# Adapting your code to CSTools guidelines

## After agreement with the coordinator of CSTools

Núria Pérez-Zanón
nuria.perez@bsc.es
Postdoctoral researcher at Barcelona Supercomputing Center


# 1. Understanding the workflow

**Open your mind:**
You may have already developed a script or sequence of functions doing your analysis, now, it's time to identify:

- necessary inputs,

- objective outputs and

- divide the process to identify the minimum function.

First, we will follow a top-bottom scheme to the identification part and then, we will adapt our code following a bottom-top scheme!

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

read files and return an object ‘s2dv_cube’

s2dv_cube

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a  vector
- $lat a vector
- $Variable … for instance ‘tas’, ‘psl’,...
- $Datasets
- $Dates
- $source_files
- ...

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

read files and return an object ‘s2dv_cube’

s2dv_cube

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a  vector
- $lat a vector
- $Variable … for instance ‘tas’, ‘psl’,...
- $Datasets
- $Dates
- $source_files
- ...

**Exercice 1**

Check the sample data included in CSTools
(Open an R session and run):

```
library(CSTools)
# maybe you need a previous step install.packages(“CSTools”)
class(lonlat_data$exp) # check the class
names(lonlat_data$exp) # visualize the names
lonlat_data$exp$Variable$varName # check the variable
dim(lonlat_data$exp) # see the size of your data
```

… could you visualize which is the model of this data?

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

read files and return an object ‘s2dv_cube’

… extra steps ...
CST_Anomaly
CST_Season

these intermediate steps will work on and return a ‘s2dv_cube’ object

CST_YourFun

Your CST function will work on a ‘s2dv_cube’ object and it can return a ‘s2dv_cube’ too

s2dv_cube

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a  vector
- $lat a vector
- $Variable … for instance ‘tas’, ‘psl’,...
- $Datasets
- $Dates
- $source_files
- ...

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

… extra steps ...
CST_Anomaly
CST_Season

CST_YourFun

The output depends on the objective of the function:

- to improve climate data
  (e.g.: downscaling, bias correction, calibration, …)
  or
- to compute statistics
  (e.g.: scores, verification, …)

*Note: plotting functions are different (they just need to work on arrays with named dimensions.

**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Variable … for instance ‘tas’, ‘psl’,...
- $Datasets
- $Dates
- $source_files
- ...

**Exercice 2**

Run the examples provided with functions CST_RainFarm (pg. 13), CST_MultiMetric (pg. 8) and PlotmostLikelyQuantileMap (pg. 21) from https://cran.r-project.org/web/packages/CSTools/CSTools.pdf to understand differences between outputs.

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

… extra steps ...
CST_Anomaly
CST_Season

```
CST_YourFun(data1, data2, method = 'method1', ...) {

        # Check the inputs

        # Compute    YourFun(data1$data, data2$data, data1$lon, method, …)

        # Re-format the output


}
```

**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a  vector
- $lat a vector
- $Variable … for instance 'tas', 'psl',...
- $Datasets
- $Dates
- $source_files
- ...

**Exercice 3**

Visualize the code of a CST function, you will be surprise of how short it is!
Open an R session and run:

```
library(CSTools)
CST_RainFARM

# Have you seen function RainFARM in the code?
```

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme

CST_Load

… extra steps ...
CST_Anomaly
CST_Season

CST_**YourFun**(data1, data2, method = 'method1', ...) {

```
YourFun(data1$data, data2$data, data1$lon, method, …) {
   # Check inputs
   Apply(
            data = list(data1$data, data2$data,...),
            target_dims = c('MyRequiredDimensions'),
            fun = .yourfun)
   # Re-format output (if needed)
}
```

**s2dv_cube**

it is an object containing the data and metadata:

- $data an **array with named dimensions**
- $lon a vector
- $lat a vector
- $Variable … for instance 'tas', 'psl',...
- $Datasets
- $Dates
- $source_files
- ...

**Exercice 4**

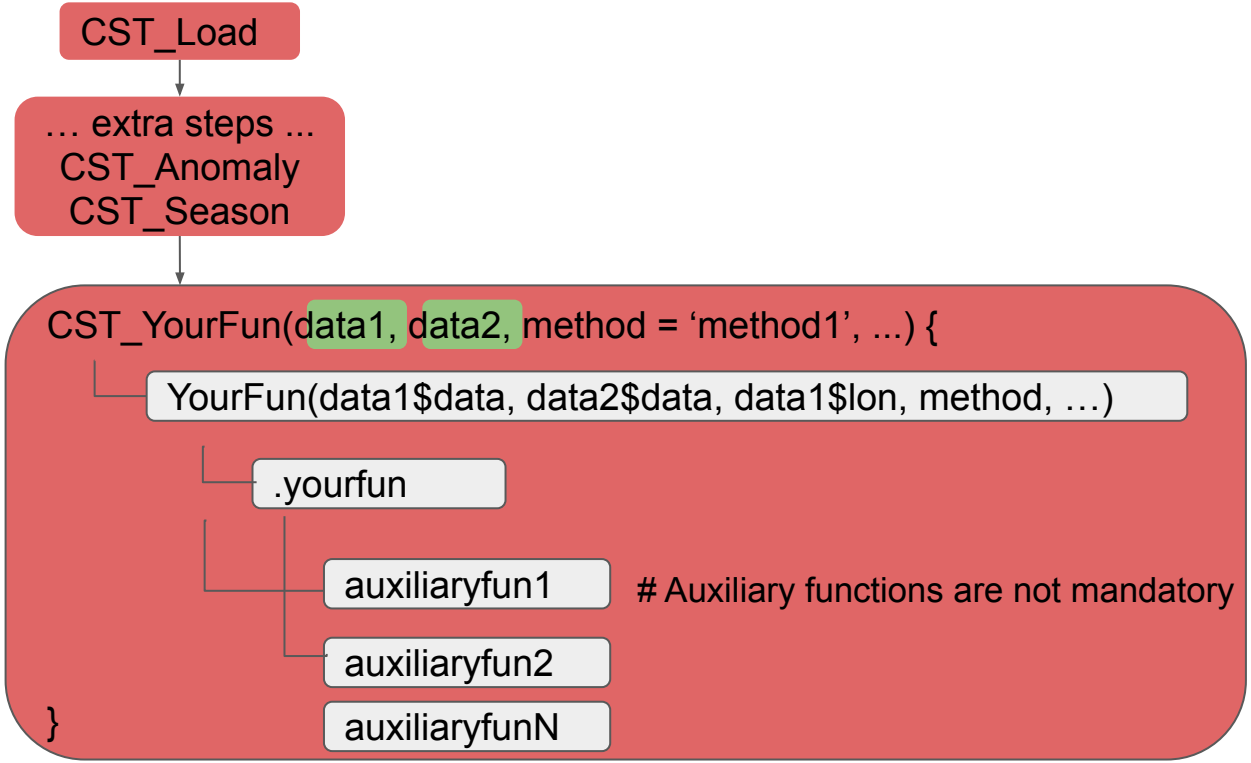
Visualize the code of a function working on arrays with named dimensions. Open an R session and run:

```
library(CSTools)
CSTools:::BiasCorrection

# Have you seen function
.sbc in the code?
```

# 1. Understanding the workflow

## 1.1 Identifying steps by a top-bottom scheme




**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Variable … for instance 'tas', 'psl',...
- $Datasets
- $Dates
- $source_files
- ...

We already know how create CST_YourFun and YourFun, we just need to follow the recipe.

Your objective is create **.yourfun**

# 1. Understanding the workflow

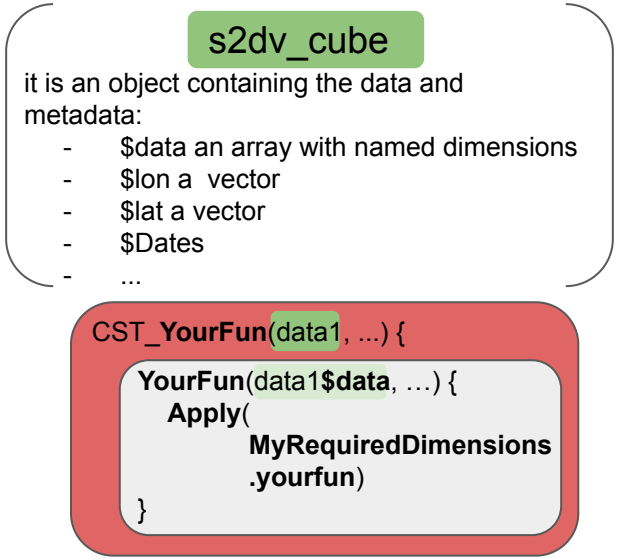
## 1.2 Coding by a bottom-top scheme

Now, it's time to understand your analysis/experiment and work in the **.yourfun**.

As this step depends on your functionality, we can just work with examples of other methods already available in the package.

However, we should try to create a useful lists of questions for future developers.

Please, if you have any suggestion, add a comment or contact nuria.perez@bsc.es.



# 1. Understanding the workflow

## 1.2 Coding by a bottom-top scheme

Before coding, it's important to know that CST_Load is able to:

- load simultaneously experimental and observational data
- load monthly and daily data
- select a region
- select a period
- regrid data (it uses cdo internally)
- adjust the number of members loaded
- adjust the start dates, etc.

**If your function was including any of this steps, please, avoid using them** and consider that your input data from data1$data contains the data already subset for your desired period and region since the user will adjust this when loading the data.

If you still need a intermediate step in your function to subset the data consider using: WeightedMean and SelBox form ClimProjDiags or Season from s2dverification or ask the coordinator of the package.

**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Dates
- ...

```
CST_YourFun(data1, ...) {
    YourFun(data1$data, …) {
        Apply(
            MyRequiredDimensions
            .yourfun)
    }
```

Ex:
X is large scale data
x is local scale data
option 1:

```
fun(X, region) {
    x <- SelBox(X, region)
    .fun(X, x)
}
```

option 2:

```
x <- Selbox(X, region)
fun(X, x) {
    .fun(X, x)
}
```

# 1. Understanding the workflow

## 1.2 Coding by a bottom-top scheme

Other functions already included in the package, or planned to be, are:

`CST_Anomaly` `CST_QuantileMapping`

Check them to avoid extra steps in your analysis.

Other functions already included are:

`CST_MultiMetric:` compares models skills correlation, RMS, RMSSS. `Exercice 6: Run the vignette Multi-model Skill Assessment`
`CST_Calibration`
`CST_BiasCorrection`

If you plan to add a new method, take a look at these functions and try to add a new option inside to be selected as a new option in parameter 'method'.

**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Dates
- ...

```
CST_YourFun(data1, ...) {
    YourFun(data1$data, …) {
        Apply(
            MyRequiredDimensions
            .yourfun)
    }
```

**Exercice 5**

Could you compare parameter 'method' in CST_Calibration function between CSTools version 1.0.1 and version 2.0.0?

You can visit the GitLab project and compare the code for the two tags: https://earth.bsc.es/gitlab/external/cstools/-/tags

# 1. Understanding the workflow

## 1.2 Coding by a bottom-top scheme

Now, you need to identify the first computation, in the minimum scale, that your analysis should do, for instance: correlation, sum, distance, …

Let me use the CST_Analogs as example.

This function aims is that, for a given experimental SLP field for one day on the synoptic scale ($exp_t$), it looks for the most similar spatial field in a period of observations ($obs_{t1-tN}$).

Translation: 'similar' could be minimum distance or correlation

The first step it to compute the 'distance' between $exp_t$ and each $obs_{t1-tN}$:
$obs_{t1}$ - $exp_t$ , $obs_{t2}$ - $exp_t$ , $obs_{t3}$ - $exp_t$, …. $obs_{tN}$ - $exp_t$
$d_1$, $d_2$, $d_3$, …. $d_N$
$cor_1$, $cor_2$, $cor_3$, …. $cor_N$
obs has dimension 'lat', 'lon' and 'time'
exp has dimension 'lat' and 'lon'
and we can consider two method for the selection criteria.

**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Dates
- ...

```
CST_YourFun(data1, ...) {
    YourFun(data1$data, …) {
        Apply(
            MyRequiredDimensions
            .yourfun)
    }
```

**Exercice 7**

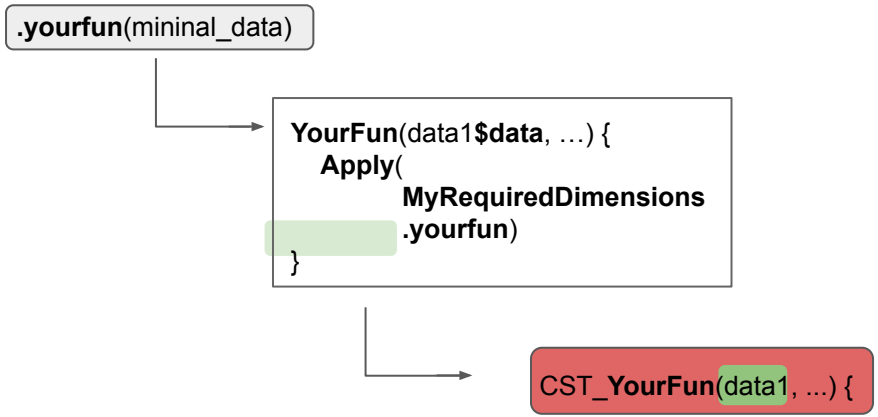
Look for function **.select** performing the computation of 'distance' and 'correlation', as explained in the box next to this, in CST_Analog.R.

# 1. Understanding the workflow

## 1.2 Coding by a bottom-top scheme

Simplifying, once you have writing the basic function you can keep the workflow by wrapping it with Apply from multiApply package.

You are ready to follow the bottom-top scheme:




**s2dv_cube**

it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Dates
- ...

```
CST_YourFun(data1, ...) {
    YourFun(data1$data, …) {
        Apply(
            MyRequiredDimensions
            .yourfun)
    }
```

**Exercice 8**

Create a function using Apply to compute the mean over latitudinal dimension in an array with dimensions lon, lat, time and member, by first defining the minimal function.

# 1. Understanding the workflow

## 1.2 Coding by a bottom-top scheme

**List of Questions**

- Can your function work with data already subset/regrided and avoid that step?
- Does your function needs anomalies or quantile mapping and they can be performed with CST_Anomaly or CST_QuantileMapping avoiding you that step?
- Could your function be merged as option in CST_MultiMetric, CST_Calibration or CST_BiasCorrection?
- Which is the minimum problem that you functionality perform? Could you identify the dimensions required for each element?

**s2dv_cube**

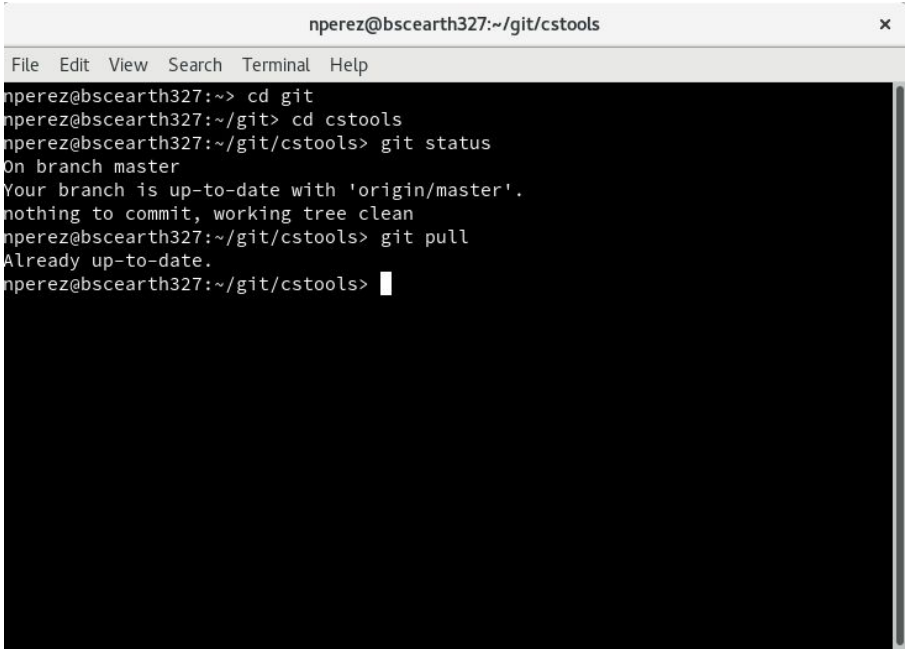
it is an object containing the data and metadata:

- $data an array with named dimensions
- $lon a vector
- $lat a vector
- $Dates
- ...

```
CST_YourFun(data1, ...) {
    YourFun(data1$data, …) {
        Apply(
            MyRequiredDimensions
            .yourfun)
    }
```

# 2. Understanding GitLab/git

**LOCAL**




**REMOTE**