month', 'day', 'time'), fun = Statistics, ncores = 4)


step <- Step(fun = Statistics,
             target_dims = c('month', 'day', 'time'),
             output_dims = 'stats')

workflow <- AddStep(data, step)
```

4) Define the function

*Test

5) choose dimensions

6) create the workflow

Reminder

attributes(data_split)$Dimensions

dataset	var	year	month	day	time	rlat	rlon
1	1	10	12	31	8	701	1021

startR and Monarch DUSTClim simulations

3. Define the HPC and submit

```
#----------modify according to your personal info---------
  queue_host = 'nord3'   #your own host name for power9
  temp_dir =  '/gpfs/scratch/bsc32/bsc32339/startR_hpc/'
  ecflow_suite_dir = '/home/Earth/nperez/startR_local/'  #your own local directory
#-----------------------------------------------------------
  res <- Compute(workflow,
                 chunks = list(year= 10),
                 threads_load = 1,
                 threads_compute = 4,
                 cluster = list(queue_host = queue_host,
                                queue_type = 'lsf',
                                extra_queue_params = list('#BSUB -q bsc_es'),
                                cores_per_job = 4,
                                temp_dir = temp_dir,
                                polling_period = 10,
                                job_wallclock = '01:00',
                                max_jobs = 10,
                                bidirectional = FALSE),
                 ecflow_suite_dir = ecflow_suite_dir,
                 wait = TRUE)
```

7) User details

8) Define the pieces of data

9) Define the cluster

Note: You need ecFlow to run it on an HPC. On Hands-on section, we will configure it and you will have it available forever.

startR and Monarch DUSTClim simulations

4. Result

```
> dim(res$output)
  stats dataset     var    year    rlat    rlon
      4       1       1      10     101     101

>range(res$output1[1,1,1,1,,])   # mean
[1] 0.4295594 1.5691717

> range(res$output1[2,1,1,1,,])  # percentile 0.25
[1] 0.05656383 0.80216140

> range(res$output1[3,1,1,1,,])  # percentile 0.9
[1] 0.8826941 4.4070848

> range(res$output1[4,1,1,1,,]) # standard deviation
[1] 0.1780865 1.4211329
```

Note: You need ecFlow to run it on an HPC. On Hands-on section, we will configure it and you will have it available forever.

startR and Monarch DUSTClim simulations

Chunking

https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/usecase/ex1_12_rotated_coordinates.R

1. Define the analysis and the workflow

```
library(startR)

path <-
'/esarchive/oper/thredds-dust/monarch-dustclim/3hourly/$var$-av_an/$var$_$year$$month$
$day$03_av_an.nc'


data_split <- Start(dataset = path,
                    var = 'od550du',
                    year = as.character(2012:2013),
                    month = sprintf('%02d', 1:12),
                    day = sprintf('%02d', 1:31),
                    time = 'first',
                    rlat = indices(400:450),
rlon = indices(100:200),
return_vars = list(time = NULL,
                      rlat = NULL, rlon = NULL,
                      lat = NULL, lon = NULL),
          retrieve = T, num_procs = 1)
```

1) Create the path with labels between $

2) Define the values that the labels can take

3) Define the rest of the variables

4) retrieve the data?

Hands-on

Hands-on
1. Configuring ecFlow

ecFlow configuration

Documentation
https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/practical_guide.md#configuring-startr

Passwordless to Nord 3

STEP 1: Open a terminal on nord3:

- **have you needed to enter a password?**
  - **No → you can skip this step**
  - **Yes → follow this slide**

1. Open a new terminal in your workstation home, run **ls .ssh**

   > do you see a file called **id_rsa.pub**?
   - No → Run **ssh-keygen -t rsa**
     Copy it, and check if the file appear now
   - Yes → open the file, and copy the key
2. Go to the terminal on Nord3, run **ls .ssh**
   > do you see a file called **authorized_keys**?
   - No → Run **mkdir .ssh/aurtorized_keys**
   - **Yes → Go to next line**

   > Open the file vim .ssh/autorized_keys
   > Copy the key → Press 'i', then, CRTL + V (save with ':wq')
3. Close the Nord3 terminal and open a new one.

Have you needed to enter a password?

Note: If you don't have a Nord 3 account go to the employees portal and do a request.
https://webapps.bsc.es/employee/

ecFlow configuration

Documentation
https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/practical_guide.md#configuring-startr

Nord3 Nickname

STEP 2: When connecting to Nord 3,

> Do you use bsc32XXX@nord3.bsc.es?

- **No, I use a shortcut (e.g.: nord3) → you can skip this step**
- **Yes, I use that line → follow this slide**

1. Go to the terminal in your workstation home, run **ls .ssh**

   > do you see a file called **config**?
   - No → Run **mkdir .ssh/config**
   - Yes → Go to next line

   > Open the file **vim .ssh/config**
   > Press 'i', then, add the following content

```
Host nord3
    HostName nord3.bsc.es
    User bsc32XXX
    IdentityFile ~/.ssh/id_rsa
    ForwardX11 yes
```

   > Edit it with your user (save with ':wq')

2. Close the Nord3 terminal and open a new one by typing **ssh nord3**
   Has it worked without needing a password?

ecFlow configuration

Documentation

https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/practical_guide.md#configuring-startr

Earth modules on Nord 3

STEP 3:

> Do you have the Earth modules installed in Nord3?

- **Yes, I use them → you can skip this step**
- **No, I don't know what is that → follow this slide**

1. Go to the terminal in Nord 3, open .bashrc → Run **vim .bashrc**

   > do you see the following lines?

```
if [ $BSC_MACHINE == "nord3" ]; then
  source /gpfs/projects/bsc32/software/modules_supp/init_modules.sh
  export PATH=/apps/modules/3.2.10/Modules/3.2.10/bin/:$PATH
  module use /gpfs/projects/bsc32/software/suselinux/11/modules/all
  module unuse /apps/modules/modulefiles/applications
/apps/modules/modulefiles/applications_bis /apps/modules/modulefiles/compilers
/apps/modules/modulefiles/tools /apps/modules/modulefiles/libraries
/apps/modules/modulefiles/environment /apps/modules/PRACE
fi
```

   - No → Press 'i', then, CRTL + V (save with ':wq')
   - Yes → Next slide

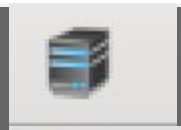
ecFlow configuration

Documentation
https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/practical_guide.md#configuring-startr

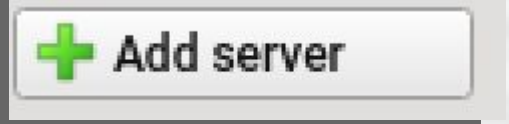
STEP 4: Open a workstation terminal using `ssh -XY`
> Load the following modules `module load ecFlow R`
> Open ecFlow by running `ecflow_ui &`

Open ecFlow and configure it

1. On the top of the window, look for this icone and click it

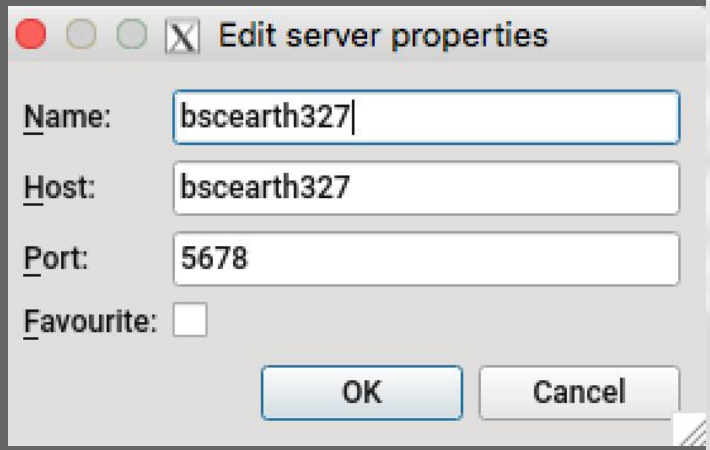


2. A new box will open, look for 'Add server' and click it



3. Fill the new box with your information

Your info →
Your info →
Mandatory number →



4. Press 'OK'

Done! We can run a startR on Nord3

Hands-on
2. Running startR

startR and Monarch DUSTClim simulations

How to read the dataset?

https://earth.bsc.es/gitlab/es/startR/-/blob/master/inst/doc/usecase/ex1_12_rotated_coordinates.R

Using startR

```
library(startR)

path <-
'/esarchive/oper/thredds-dust/monarch-dustclim/3hourly/$var$-av
_an/$var$_$date$03_av_an.nc'

date <- c('20131229', '20131230')
# two temporal dimensions: one for days and another four hours
data_split <- Start(dataset = path,
            var = 'od550du',
            lev = 'all',
            date = date,
            time = 'all',
            rlat = 'all',
            rlon = 'all',
            return_vars = list(lev = NULL, time = NULL,
                               rlat = NULL, rlon = NULL,
                               lat = NULL, lon = NULL),
            retrieve = TRUE, num_procs = 1)
```

Explore the object to get:

- The coordinates, The time and the Variable units
  HINT: attributes(), names(), $Variables
- Can you load one more day?
- Can you load the data only for the 00 h
  HINT: 'first', indices(1)
- Can you load the data getting dimensions YEAR and Julianday?

Note: We get an array object

```
> dim(data_split)
dataset  var    lev  date  time   rlat   rlon
      1    1      1     2     8    701   1021
```

Thanks for your attention!
Questions and Answers

An-Chi Ho an.ho@bsc.es
Núria Pérez-Zanón nuria.perez@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación