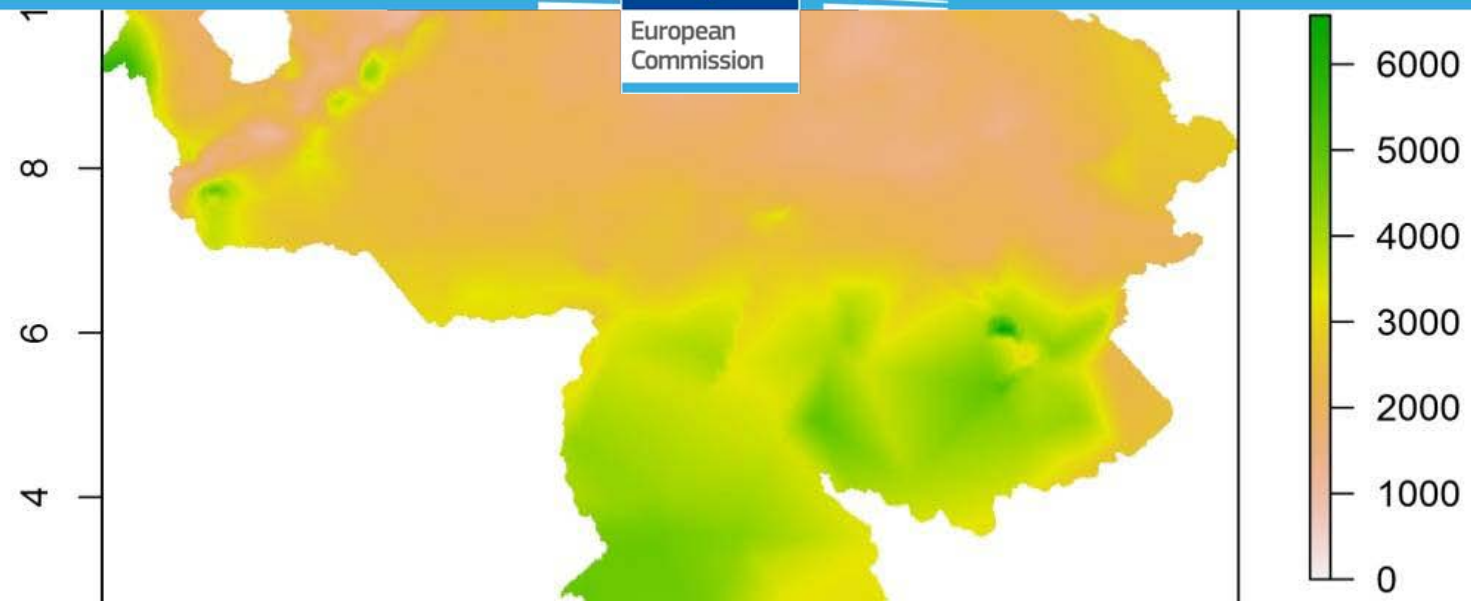


JRC TECHNICAL REPORTS

The REFRAN-CV Web Application

Technical Documentation
Install Instructions
Admin and User Guide

Michele Chinosi
Juan Arevalo Torres
César Carmona Moreno

June 24, 2013 – ver 1.0-alpha3

Joint Research Centre

European Commission
Joint Research Centre
Institute for Environment and Sustainability

Contact information
Cesar Carmona Moreno
Address: Joint Research Centre, Via Enrico Fermi 2749, TP 440, 21027 Ispra (VA), Italy
E-mail: cesar.carmona-moreno@jrc.ec.europa.eu
Tel.: +39 0332 78 9654

http://ies.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

This publication is a Reference Report by the Joint Research Centre of the European Commission.

JRC86571

Printed in Italy

Abstract: This document is intended to be complete technical reference documentation for the REFRAN-CV web application. It is divided into 4 parts. Part 1 describes the requirements for installing and executing the web application (also considering different Operating Systems) as well as results of the compatibility tests performed. It also offers a Quick Execution section to start using the web application in a minute. Part 2 presents the web application roadmap, its design principles stressing on compatibility with previous version and the reference description document, the structure of the application (with a detailed description of each module) and of the folders. Part 3 is completely dedicated to the R code, highlighting all the changes, the criticisms and corrections done with respect to the original version, with a Section dedicated to the test data used while building the application. Part 4 is for the future works list, being the application at this stage only an incomplete prototype, with a future suggested roadmap and other remarks. Conclusions close the document.

Table of Contents

Part 1: Requirements and Quick Start

1.1. REFRAN-CV Requirements

The minimum requirements to run the framework on all architectures are the following:

- A web server (the Apache HTTPD server[1] version 2.4.x is suggested)
- PHP[2] version 5.2 or above (PHP version ≥ 5.4 is suggested)
- The R[3] environment version 3.0.1 (May 2013)[4]
- An updated web browser (Mozilla Firefox or Google Chrome are suggested)

To develop and test the REFRAN-CV web application we used the following tools (they are all open source or available for free except for Microsoft Windows 7):

- Microsoft Windows 7[5]
- RStudio[6] (0.97.336)
- Netbeans[7] (7.3), a free open source IDE
- APIGen[8] for PHP to automatically generate PHP functions documentation
- jQuery[9] and jQueryUI[10] to add graphical effects to web page elements
- Tablesorter[11] (2.0.5) to display big tables in a fancy way
- ApacheFriends XAMPP[12] Portable Lite 1.8.1 for Windows which includes Apache HTTPD 2.4.3 and PHP 5.4.7
- NSIS[13] (Nullsoft Scriptable Install System) for Windows, a professional open source system to create Windows installers.

---

[1] http://httpd.apache.org/

[2] http://www.php.net/

[3] http://cran.r-project.org/

[4] Even if we started working using R version 2.15.2 (October 2012)

[5] http://windows.microsoft.com/en-US/windows7/products/home

[6] http://www.rstudio.com/

[7] http://netbeans.org/

[8] http://apigen.org/

[9] http://jquery.com/

[10] http://jqueryui.com/

[11] http://tablesorter.com/docs/

[12] http://www.apachefriends.org/en/xampp-windows.html

[13] http://nsis.sourceforge.net/Main_Page

For testing the framework we used the following machine:

- DeLL Optiplex 780
- PC Intel(R) Core(TM)2 Duo
- CPU E8400 @ 3.00 GHz
- RAM 8 GB
- Microsoft Windows7 Professional @ 64-bit operating system.

We noticed during tests there may be errors raised by the R environment if the there are too many data to process with respect to the equipment of the computer where you run the REFRAN-CV application from. We suggest using a high-performance computer to run the application, especially if you have huge amount of data. Rather, consider using a 64-bit architecture and replace the 32-bit R environment installed in REFRAN-CV as default with its 64-bit version. To do this, you can just follow these steps:

- Open the `xampp\htdocs\refran-cv\mockup\include\config.php` file and change the content of the `$RRootPath` variable with the new one, already present in `config.php` as comment (the default one is: `$RRootPath .='\\xampp\\R-3.0.1\\bin\\R.exe';`)
- Save the file and start again the application.

Moreover, for development and testing purposes a versioning system has been set up and used. So far it consists in a private free personal GIT[14] account provided by Assembla[15]. Due to development and security reasons this account will remain not publicly available until a different decision will be taken. In the meanwhile it could be possible to request an access with read-only permissions only by email to the author. Each request will be evaluated and if accepted the access will be granted.

Unix/Linux and Apple MacOS versions are under development. Being REFRAN-CV a web application, it can be run as it is in each environment having the minimum requirements listed above. The OS-depending part concerns the choice of the most appropriate version of both the web server and the R environment.

1.2. Browser Compatibility Test

All the pages have been tested using the W3C XHTML validator[16] and they result now W3C compliant. Also CSS files have been checked and all the code we developed in-house is W3C compliant. Nonetheless we could not put the W3C CSS badge because some CSS code imported from external projects (e.g. jQueryUI) are not actually W3C compliant. We successfully tested the framework on:

- Microsoft Windows 7 with XAMPP-PORTABLE 1.8.1 (Apache HTTPD 2.4.3, PHP 5.4.7)

[14] http://git-scm.com/

[15] https://www.assembla.com/home

[16] http://validator.w3.org/

    - Mozilla Firefox 21.0[17]
    - Google Chrome 27.0 (this is the suggested browser)
    - Microsoft Internet Explorer 8.0
    - Opera 12.15
- Unix/Linux
    - To be done
- Apple MacOS
    - To be done

1.3. The REFRAN-CV Package

The REFRAN-CV application has been developed and tested locally using the local stand-alone environment provided by the ApacheFriends XAMPP software. The Portable Lite version of XAMPP software allows having a full working environment within a single folder without the need of installing it. Hence, it is possible to move the whole environment simply by copying the root folder. This guarantees that a working version of the application can be moved across different computers and/or distributed being reasonably sure the final result looks the same. Thus, starting from this stage of development the REFRAN-CV web application is also provided as a self-contained package on a USB key or as a ZIP file, coming also with batch files to ease the automatic run and stop procedures. The content of the portable package is as follows:

- The `xampp` folder, which is the main root folder containing the web application files along with all the necessary software, in particular:
    - The Apache HTTPD server
    - The PHP language
- The R environment
- The `xampp-ExampleFiles` folder, with some example input data tested and verified
- The link to the PDF version of this document (a DOC version is also provided)
- The developed PHP functions documentation in the `refran-cv\mockup\PHPdoc` subfolder
- Two batch files[18]:
    - `START.bat` to start the environment and open the application into the default web-browser

[17] Sometimes Firefox could not access your file system to copy or modify files. This is probably due to the rights configuration of your browser or Windows account especially if you have not administrative rights. Please check with your system administrator or simply try using another browser.

[18] The Unix/Linux and Apple MacOS versions of batch files are still under development

- `STOP.bat` to stop the environment and shut-down the server

We are also able to provide the REFRAN-CV web application as an auto-installer package in two different versions:

- `REFRAN-CV-light` (~ 200 MB), which includes only the REFRAN-CV application and all the required software (XAMPP, R, PHP and batch files) and documentation
- `REFRAN-CV-full` (~ 352 MB), which includes also sample input/output data from Venezuela.

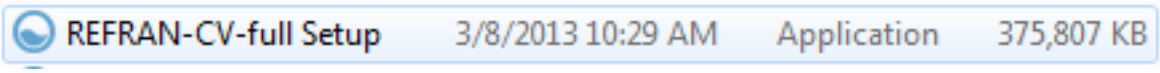
All the code developed for the REFRAN-CV application is released under the EUPL v1.1 licence (see Sections 2.3 and 2.3.2 for more details).

1.4. Quick Run

1.4.1. Run from the Installer

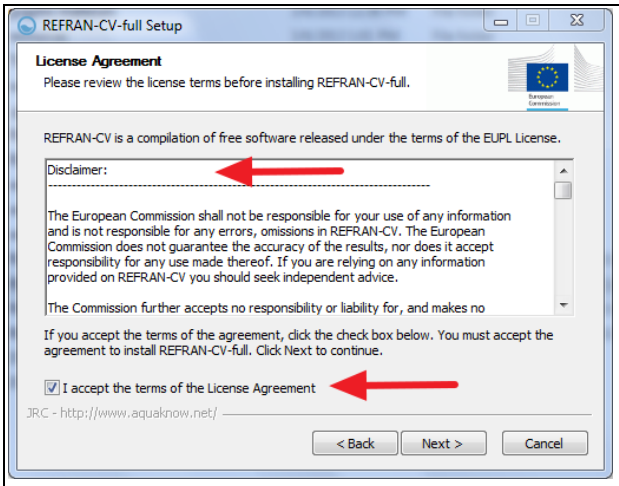
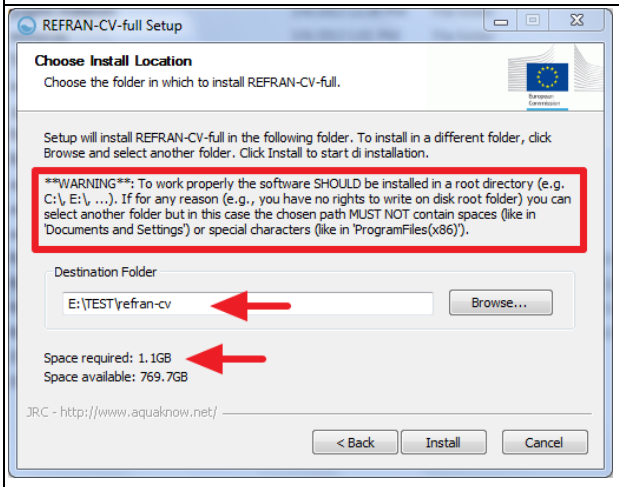
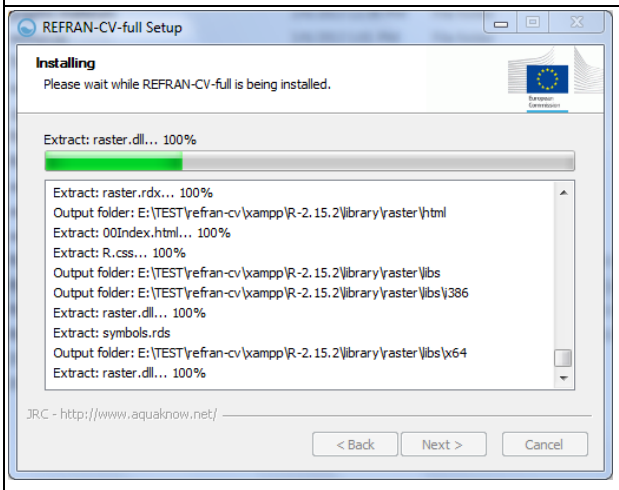
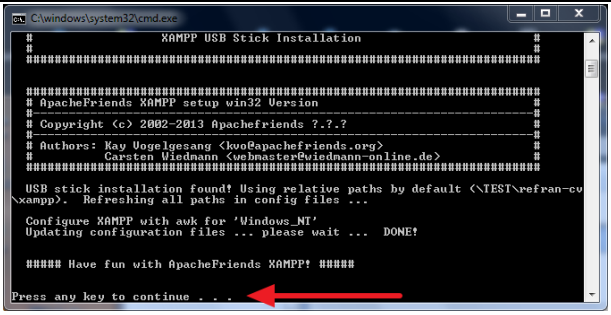
Using the installer (both the light and the full version) is probably the simplest way to start using REFRAN-CV. In order to run the REFRAN-CV web application starting from the installer simply follow these steps:

- Double-click on the executable



- Follow the instructions giving particular attention to the choice of the destination folder (see Table 1)

	This is the first window of the REFRAN-CV application installer.

	The second step requires the user to read the full disclaimer and the EUPL licence text and check the box at the bottom before going to the next step.
	This step is crucial for the installation procedure. The user must select an installation folder satisfying particular requirements: - The user must have enough rights to access the folder - There must be enough space left on the device - The folder path must not contain special characters or spaces.
	If all the gathered information are right, the installation process starts copying all the files into the destination folder.
	At the end of the copying process a Windows shell will open. The user has nothing to do here except waiting the "Press any key to continue..." message to appear and then press a key.

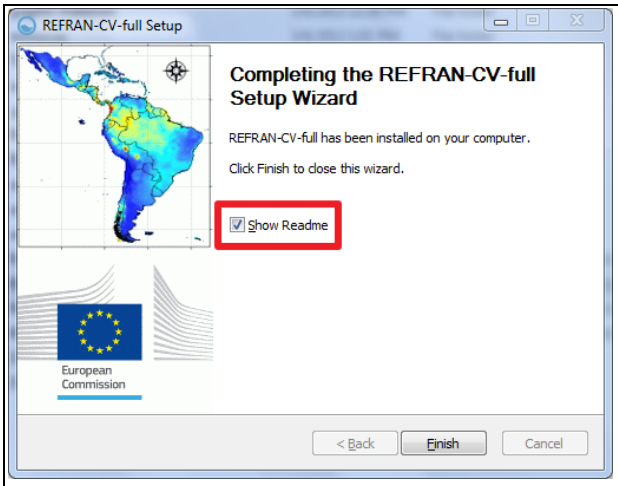
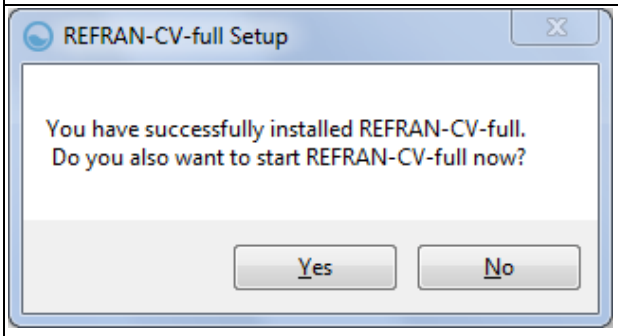
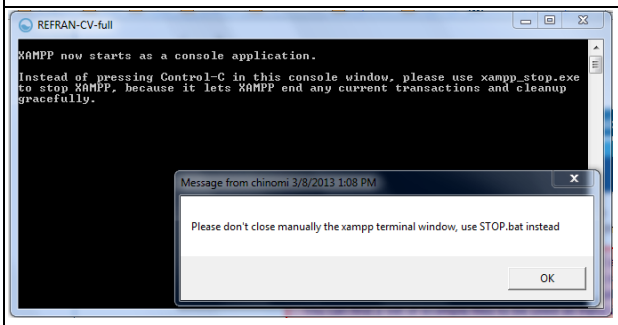
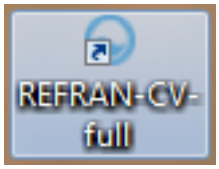
	At the end of the installation process the user is given the opportunity to read the README file.
	Once the installation procedure finishes, a window asks the user whether to run REFRAN-CV immediately or not. If you choose "Yes" after few seconds another window appears.
	You should NOT close this window by hand but you must keep this window opened, maybe in background. This window will disappear once you properly stop the REFRAN-CV application. You can close the warning message.

Table 1: The REFRAN-CV installation procedure

- At the end of the installation process you can find a shortcut on your Desktop to start the REFRAN-CV application.



- Now you can proceed to Section 1.6 to give a first look to the landing pages of the REFRAN-CV web application.

1.4.2. Run from an USB key

To run the REFRAN-CV web application starting from the USB key simply follow these steps:

- Insert the USB key and open it

- Click on the `START.bat` file

> ! A shell/terminal window could appear. Do not close it manually otherwise the web server will not work properly. Leave the window opened. It will be closed automatically at the end of the shutdown procedure.

- That's it! If all goes well your default web-browser should now be opened on the landing page of the application (see Section 1.6).
- Remember when you finish using the application to shut-down all the services by clicking on the `STOP.bat` file or using the special button on top of every web page.

1.4.3. Run the package from a computer

In order to run the REFRAN-CV web application from your computer – starting from an USB key or from a downloaded package – follow this procedure:

- Unzip the compressed archive in a root directory (e.g. `E:\`)
- It is crucial putting the `xampp` folder in the root of a disk or partition (e.g. `E:\xampp`) or in a folder whose path does not contain special characters or spaces.
- Click on the `START.bat` file (you should find a shortcut on your Desktop)

> ! A shell/terminal window could appear. Do not close it manually otherwise the web server will not work properly. Leave the window opened. It will be closed automatically at the end of the shutdown procedure.

- That's it! If all goes well your default web-browser should now be opened on the landing page of the application (see Section 1.6).
- Remember when you finish using the application to shut-down all the services by clicking on the `STOP.bat` file.

1.4.4. Final remarks

> ! Antivirus, anti-spyware and firewall programs may block the execution of the Apache webserver (often named `httpd.exe`) and/or of the R environment (both the `R.exe` command and the `R-term.exe` command). If you are using the XAMPP environment, also the `xampp-start.exe` file could be blocked. Other software (e.g. Skype) may use the default port numbers 80 and 443. We set REFRAN-CV to use other ports, but maybe they interfere with other applications.

> # Within the web application package a `xampp-ExampleFiles` folder is given with input files for modules from 1 to 6 already tested along with sample outputs.

> ! If you know you are running the REFRAN-CV web application on an old or slow machine or even if you would encounter error messages like "*Maximum execution time exceeded*", then try opening the `\xampp\htdocs\refran-cv\mockup\include\config.php` file and change the value of the `define('EXECUTION_TIME',0);` variable by increasing it (the number represents seconds given to the script to finish its execution – `0` means 'no PHP limits').

1.5. How to install REFRAN-CV from scratch

In order to install the REFRAN-CV web application on systems with a web server and PHP already installed, or even to install it on systems with different or customized operating systems, follow the procedure described hereinafter.

- Install all the required software as listed in Section 1.1.
- Open the application `xampp\htdocs` folder.
- Copy the folder `refran-cv` and all its content to the public web folder of your system (typically, public web folder name is one of the following: `htdocs`, `www`, `public_html`).
- Using a text editor open the `config.php` file located at: `[public web folder]\refran-cv\mockup\include\config.php`.
- Change the value of the variable `define('RPATH',$RRootPath);` to point to the absolute path of the R executable, paying attention to escape all the special characters (like the double '`\`' in the predefined path). The definition of the R path is done in the three rows before the definition with `define()`.
- Start the web server.
- Open your preferred browser and point it to the application root.

1.6. The REFRAN-CV landing pages

In this Section we provide the screenshot of the three main landing pages of the REFRAN-CV application, i.e. the HOME page and the first screen of both the execution per-module choice and the automatic execution choice.

1.6.1. The REFRAN-CV home page

Figure 1 shows the REFRAN-CV home page. In the header, just below the European Commission, RALCEA and EUROCLIMA logos, it is possible to see the red button to stop the execution of the application. The bottom of the page has the logos of all the other partners. The main content of the page presents a disclaimer being a REFRAN-CV for the time being a preliminary version of the application and not the final one. Just below the disclaimer there are two buttons to choose whether to start the execution per-module (or step-by-step) or the automatic execution of the whole process. At the bottom of the main content part of the page there is a link to open or download this document.

Figure 1: The REFRAN-CV home page

1.6.2. The step-by-step home page

In Figure 2 it is possible to see the 2nd module opening page of the step-by-step execution. It is very close also to all the other pages, so it could be taken as a template. On top of the page, just below the header logos, there are two toolbars. The first one contains the navigation menu, while the second one (with the grey background) shows the title of the module, a set of tools (in the preliminary version they are the two arrows to navigate among modules, a button to start again the selected module and a link to the home page), a permanent link to open or download this document, the STOP button in red and a small, quick contextual guide.

The main part of the page is filled with the interface to the programming logic. In the example the configuration form for the Module 0 (R packages and repository) is shown.

Under the form there is a small red button on the left side to reset the work session (useful only if you want to start again the execution of a step with completely new data discarding the previous results).

Figure 2: Step-by-step execution

1.6.3. The automatic execution home page

Despite of appearances, Figure 3 is slightly different from the Figure 2. Indeed, the only relevant difference between the two pages is the big red warning in the middle of the page. The warning box is due because running the whole process at-once could take two hours or more on slow computers.

Figure 3: The automatic execution home page

Part 2: Design Principles and Structure of the Application

2.1. Requirements Analysis

This work started taking in consideration pre-prepared material. More in details:

- The document: EUROCLIMA project, "Software description: Regional frequency analysis of climate variables", Ispra, Italy, September 2011
- REFRAN-CV mock-up application: a static demo web application showing the desired behaviour of the REFRAN-CV application to be developed
- The R script about the Chilean case example named ComandosR_8regiones.r and the ARF-LM-VERSION2.12.r (dated 25th of November, 2011) by Jorge Nuñez as downloaded from the Cazalac server[19].
- The Chilean data provided along with the R scripts.

During the developing process we did a lot of changes to both the application and the R script, as discussed in the following of this document and in particular:

- The structure of the software is described in the following Sections
- The R script will be analysed and discussed in Part 3: The R Script
- The working data sets are analysed in Section 2.6.

2.2. Design principles

In this Section we summarize main design principles which guided us through the REFRAN-CV application development. For a general description of the application structure refer to Sections 2.4, 2.5 and 2.6.

The REFRAN-CV design has been driven initially from the software specification document and the mock-up, adding few other requirements.

2.2.1. REFRAN-CV execution behaviours

The REFRAN-CV application has been designed to be executed in three different ways:

- The user can choose which module to run (providing the appropriate input information)
- The user can run a subset of subsequent modules (e.g. modules from 2 to 5, or from 3 to 6)
- The user can run the whole REFRAN-CV process at once

While the single-module and the run-at-once execution modes have already been implemented in the preliminary version of the application, the possibility to choose a subset of modules to be executed needs further development (i.e. web forms and PHP functions are ready but there is some logic still missing).

[19] http://www.cazalac.org/documentos/atlas_sequias/chilean_case_example/

2.2.2. Technical design

From a technical point of view, the objectives were portability and modularity. The REFRAN-CV application is a web application and its final release was foreseen to be as a module embedded in a framework (e.g. Aquaknow). But for the sake of testing and to meet the requirements of partners that do not want to disclose on the Internet their data or do not have a network system ready to host the REFRAN-CV application, we develop the first version of REFRAN-CV using an Apache-MySQL-PHP portable environment (Apachefriend's XAMPP) which can run locally. Thus, the REFRAN-CV application can be used on every Windows PC without having the required software (see Section 1.1) previously installed and without the need of accessing Internet.

2.2.2.1. The input/output

The REFRAN-CV does not use a RDBMS (like MySQL) because the application does not need to store and preserve data. It only takes input values and parameters and produce output files and maps. Execution data are temporarily stored using the PHP `$_SESSION` variables. Rather, input parameters are saved using XML files, which replace the `.conf` files foreseen in the mock-up.

Almost all the output values are computed by the underlying R environment, and thus they are exported in CSV/TXT files, JPG images or GeoTIFF maps.

At the end of each module (or at the end of the whole process), a list of all the XML, R and log files is provided to the user. In case of multiple module execution (the user wants to execute more than one module, or a subset of modules, or even the whole process) all the files are offered to the user (i.e. at the end of module 6 you can download also XML files of all the previous modules).

2.2.2.2. Files and folders management

In order to keep under control the size on disk of the application, each time the REFRAN-CV application starts, a new working directory is created and all the output files are stored under it. After a timeout (whose limit can be set by the user) the working directory is deleted. The only kept file is the admin log file, which is saved in the `log` folder outside the working directories space.

Compressed ZIP files created at the end of each execution to save the `output` and `images` folders are built incrementally. This allows users to save files after having finished the execution of the script. They are already ready for export in case of automatic execution of all the modules at once. Users could even save files at the end of each module: main disadvantage is e.g. saving `images_04.zip` have inside also module 3 images if module 3 has been executed before module 4, and the same thing happens for the `output` zip file.

However, some logs cannot be saved (e.g. those of functions to create the initial user environment) because first of all the system has to check whether the `tmp` directory exists and only if it exists it can write logs (it is not possible to write logs in a non-existent position).

2.2.2.3. Naming convention

In REFRAN-CV names are the fulcrum. We decided to use the prefix "`refrancv_`" in functions name and we introduced the "`Refrancv`" package name in order to ease the integration in other software. Furthermore, the REFRAN-CV uses the same names of the R variables as

- XML node names,

- PHP/HTML form fields,
- PHP $_SESSION variables.

This means that users must pay attention if they change the R templates, because they must also rename the related XML element names and PHP variables. Moreover, also module names are keys to activate particular functions or branches of execution. This was due because it is not possible to dynamically create variable names in the R template files.

2.2.2.4. Default values and forms

All the input fields have a default value, in case the user would not specify an input. There are few exceptions for those fields which are strictly mandatory. The function responsible for the form creation is parameterized. Generally speaking its algorithm could be sketched as follows:

1. it takes the module name
2. it look for the XML file of that module
3. it reads all the element names from the XML file
4. for each element name it checks whether the requested value has been already set (by looking through the `$_SESSION` variables or in the file folders); otherwise it adds a new field to the form with its label, the default value and a short description
5. the field name, the label, the default value and the description are taken from the XML file

Once a form is submitted, REFRAN-CV executes another parameterized function running the following workflow:

1. it takes the module name
2. it looks for the XML file and for the R template
3. it reads all the values from the submitted form and does some checks
4. if the value is a file, it uploads the file
5. it saves/updates `$_SESSION` variables
6. it substitutes into the R template all the placeholders with the actual value
7. it saves the data into the XML file

2.2.2.5. Execution time

The R code execution, the files compression to create the ZIP archives, the CSV files display and the customized map creation are time expensive tasks. Even on the test machine the predefined PHP timeout for script execution is not enough in many cases. Thus we had to add a mechanism to let user change the maximum execution time (via the `config.php` file) and to change it only when really needed.

2.2.3. Graphical User Interface and Integration

Integration in Aquaknow portal (which is a Drupal-based system) is foreseen and thus no particular attention has been given to the layout of the framework so far. Instead, modularity has been an

important asset. Also for what concerns the use of advanced and 'pretty' functionalities like those offered by frameworks like Modernizr, Symfony, Script.aculo.us, MooTools, Dojo, ExtJS and so on, we preferred to mostly adopt basic HTML elements with a pinch of CSS. Upon request, we added the full JQueryUI support implementing some layout effects for a better readability (i.e. the Accordion widget, the progress bar, buttons, icons and links).

To help users in finding their working directory, each result page shows the working dir address as link: the user can thus open the working folder directly within the web browser.

We opted for developing REFRAN-CV adopting an asynchronous interaction with the user (near to the classical MVC pattern). This was due primarily to:

- Strict deadline: we started the development on October 2012 having only few months to finish but in the meanwhile the R script and the software requirements changed a lot of times
- File transfer: it cannot be done in a synchronous way but it requires client-server interaction
- Auto-generation of R code: the R code is automatically generated at each execution starting from templates with placeholders
- Execution server-side of R script into the R environment which is outside the web server

Localization of the software is another asset. The R script is written mainly in Spanish while the REFRAN-CV is written in English upon request. Most of example data sets are in Spanish while the XML files have labels written in English. As far as the REFRAN-CV application is used in Latin America countries there are no many difficulties. However, REFRAN-CV has been designed to work in a more general environment and adopted by countries in other continents. Thus means an English version should be created and then used as the main localized distribution. For the time being it is possible to use input CSV files with English headers. English labels are translated by REFRAN-CV in Spanish before executing the code. All the new parts of REFRAN-CV have been developed adopting English as language.

2.2.4. The R back-end

The R script has been divided into 6 parts and the needed code to join together the modules has been added and tested (see Part 3: The R Script for more details).

The short description of each XML node referring to a file the user should select when he or she fills each form has the name of the default file as foreseen in this particular version of the R script (i.e., this version of the R script uses the `BaseDatosEstaciones.csv`). But this is not a constraint: the user can upload files with any name. This should be intended as guidance for the user to understand which kind of file the system is requesting at that particular point.

Similarly, the name of the output CSV files is hard-coded into the R script, so as long as the R script remains the same, the output file name of each module will have a pre-defined value (even if the user can in a second moment change the file name).

2.3. Licence

All the code developed for the REFRAN-CV project is published under EUPL[20] v1.1 (European Union Public Licence) licence (approved by the Open Source Initiative and the Free Software Foundation and compatible with the GPLv2 and other licences). In order to develop and run the application several third parties software components have been used. Before using them we double-checked every licence and they are all released under an open-source licence (GPL, MIT and others). Please check every single licence of the contained products to get an overview of what is allowed and what you are not allowed to do. In the case of commercial use please take a look at the single product licenses.

2.3.1. Disclaimer (points 7 and 8 of the EUPL v1.1)

7. Disclaimer of Warranty

The Work is a work in progress, which is continuously improved by numerous contributors. It is not a finished work and may therefore contain defects or "bugs" inherent to this type of software development. For the above reason, the Work is provided under the Licence on an "as is" basis and without warranties of any kind concerning the Work, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy, non-infringement of intellectual property rights other than copyright as stated in Article 6 of this Licence. This disclaimer of warranty is an essential part of the Licence and a condition for the grant of any rights to the Work.

8. Disclaimer of Liability

Except in the cases of wilful misconduct or damages directly caused to natural persons, the Licensor will in no event be liable for any direct or indirect, material or moral, damages of any kind, arising out of the Licence or of the use of the Work, including without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, loss of data or any commercial damage, even if the Licensor has been advised of the possibility of such damage. However, the Licensor will be liable under statutory product liability laws as far such laws apply to the Work.

2.3.2. Specific Disclaimer for REFRAN-CV

The European Commission shall not be responsible for your use of any information and is not responsible for any errors, omissions in REFRAN-CV. The European Commission does not guarantee the accuracy of the results, nor does it accept responsibility for any use made thereof. If you are relying on any information provided on REFRAN-CV you should seek independent advice.

The Commission further accepts no responsibility or liability for, and makes no warranties, that functions contained at REFRAN-CV will be uninterrupted or error-free or that defects will be corrected.

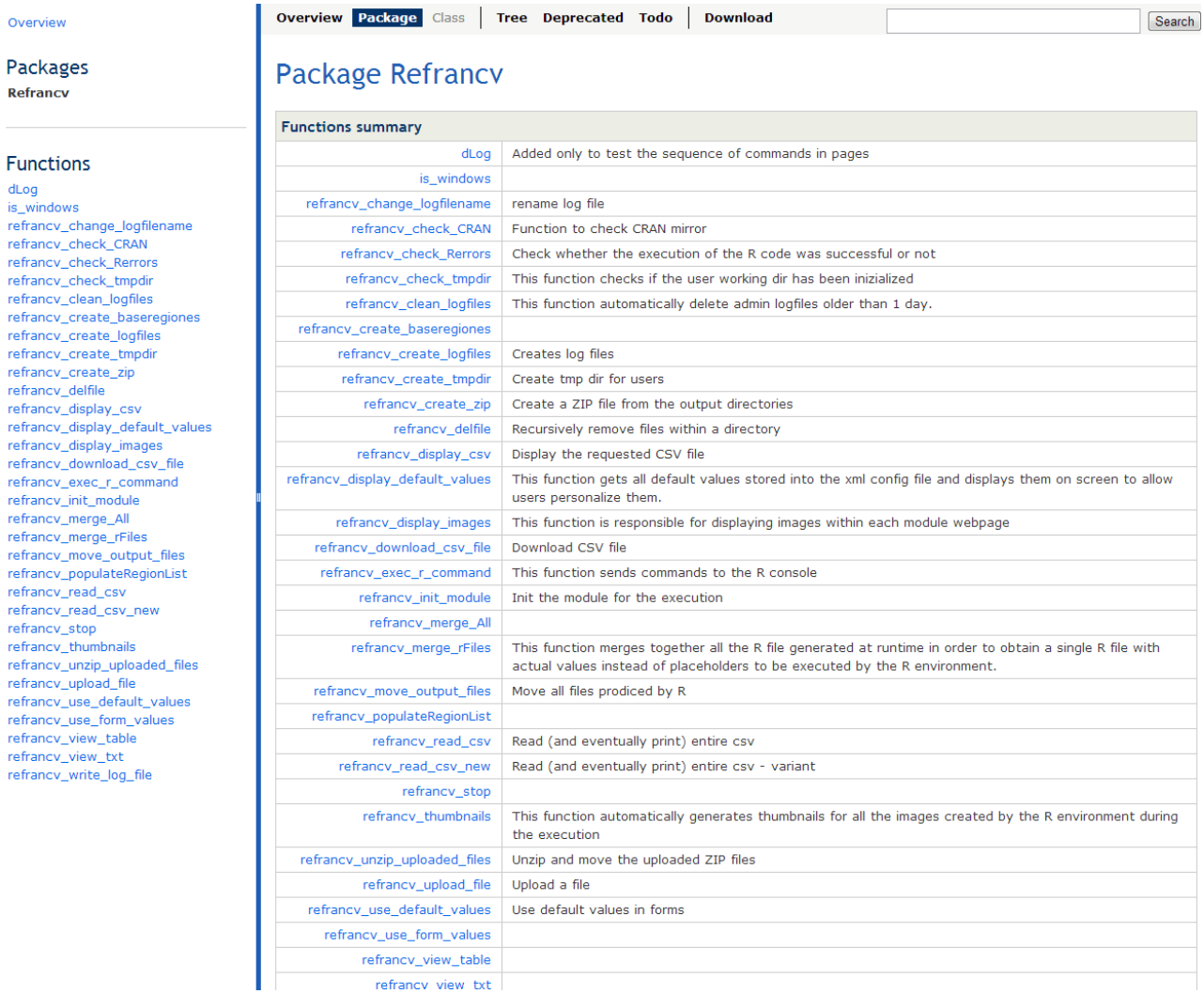
[20] http://joinup.ec.europa.eu/software/page/eupl/licence-eupl

2.4. Notes and details for each module

In this Section an overview of each module is given. Furthermore, PHP functions and configuration file and XML templates are presented.

2.4.1. Functions

The REFRAN-CV application is given with full functions reference automatically generated using the ApiGen system. A copy of the reference can be seen by opening the link "`PHPFuncRef`" in the REFRAN-CV root folder if you install the application using the installer (see Figure 4).



Overview

Packages
Refrancv

Functions
dLog
is_windows
refrancv_change_logfilename
refrancv_check_CRAN
refrancv_check_Rerrors
refrancv_check_tmpdir
refrancv_clean_logfiles
refrancv_create_baseregiones
refrancv_create_logfiles
refrancv_create_tmpdir
refrancv_create_zip
refrancv_delfile
refrancv_display_csv
refrancv_display_default_values
refrancv_display_images
refrancv_download_csv_file
refrancv_exec_r_command
refrancv_init_module
refrancv_merge_All
refrancv_merge_rFiles
refrancv_move_output_files
refrancv_populateRegionList
refrancv_read_csv
refrancv_read_csv_new
refrancv_stop
refrancv_thumbnails
refrancv_unzip_uploaded_files
refrancv_upload_file
refrancv_use_default_values
refrancv_use_form_values
refrancv_view_table
refrancv_view_txt
refrancv_write_log_file

Overview | Package | Class | Tree | Deprecated | Todo | Download | Search

Package Refrancv

Functions summary	
dLog	Added only to test the sequence of commands in pages
is_windows	
refrancv_change_logfilename	rename log file
refrancv_check_CRAN	Function to check CRAN mirror
refrancv_check_Rerrors	Check whether the execution of the R code was successful or not
refrancv_check_tmpdir	This function checks if the user working dir has been inizialized
refrancv_clean_logfiles	This function automatically delete admin logfiles older than 1 day.
refrancv_create_baseregiones	
refrancv_create_logfiles	Creates log files
refrancv_create_tmpdir	Create tmp dir for users
refrancv_create_zip	Create a ZIP file from the output directories
refrancv_delfile	Recursively remove files within a directory
refrancv_display_csv	Display the requested CSV file
refrancv_display_default_values	This function gets all default values stored into the xml config file and displays them on screen to allow users personalize them.
refrancv_display_images	This function is responsible for displaying images within each module webpage
refrancv_download_csv_file	Download CSV file
refrancv_exec_r_command	This function sends commands to the R console
refrancv_init_module	Init the module for the execution
refrancv_merge_All	
refrancv_merge_rFiles	This function merges together all the R file generated at runtime in order to obtain a single R file with actual values instead of placeholders to be executed by the R environment.
refrancv_move_output_files	Move all files prodiced by R
refrancv_populateRegionList	
refrancv_read_csv	Read (and eventually print) entire csv
refrancv_read_csv_new	Read (and eventually print) entire csv - variant
refrancv_stop	
refrancv_thumbnails	This function automatically generates thumbnails for all the images created by the R environment during the execution
refrancv_unzip_uploaded_files	Unzip and move the uploaded ZIP files
refrancv_upload_file	Upload a file
refrancv_use_default_values	Use default values in forms
refrancv_use_form_values	
refrancv_view_table	
refrancv_view_txt	

Figure 4: REFRAN-CV Functions Reference

2.4.2. Module 0

This module gathers all the needed information to check, download and load R packages required by the REFRAN-CV application to run properly. The list of packages is provided as a read-only field just to let users see the list of required packages. However, the user could change the R CRAN mirror to download missing packages. The preferred mirror can be chosen from the list of all the

official available R CRAN mirrors (as published in the mirrors page of the R project website[21]). The default mirror is the Austrian CRAN mirror.

Module 0 is executed each time the application starts or each time the session is reset. It does not produce output files. Module 0 checks if the Internet connection is available before showing the form. If it cannot reach the CRAN repository it will use a local folder to install required packages (provided with the REFRAN-CV package).

2.4.3. Module 1

The module 1 is responsible for the initial input data load and check. The required input data are the country code, the Stations Database CSV file and the Records Database CSV file. While the country code can be selected from a list of all the ISO 3166-1 alpha-3 codes, the two databases must be uploaded by the user. Refer to Section 2.6 for the CSV data format. It is also possible to leave the country code unassigned: in this particular case REFRAN-CV will take the default value from the `module1Config.xml` file. In case a user wants to run REFRAN-CV for an extraterritorial region (e.g. the whole Latin America, North Europe, Chile and Venezuela together, West Africa and so on) the special value to be put into the XML file is "`XXX`" (which is by the way the original default value). There are no differences in executing REFRAN-CV until the third module and only for displaying stations maps. Please refer to Section 2.4.5 for further details.

Among other checks performed by this module, it checks if CSV or CSV2 format is respected, it corrects typos on the columns name and checks if each station declared into the Stations Database is also present into the Records Database and the other way round, removing from the databases all the missing references. Indeed, keeping asymmetrical data could determine statistical errors.

Before checking the file format, this module tries to translate CSV files with English headers into predefined labels. This is very important since column labels are used by the R script as variable names and so as long as the R script remains unchanged this conversion is fundamental. It is achieved by applying the function in the box below to each input file.

```
gsub2 <- function(origin, destination, fields) {
  for(i in 1:length(origin))
    fields = gsub(pattern=origin[i],replacement=destination[i],
                  fields,ignore.case="TRUE")
    return(fields)
}
fromEN<-c('Id_station.*','Countr.*','Station_name.*',
          'Lat.*','Lon.*','altitud.*','Regio.*')
to<-c('id_estacion','PAIS','nombre_estacion',
      'Lat','Long','Altitud','Region')
names(BaseDatosEstaciones) = gsub2(fromEN,to,names(BaseDatosEstaciones))
```

[21] http://cran.r-project.org/mirrors.html

The CSV format check is performed by executing the `f.checkFiles` function as in the `module1.template` file. Simple substitutions are performed on column names if they are not well-formatted, by executing simple transformations like the one in the box below.

```
names(BaseDatosRegistros) =
  gsub(pattern="anio$",replacement="Anio",
names(BaseDatosRegistros),ignore.case=TRUE)
```

Missing references are found by executing a query on both files, like the example below that finds all the stations declared into the Stations Database but not into the Records Database and remove them.

```
estNOTINreg<-sqldf("select d1.id_estacion from BaseDatosEstaciones as d1
                   where d1.id_estacion NOT IN
                   (select d2.id_estacion from BaseDatosRegistros as d2)
                   group by d1.id_estacion")
if (length(regNOTINest[[1]])>0)
    BaseDatosRegistros<-sqldf(c("delete from BaseDatosRegistros
                            where id_estacion in
                            (select id_estacion from regNOTINest)",
                            "select * from BaseDatosRegistros"))
```

The output files produced by the module 1 are new versions of Stations Database and Records Database files and a plain text file with the chosen country code. Both Stations Database and Records Database files have a timestamp into their name in order for the user to easily find the right version. Furthermore, it also produces one new CSV file called `PP.csv` (the name was already hard-coded inside the R script), needed by module 2. Also, it creates and saves other two files: `benchmark.csv` with process and time statistics and `r_code.txt` with the actual code executed by the R environment. The latter could be copied and pasted into an R environment (like 'R Studio') and executed from there without any further modification (except the input files full paths).

2.4.4. Module 2

Module 2 is responsible for the exploratory data analysis. It requires as input the plain text file with the country code saved at the end of module 1, the `PP.csv` CSV file and the Stations Database checked and cleaned after the module 1 execution (e.g., `BaseDatosEstaciones_130418-083743UTC_checked.csv`). Homogeneous regions are identified from within the Stations Database. Sometimes they refer to administrative boundaries, while in other cases they have been identified on a climatic base. Thus, they are created by executing a query on the `Region` field of the stations database. Advanced users could change the R script in order to group stations in a different way. In the newest version of the R script made by J. Nuñez, regions are created from scratch without the need for a base map. Unfortunately, this version was sent to us the 30th of May 2013, too late to be implemented in this version of REFRAN-CV.

During the execution of this module several files are created, the most important being:

- the complete database (a.k.a. `BaseCompleta`)
- the intermediate database (a.k.a. `BaseDatosIntermedia`)
- the total database with L-moments (a.k.a. `BaseDatosTotal_Lmomentos_Regional`)
- all the regions' CSV files
- a set of results CSV files (e.g. `ResultadosARFD`, `ResultadosARFH`, `ResultadosARFZ`, ...)
- the summary statistics file

Furthermore, module 2 computes several graphics for the entire dataset as well as the L-moments. An example output can be seen for Venezuela's data in Figure 5 to Figure 10.

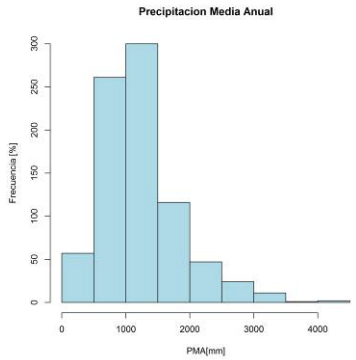


Figure 5: Mean Annual Precipitation (MAP) for Venezuela

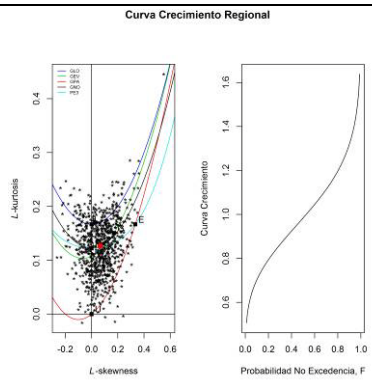


Figure 6: Growth curve

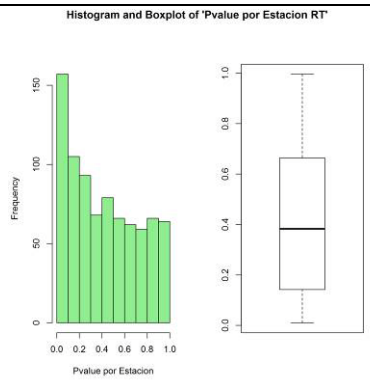


Figure 7: Stations' P-value

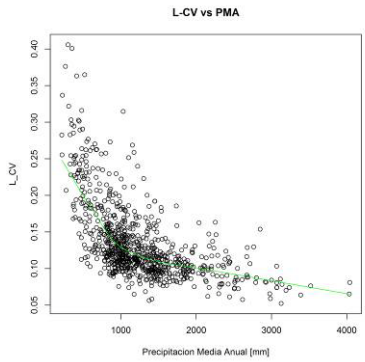


Figure 8: Scale/Dispersion (L-CV) vs. MAP

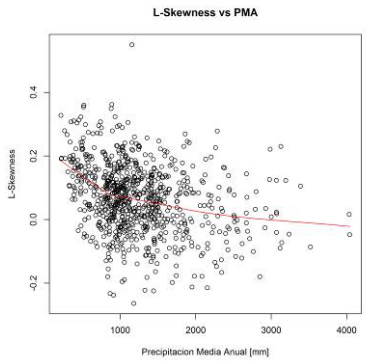


Figure 9: Skeweness vs. MAP

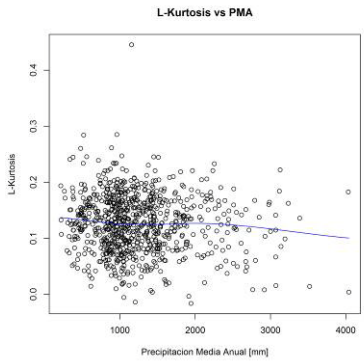


Figure 10: Kurtosis vs. MAP

2.4.5. Module 3

Module 3 performs the Regional Frequency Analysis (RFA) using homogeneous regions, by selecting the probability distribution function for each homogeneous group.

The requested input files of this module are:

- the Country code stored in a plain text file

- a CSV file created in second module with the regions' name list (e.g. `NombreRegiones.csv`)
- the Consolidated DB CSV file without NA values (e.g. `BaseConsolidadaXXX_sin_NA.csv`)
- the Complete DB CSV file (e.g. `BaseCompletaXXX.csv`)
- the regions dataset: a ZIP file with all the CSV files of the Regions (if needed)
- the Regional L-moments Dataset (e.g. `BaseDatosTotalXXX_LmomentosRegional.csv`) created in the 2nd module

Often the '`XXX`' is substituted by the ISO country code. The region dataset is required only if the execution starts from this module. Otherwise the REFRAN-CV application uses the CSV files set created in the previous module. The same happens for the Regional L-moments Dataset, which is used in this module only to create the stations comparison plot and the L-moments maps.

Because of the size of the Complete DB CSV file, we had to increase the value of the maximum post / upload filesize in `php.ini` file setting it to 30 MB.

The core part of this module does almost the same analysis of the previous module but in this case all the performed statistical analyses are done on each region file and not on the whole country. Several files are created during the execution of this module, the most important being:

- the L-moments total databases (a.k.a. `Total_Lmomentos.csv`, `BaseDatosTotalXXX_Lmomentos.csv`, `BaseDatosTotal_Lmomentos.csv`)
- the summary statistics files (a.k.a. `ResultadosSummaryStatistics.csv`)
- all the regions' CSV files
- a set of results CSV files (e.g. `ResultadosARFD`, `ResultadosARFH`, `ResultadosARFZ`, `ResultadosRMAP`, `ResultadosRlmoments`...)
- the regional quantile file (a.k.a. `ResultadosRegionalQuantiles.csv`)

Furthermore, module 3 computes several graphics for the entire dataset as well as the L-moments for each region. An example output can be seen for Venezuela's data in Figure 11 to Figure 16.

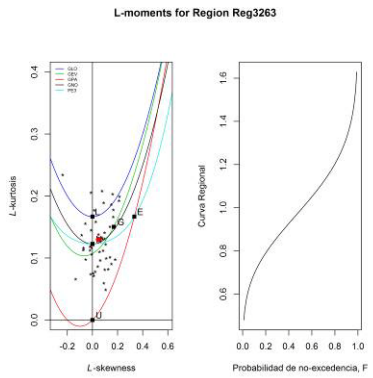


Figure 11: L-moments for a region

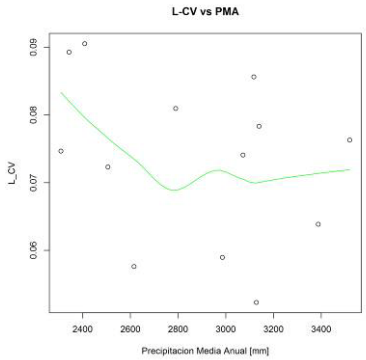


Figure 12: L-CV vs Mean Annual Precipitation (MAP)

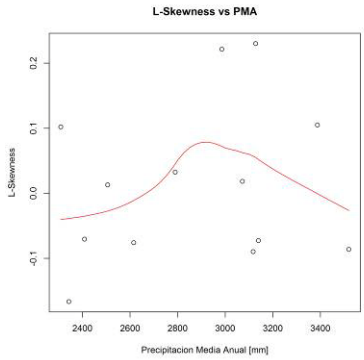


Figure 13: Skeweness vs MAP

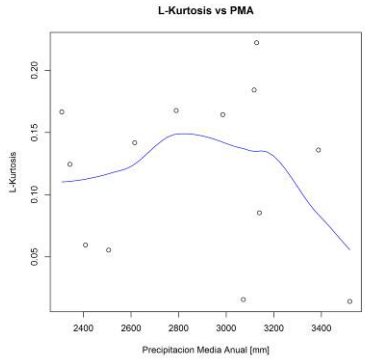


Figure 14: Kurtosis vs MAP

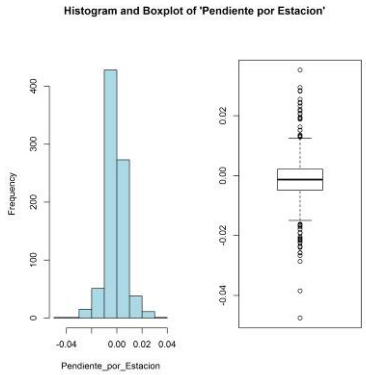


Figure 15: Slope/gradient for stations

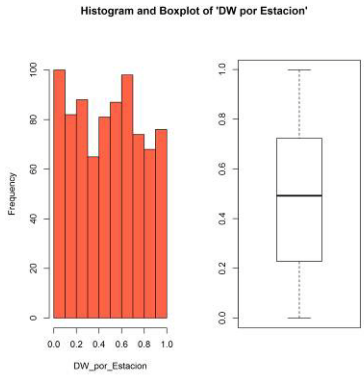


Figure 16: Results of Durbin Watson test

Once the regional analyses are finished, users can select one or more regions and open a map showing where the stations are placed (getting this information from the Latitude / Longitude columns of each station) and the four L-moments computed for each station, as in Figure 17.

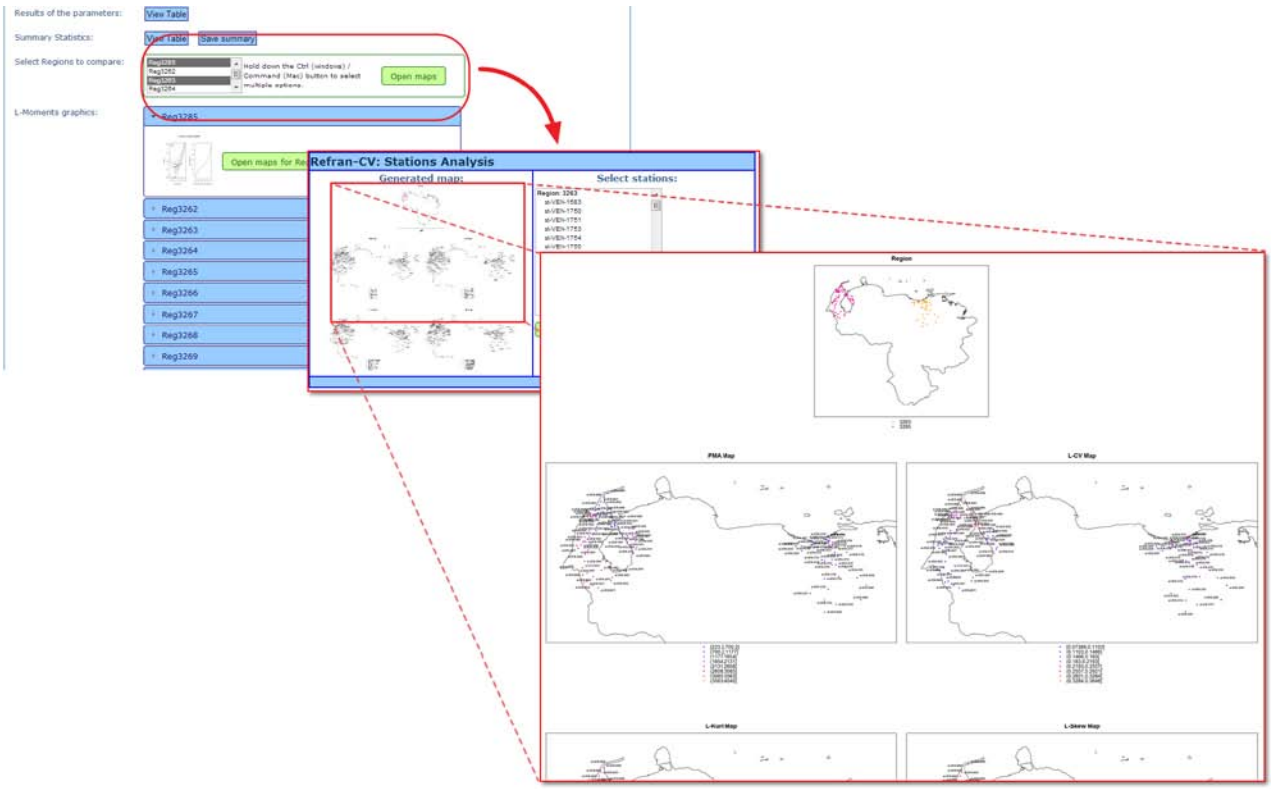


Figure 17: Stations / L-moments map

From the Stations / L-moments map page the user can select up to four stations (Figure 18), belonging to the same or to different regions (among those selected) in order to create four different plots comparing their analytical values (all the values available in their registers: the mean monthly precipitation), as in figures from Figure 19 to Figure 22. They are:

- a barplot with the Average Monthly Precipitation (Figure 19)
- a multiline plot with rainfall values for each station (Figure 20) – in this plot each row has its own Y scale, useful to see peaks and no-data intervals
- a multiline plot with rainfall values for each station (Figure 21) – in this plot the Y scale is the same, useful to compare values of the stations
- an overlaid plot (Figure 22) with all the values together, useful to easily see main differences among stations.

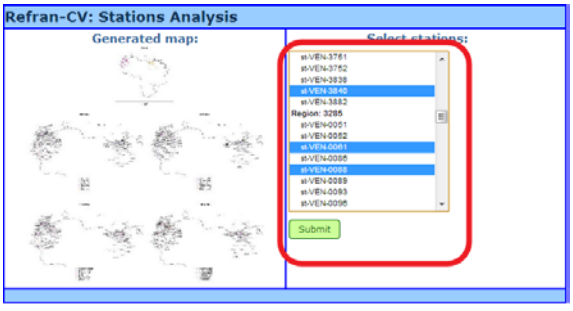


Figure 18: Selection of stations

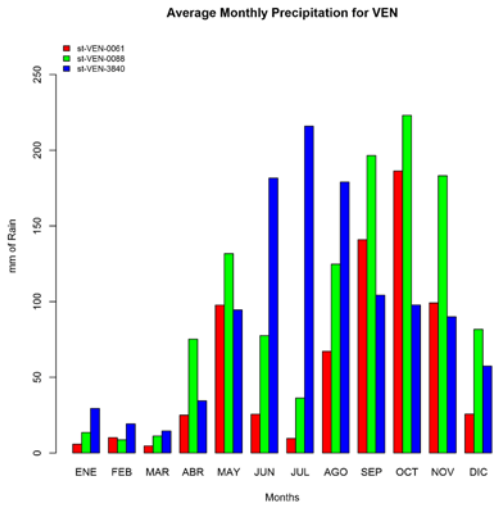


Figure 19: Average Monthly Precipitations for VEN

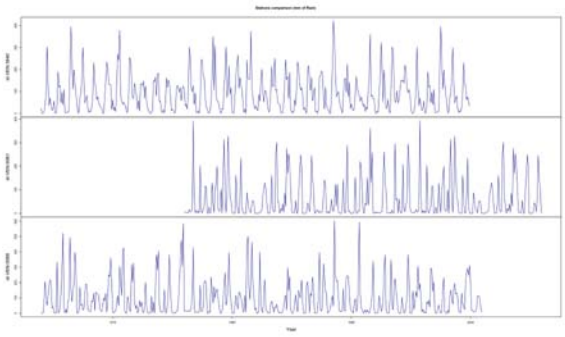


Figure 20: Comparison of stations (different Y scales)

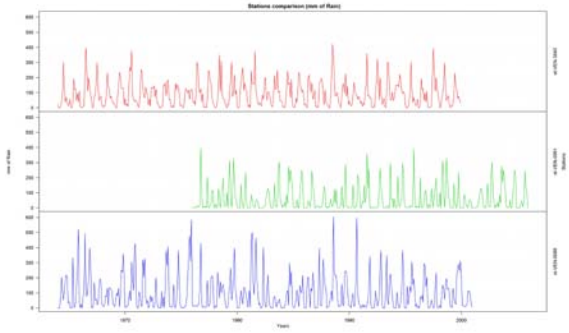


Figure 21: Comparison of stations (same Y scales)

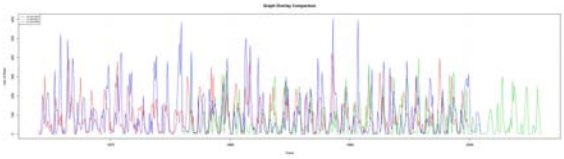


Figure 22: Overlaid plot for VEN

As introduced in Section 2.4.3, it is possible to generate the stations maps also in case of extraterritorial data. As it is possible to see in Figure 23 (showing fake data), if the special value "XXX" is used as country code, REFRAN-CV will display stations on the entire world map.

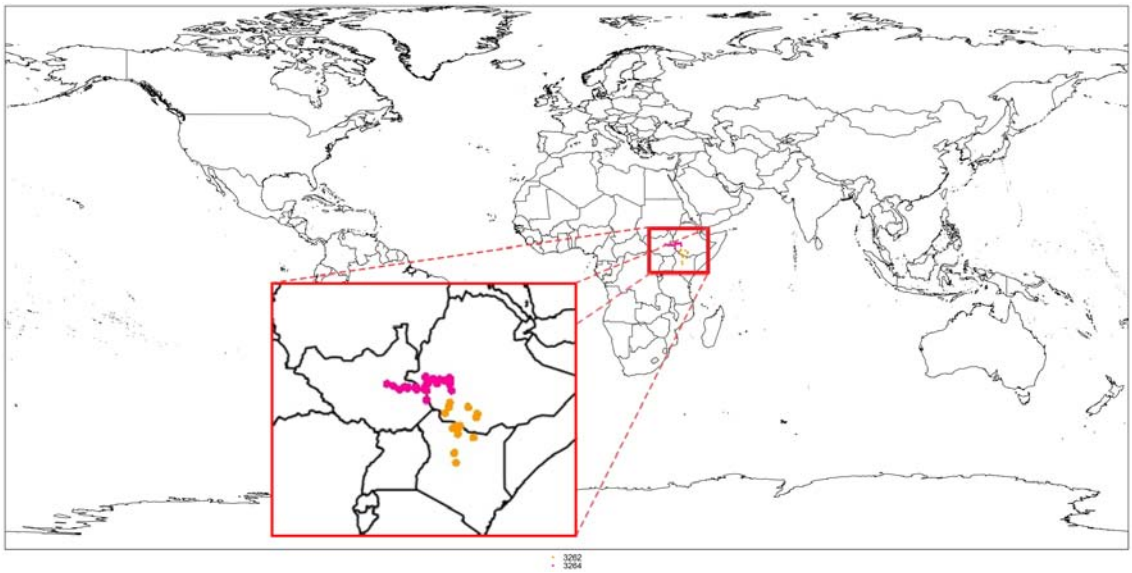


Figure 23: Cross-countries stations

To help users visualize the selected regions, the picture generated by REFRAN-CV in this case is four-times bigger than in the other case. Then, for L-moments maps, it zooms to the smallest region computed to show the stations, as it is possible to see in Figure 24.

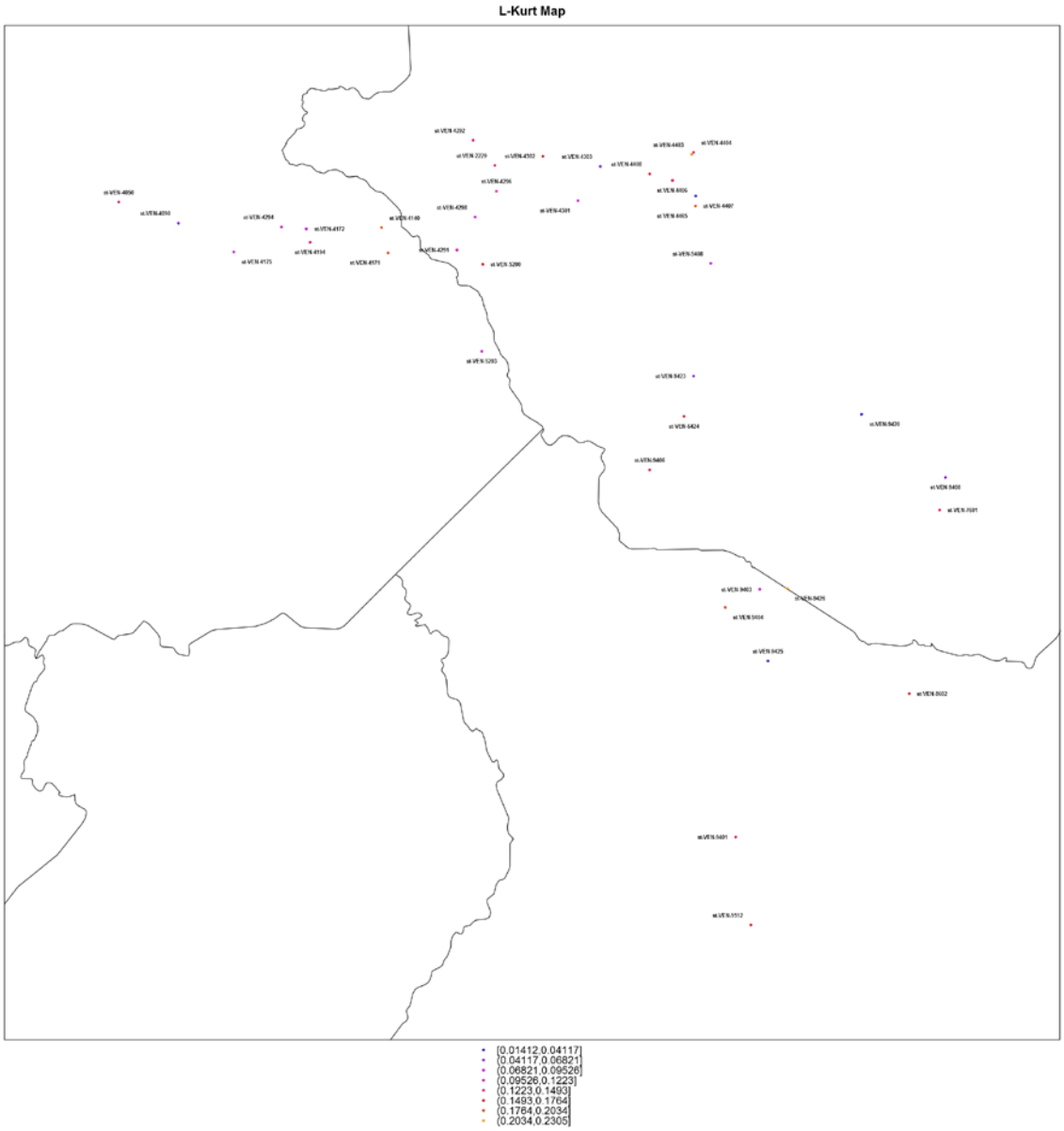


Figure 24: Cross-countries stations L-kurtosis map

2.4.6. Module 4

Module 4 performs the calculation of interpolation parameters. In Module 3, the L-moments are defined for each station. In order to create spatially-explicit maps, this information needs to be interpolated to areas where no stations are available in the region. The method adopted in this module to interpolate data is the `DEoptim` function which performs global optimization by differential evolution. Originally there were other two methods, the nonlinear (weighted) least-squares (`nls`) and the nonlinear minimization (`nlm`, minimization of the function `f` using a Newton-type algorithm), but with the last version of the R script they do not work anymore as expected and thus you could find their code into the script template even if they have been

commented. The optimization requires as input files two of the files generated during the Module 3 execution: the L-moments results file (e.g. `ResultadosRlmoments.csv`) and the RMAP results file (e.g. `ResultadosRMAP.csv`).

The output of this module is made of a set of three files with the parameters used to compute the optimization for the 2nd, 3rd and 4th L-moment, respectively Scale/Dispersion, Skeweness and Kurtosis (e.g. `paLCV.csv`, `paLSk.csv` and `paLKurt.csv`). Below an excerpt from the latter:

```
"par1","par2","par3"
0.299999999999774,-0.00896573491326376,0.158749277162349
```

Furthermore, also three graphics are produced as graphic result, as presented from Figure 25 to Figure 27.

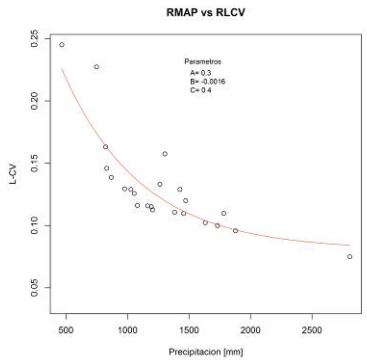


Figure 25: RMAP vs RL-CV

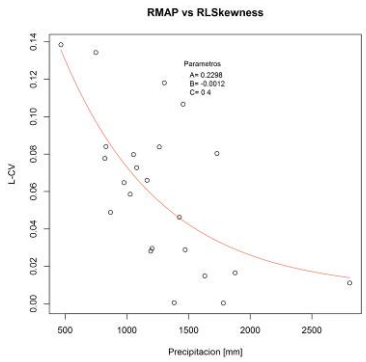


Figure 26: RMAP vs RL-Skeweness

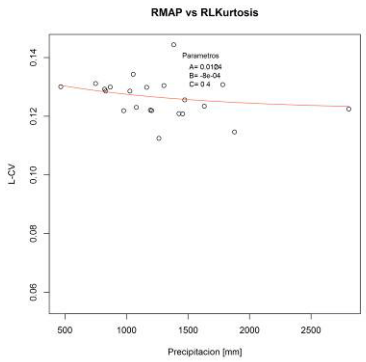


Figure 27: RMAP vs RL-Kurtosis

2.4.7. Module 5

Module 5 generates L-moments maps using interpolation parameters calculated in Module 4 and an annual precipitation map in GeoTIFF format. Furthermore in the last version of the R script, Module 5 is also responsible for calculating the Inverse Distance Weighted (`idw`): the function `idw` performs just as `krige`[22] without a model being passed, but allows direct specification of the inverse distance weighting power. More in details, Inverse Distance Weighting (IDW) is a type of deterministic method for multivariate interpolation with a known scattered set of points. The assigned values to unknown points are calculated with a weighted average of the values available at the known points.

This is actually the first REFRAN-CV module which works on a map. Hence, the inputs for this module are the following:

- the base map with mean annual precipitations (MAP) data, in GeoTIFF format
- the map projection, useful if you want to change the default map projection (default value is the EPSG:4326[23] a.k.a. WGS84

[22] Kriging is a geostatistical estimator that infers the value of a random field at an unobserved location (e.g. elevation as a function of geographic coordinates) from samples.

[23] http://spatialreference.org/ref/epsg/4326/

- the value for '`NoData`' regions of the map (i.e. all the pixels not to be considered in the analysis have a value very different from admissible values, like `-999` or `-3.14e38`)
- the consolidated database without NA values (this file is required for `idw` computation)
- the interpolation parameters files:
    - parameter file for Scale/Dispersion (2nd L-moment)
    - parameter file for Skeweness (3rd L-moment)
    - parameter file for Kurtosis (4th L-moment)

The output files of Module 5 are five new maps, one for each L-moment, the IDW map and a multi-band GeoTIFF map which merges together the base map with all the other 4 maps. The latter is the input for the final module. The L-moments and IDW maps for Venezuela can be seen from Figure 28 to Figure 31.

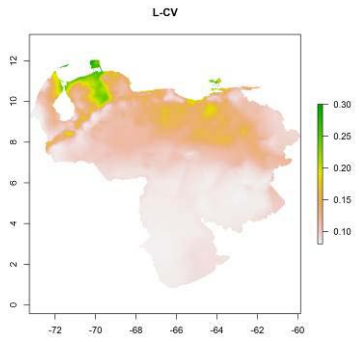


Figure 28: L-CV map for Venezuela

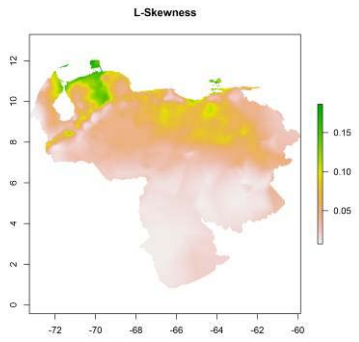


Figure 29: Skeweness map for Venezuela

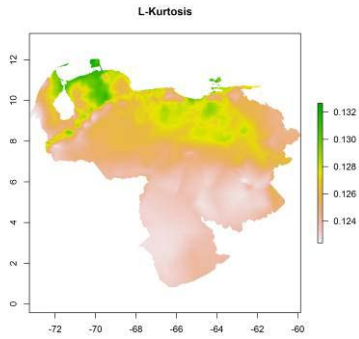


Figure 30: Kurtosis map for Venezuela

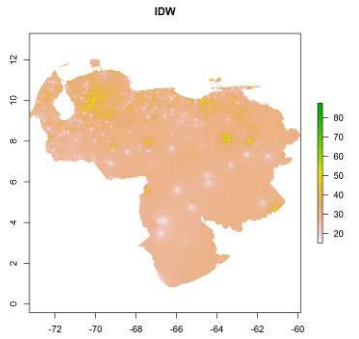


Figure 31: Inverse Distance Weighted for Venezuela

The output maps in this module can be graphically modified by changing their title, the colours scale, the axis labels, the resolution and dpi value and by adding or hiding an over-layered grid. This feature was a specific request in the software description document. Figure 32 shows an example map with some of the listed options changed.

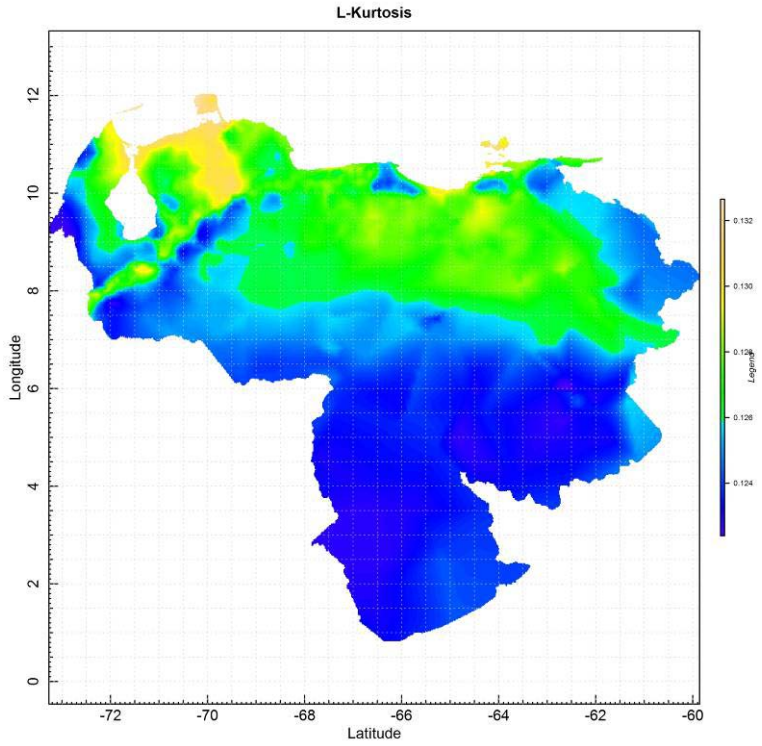


Figure 32: A customized L-Kurtosis map for Venezuela

2.4.8. Module 6

The Module 6 generates the final product map of REFRAN-CV. The aim of this module is to produce four maps showing an estimation of the return periods of N years, with N chosen in a set of values offered to the user: 5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100. The user can choose from one to 4 values. To calculate return period `T=1/P` or `T=1/(1-Pn)`, where `Pn` is the probability of non-exceedance, the calculations in this module do the inverse, i.e. estimate the probability of non-exceedance as a function of the Return Period. As an example:

```
Pn[i]=(N-1)/N # The probability associated with return period of N years
```

The multi-band map produced as output of the fifth module is casted into a `regdata` data structure. An object of class `regdata` stores summary statistics of the data for the sites in a region. It is a data frame with each row containing data for one site. The columns should contain the site name, record length and L-moments and L-moment ratios, in the order: mean, L-CV, L-skewness, L-kurtosis, etc.

The core part of this module is therefore the calculation of `regtst`, which computes discordancy, heterogeneity and goodness-of-fit measures for regional frequency analysis. The `regtst` is executed six times over the multi-band map of Module 5, interpreted as `regdata` data structure, changing the number of simulations used in the calculation of the heterogeneity and goodnessof-fit measures. The `regtst` is computed for each map pixel.

The inputs required to run this part of the module are the multi-band map created at the end of ‘Module 5’, the ‘`NoData`’ value as in ‘Module 5’ and the preferred projection.

Figure 33 to Figure 36 show the result maps for Venezuela.

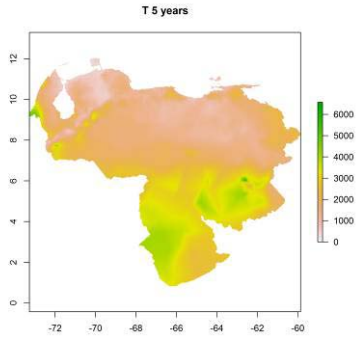


Figure 33: Estimate return period of 5 years for Venezuela

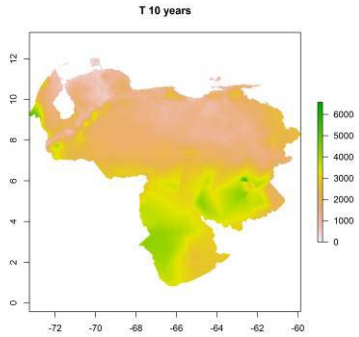


Figure 34: Estimate return period of 10 years for Venezuela

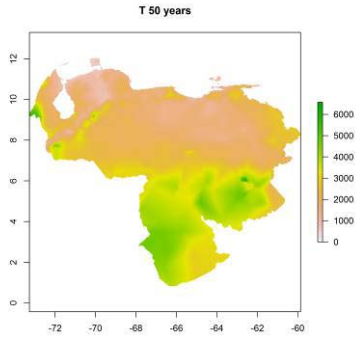


Figure 35: Estimate return period of 50 years for Venezuela

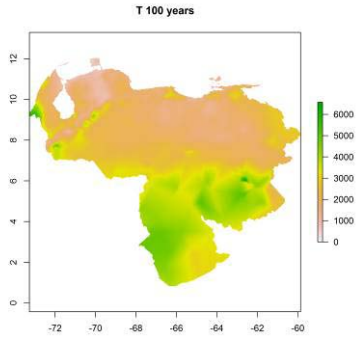


Figure 36: Estimate return period of 100 years for Venezuela

Furthermore, users can also specify values for `Quantile` and `mmOfRain` in order to compute the Return Period Maps for the specified values. Example output maps for a quantile value of 0.8 and for 3000 mm of rain can be seen in Figure 37 and Figure 38.

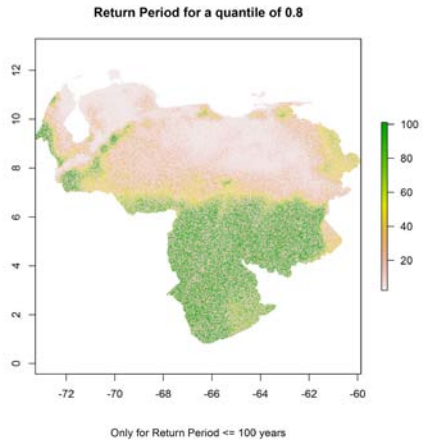


Figure 37: Return Period for a quantile of 0.8 in VEN

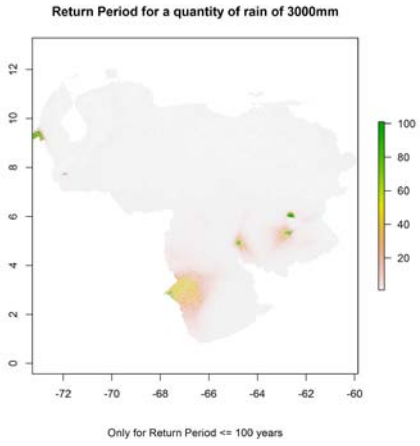


Figure 38: Return Period for a quantity of rain of 3000 mm in VEN

The return period map based on quantity of rain and quantile share the same model and the same parameters of the other maps but substitute the original mean annual precipitation value with an hypothetic value expressed respectively as a new `mmOfRain` value or a percentage of the original value. While the latter is computed for every pixel starting from pixel's original value, the former fix the same value for every pixel of the map. We decided not to represent return periods higher than 100 years because the reliability of the model would be very poor.

This module is very time-consuming and could take more than 2 or 3 hours to finish on a slow PC (see Section 2.6.5 for more accurate estimations). It could be possible in a future release to slice the merged map in order to compute the final calculation on smaller sets of data and then merge the results together again. This way there could be possible to execute the REFRAN-CV also on computer with fewer hardware resources.

2.4.9. ModuleAll

ModuleAll is the module in charge of collecting all the required information needed to run the whole process from the beginning. It asks users to provide all the input at the very beginning and then it merges together all the R template on-the-fly. Once checked the input files and the code, it starts the execution of the whole script until the generation of Return Period Maps. Figure 39 shows the start page for the ModuleAll execution.

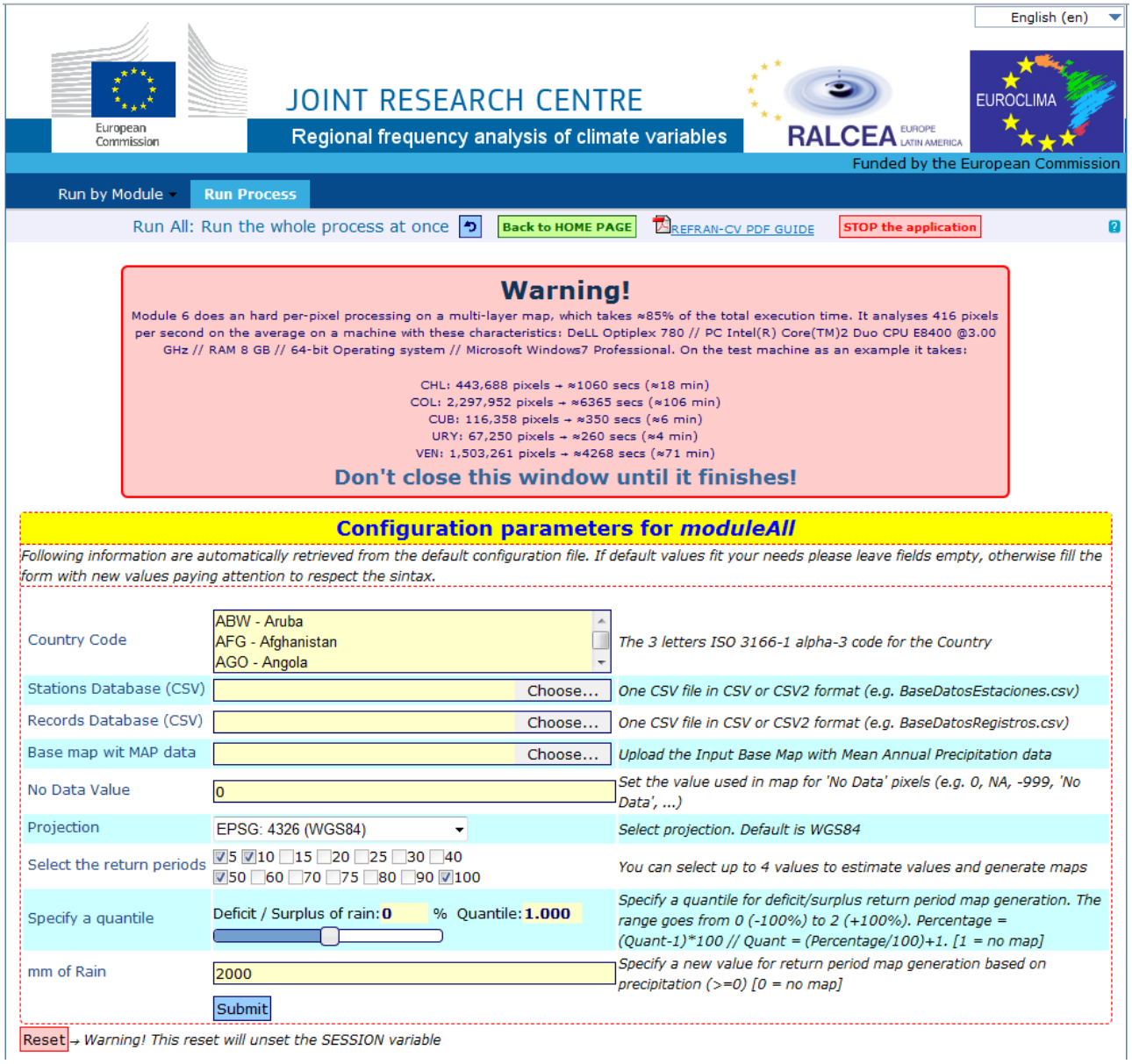


Figure 39: Top of the ModuleAll starting page

2.4.10. Others

In this Section we show few important notes about the general configuration file and the structure of XML templates.

2.4.10.1. Config.php

The `config.php` file contains main constants used by the REFRAN-CV application. In particular, it holds the absolute path of the local R packages folder and of the R executable, the maximum PHP execution time, the default language (even if in this preliminary version localization it is not used), the Internet connection status, the default R CRAN mirror (as a back-up choice if the selected mirror would not work properly) and a value expressed in hours referring to the time-to-live of the user working folders. Hereinafter it is possible to read an excerpt of the file.

```
define('RPKGDIR',$RPkgDir);
define('RPATH',$RRootPath);
```

```
define('STOPPATH',$RefranSTOP);
define('EXECUTION_TIME',0);
define('DEFAULT_LANGUAGE','en');
define('CONNECTED', TRUE);
define('DEFAULT_CRAN','http://cran.us.r-project.org');
define('TMP_TIMEOUT',3); // Timeout for users tmp dir (hours)
```

2.4.10.2. XML templates

In order to collect all the user input parameters and pass them to the application, we chose to use XML files, instead of plain .conf files as foreseen in the initial mock-up. Although the XML structure could seem harder to understand from many people, XML files offer many more possibilities than plain files. For the time being, we would only need few parameters to run the application. Nonetheless, we created XML files with a sort of 'hook' to add further attributes in future releases. Moreover, working with XML files makes much easier to localize files in different languages by adopting different namespaces. The REFRAN-CV application uses XML files to dynamically create input forms. The XML files indeed provide not only the fields name but also the label of each field, a short description and the default value. You can see an example structure in the example below. The general structure and constraints of the XML files is as follows:

- The filename must be of the form: `*modulenumber*Config.xml` with `modulenumber` substituted with the name of the module (e.g. `module0Config.xml`, `moduleAllConfig.xml`).
- The root element must be the `modulenumber`.
- Then `author` and `documentation` sections follow.
- Under the `parameters` node it is possible to list all the required parameters (e.g. all the allowed projections).
- Then the main part under the `rcode` node. Each child node shares the same structure:
    - the `name` of the element, which must be the same as the variable name in the R template (this is a necessary hook to dynamically build the final version of the R code to be executed);
    - the `label` attribute (mandatory, this is the label printed on screen near each field);
    - a `short` description (optional);
    - the default value of the element, which corresponds to the default value we want to set for a particular variable of the R script.

```
<?xml version="1.0" encoding="UTF-8"?>
<module*number*>
 <author/>
 <documentation/>
  <parameters>
```

```
    ... list of parameters ...
  </parameters>
  <rcode>
   <projection label="Map projection"
               short="Select projection"
               other_attributes="..."/>
   <MergedMaps label="Multi-band GeoTIFF map"
               short="This is the map generated in module 5 (e.g.
                      MergedMaps.tif)">
      output/examplemap.tif
   </MergedMaps>
  </rcode>
</modulenumber>
```

The short description of each node referring to a file the user should select has the name of the default file as it is created with this version of the R script. The name of the output CSV files is hard-coded into the R script, so until the R code remains the same, the output file name of each module has a pre-defined value (even if the user can in a second moment change the file name).

2.5. Folder Structure

The list structure presented in this section reflects the folder structure of the application. For the sake of simplicity and better readability only the relevant folders are mentioned hereinafter.

- **`xampp`**: the application root folder
    - **`apigen`**: the installation folder for the ApiGen tool
    - **`R-3.0.1`**: the root folder for the R portable environment
    - **`REFRANCV-Rpackages`**: the local R packages repository
    - **`htdocs`**: the public web folder
        - **`refran-cv`**: the application root folder
            - **`mockup`**: the application mockup root folder (at this stage of development, this is actually the root folder of the application). It contains the `index.php` file.
                - **`conf`**: in this folder all the configuration XML files are stored
                - **`css`**: here it is possible to see all the Cascading Style Sheets files used within the application. For backward-compatibility with the initial mockup all the CSS files have been kept.
                    - **`images`**: the graphical files referenced into the CSS files
                - **`docs`**: all versions of this file and supporting materials

- **`html`**: here all the PHP files of the pages are stored. This folder is called "html" even if it contains PHP files: this is due to backward-compatibility with the initial version.
- **`images`**: many various graphical files
- **`include`**: this folder contains the main application files (`functions.php` and `config.php`) and other support-files.
  - **`R`**: all the template files for R scripts (one for each module) are stored in this folder
  - **`Lang`**: all files needed for the localization of the application
  - **`SHP`**: the Shape files need in module 3 to create the map of stations
- **`js`**: the Javascript files repository. This folder also has few sub-folders dedicated to particular extensions.
- **`log`**: all the application log files will be stored in this folder.
- **`PHPDoc`**: this folder is used by the ApiGen tool to store the automatically generated PHP functions documentation.
- **`tmp`**: the root directory where all the users' working directories will be created and stored. This is called "`tmp`" because after each run, if the timeout is reached, the working directory is trashed.

- **`xampp-ExampleFiles`**: an helping folder for the portable version of REFRAN-CV with sample data or for the full installation package (see Section 1.3).

2.6. Description of input data

2.6.1. Format of input data files

Despite of all the different file formats we had to work with, the R scripts we developed now accept files only in standard CSV or CSV2 formats. "CSV" is not a single, well-defined format, but we refer to the format as described in RFC 4180[24] where:

- each record is located on a separate line, delimited by a line break (CRLF);
- there maybe an optional header line appearing as the first line of the file with the same format as normal record lines;
- within the header and each record, there may be one or more fields, separated by commas;

[24] http://tools.ietf.org/html/rfc4180

- each line should contain the same number of fields throughout the file;
- spaces are considered part of a field and should not be ignored;
- the last field in the record must not be followed by a comma;
- character separator for decimals is a period/full stop: "`.`"

The CSV2 format definition is taken from the R `read.csv2` and `write.csv2` commands. While CSV format is mainly used in USA/UK documents, Germany, Netherlands and many other European countries use a comma as decimal separator. Thus, the CSV2 format reads and saves files adopting the comma as decimal separator and the semi-colon as field separator[25].

Using REFRAN-CV you can use both formats but it is important not to mix the two formats, e.g. using the semi-colon as field separator and comma as decimal separator. However, a simple format check has been introduced in module 1.

2.6.2. Format of data fields

CSV files can optionally have a header. However, the REFRAN-CV input CSV files MUST have a well-formed header because some of the fields are read by the R code and used to match data or read columns. This means that the headers should be CSV/CSV2 compliant AND have as field names pre-defined values (many of those at the moment are in Spanish because the original script has been produces in Latin America even if it is also possible to use CSV files with English labels). Here follows the list of mandatory fields for each of the two input files required by the first module of REFRAN-CV (consider them case sensitive, because R is a case sensitive environment).

- Stations Database (a.k.a. BaseDatosEstaciones):
  - `id_estacion` – Station IDs
  - `Lat` – Latitude
  - `Long` – Longitude
  - `nombre_estacion` – Station's name
  - `Altitud` – Altitude
  - `Region` – Region (its values could be numbers or strings)
- Registers Database (a.k.a. BaseDatosRegistro):
  - `id_estacion` – Station Ids
  - `Anio` – Year
  - `ENE`,`FEB`,`MAR`,`ABR`,`MAY`,`JUN`,`JUL`,`AGO`,`SEP`,`OCT`,`NOV`,`DIC` – List of months (three letters abbreviation from January to December)

[25] Refer to http://en.wikipedia.org/wiki/Decimal_mark#Influence_of_calculators_and_computers for a full list of countries using different formats.

In the box below it is possible to see an excerpt of the first few rows of both the main input files taken from Venezuelan dataset in CSV format (the bold typeface has been added to highlight the header rows), respectively they are the Stations Database and the Records Database.

```
id_estacion,pais,nombre_estacion,Lat,Long,Region
st-VEN-0051,VEN,GUARERO,11.365,-72.065,3285
st-VEN-0052,VEN,PARAGUAIPOA_GUANA,11.353,-71.967,3285
st-VEN-0061,VEN,RANCHO_GRANDE,11.244,-72.123,3285
...

id_estacion,Anio,ENE,FEB,MAR,ABR,MAY,JUN,JUL,AGO,SEP,OCT,NOV,DIC
st-VEN-6424,1958,NA,NA,NA,101,205,528,395,251,144,103,62,33
st-VEN-6424,1959,0,19,87,61,191,380,475,228,267,84,196,39
...
st-VEN-6424,1976,18.3,7.9,46.9,NA,233,669.4,292,592.8,96.9,184.5,62.9,22.9
...
```

Furthermore, in the box below it is possible to see an example of the English version of the accepted headers for the minimum set of mandatory fields. Also in this case the labels format is a constraint.

```
id_station,Country,Station_name,Lat,Long,Altitude,Region

id_station,Year,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

2.6.3. The Input Map format

Apart from the two main CSV files, REFRAN-CV application needs an input base map. This map must be a raster file in TIFF format holding the mean annual precipitation values on it. It is very important to set all the values outside the country boundaries to `0` or `NoData` to allow the R script to process them properly. Values such as `-3.4e+38` could raise errors when the script tries to compute means or divisions. However, in the last version of REFRAN-CV the user can specify the value to be used as '`No Data`', even if we strongly encourage users always to use values like `0` or `-999` (if `0` can be considered an admissible value for rain precipitation).

The projection of the map is not so much relevant because it is possible to re-project the map using the REFRAN-CV application. However, the suggested projection is the `EPSG:4326 (WGS84)`[26].

2.6.4. Example Data Sets

In this paragraph we want to briefly present the datasets we used to develop, test and improve the REFRAN-CV application.

[26] http://spatialreference.org/ref/epsg/4326/

- Chile: they were a set of data built from scratch by-hand. They were very realistic than those calculated from the R script but the format was completely different from the template foreseen for CSV files, and also for stations IDs, rows/columns, and so on. Thus in practice they could be used only with the `ComandosR_8regiones.r` script which came with them.
- Latin America: These data were downloaded from the Cazalac Chilean example data repository. Their format was better than the Chilean data but these files refer to many data across all the Latin America. Many of them have been invented and thus they are not realistic (e.g. they report a mean annual precipitation of more than 15000 mm near Santiago del Chile or more than 2000 mm in a desert area or even an Altitude value for 9999 mt at 'Los Cerrillos' Airport in Santiago). Moreover, only few of them refer to Chile, too little to be meaningful.
- Colombia: This dataset is very accurate and tested. There were few problems on the file format and because of typos and personalization made by the Colombian partner. We could start working on these data only from January 2013.
- Venezuela: The main problem in using this dataset with REFRAN-CV application was the input files format. Indeed, they were prepared and tested for a different version of the script. Furthermore, the base map was missing because of a different creation process. The map provided with the script was poor in quality. We could start working on these data only from February 2013.
- Cuba: We had also the opportunity in March 2013 to test REFRAN-CV with Cuban dataset. The dataset is very accurate. We encountered only few errors in coordinates' definition and on the values scale of the base map.
- Uruguay: We had also the opportunity in March 2013 to test REFRAN-CV with Uruguayan dataset. The dataset is very accurate even if it has only 53 stations. We encountered only few errors in coordinates' definition and on the values scale of the base map. Furthermore, Uruguayan data are all grouped in only one region. The base map also in this case was very poor in quality (67250 pixels).

2.6.5. Execution Statistics

Using as test machine the one described in Section 1.1 we depict in this Section some execution statistics base on input data quantity and quality (especially the quality of the input raster base map is a key impact factor for performances).

The two modules which impact more on the execution time are 'Module 5' and 'Module 6' because they work with the rasters computing the `IDW` (Module 5), `as.regdata` and `regtst` (Module 6) values (as already described in Section 2.4) for each pixel.

	# Stations	Register DB length	# Regions	Map Pixel	Module 5		Module 6		Module ALL		Pixel / ALL
					sys.self	elapsed	sys.self	elapsed	sys.self	elapsed	
CHL	123	4463	8	452445	1.133	36.723	5.49	972.67	**7.88**	**1046.12**	**456**
COL	776	28188	38	2297952	7.804	501.825	40.06	5665.99	**49.88**	**6330.98**	**406**
CUB	299	11110	22	166358	0.325	18.74	1.32	235.43	**2.93**	**333.78**	**494**
URY	53	1749	1	67250	0.26	11.36	1.11	212.71	**2.01**	**250.02**	**316**
VEN	819	29567	23	1503261	5.878	373.34	20.43	3703.68	**26.81**	**4240.83**	**406**

Table 2: Some statistical results

In Table 2 we considered:

- as ‘number of Stations’ and ‘Register DB length’ the values we obtained after the initial check and cleaning of the files;
- as ‘number of Regions’ the effective number of regions created;
- as execution times for ‘Module 5’, ‘Module 6’ and ‘Module ALL’ the order of magnitude of a mean value (in seconds);
- the ‘Pixels / ALL elapsed time’ value as an estimation indicator for the performances, indicating how many pixels/second have been processed.

It is worth noting how:

- there is no direct relationship between the number of stations or the length of the register and the total elapsed time
- the ‘Module 5’ and ‘Module 6’ `sys.self` and `elapsed` times are proportionally related to the size of the input base map (which determines the size of the merged multi-band map at the end of ‘Module 5’)
- the computed ratio between the map size and the overall elapsed time could be considered quite constant with a mean value of 416 processed pixels per second (the distance from the mean is due to all the other processing instructions not strictly related to map manipulation).

In the test we made, the module 6 CPU and RAM consumption (which is the most onerous in terms of machine resources) has been monitored with a process monitor software[27] and the `Rterm.exe` process uses approximately the 50% of the CPU and 1,32 GB of memory, with peaks up to 1.57 GB.

[27] E.g. the task manager already embedded in Microsoft Windows, but you could also use valid alternatives like the Sysinternals Process Monitor or the Process Hacker available in Sourceforge.

Part 3: The R Script

3.1. The R script

As already introduced, we started in October 2012 using the `ComandosR_8regiones.r` and the `ARF-LM-VERSION2.12.r` scripts. We made several modifications to the script in order to adapt it to REFRAN-CV needs. Furthermore, we found several errors (also in parameters and into the script logic) and typos. Then we worked with the Colombian partner in order to improve the script and integrate it with new parts.

The final version of the script is quite near to the one we developed with Colombia even if we changed it again integrating pieces of the March version of the code by Jorge Nuñez and implementing suggestions arisen after the meeting in Latin America held in March 2013.

We then modified the final version of the R script to divide it into the 6 modules of REFRAN-CV (plus one introductory module only for packages management), adding all the code needed to join together again all the pieces as well as write and read all the intermediate results.

We also turned all the plots into JPEG images creation. This was due to the fact that R automatically saves plots in one single PDF file putting each image into a different page. Exporting each plot as JPEG allows us to display outputs without converting PDF pages in images.

In all the modules all the input values have been substituted with a placeholder using the following format: `@name_of_the_placeholder@`, where the name of the placeholder must be the same of the element name of the corresponding XML template file. Every time a module is executed, the placeholder is substituted with the actual value.

Furthermore, at the end of each module we added all the needed `write.csv` commands useful on one hand to save values and on the other hand to make them available to the next module. Similarly, at the beginning of each module we added `read.csv` commands to read data from the previously saved CSV files. At each run, following steps are executed:

- the `module0.template` R file is copied into the user's folder
- the running module's R template file is copied into the user's folder too and it is customized with user's parameters and data, by replacing also the placeholders with the actual values
- the two templates are merged together
- the final R script is saved and executed.

This trick allows the REFRAN-CV modules to be executed singularly or in set.

Instead, when the process is called to be executed at once, all the R template files are copied into the user's folder and merged together. In this case all the lines with unnecessary placeholders are commented. Then the resulting file is filled with the user parameters and finally the execution starts.

Moreover, we added to each module template few instructions to log the execution time for statistical purposes using the `proc.time()` command.

After each execution of a module (or a set of modules) it is possible to download the R script actually executed. This file could be opened in another R environment (like RStudio) and executed (the only required changes are the absolute paths of the input files).

It is possible to see other minor changes and choices by reading comments we put directly into the R code for each module.

Lastly, we sprinkled the R template files with particular comments following the format:

```
#MSG#Message to display
...
#MSG#This is an example message...
```

When a module is executed, the R source code is parsed and when a comment line of this kind is found, it is printed on screen to the user who can follow the execution of the R code.

3.1.1. Module0.template

We changed quite completely the code of this module to make it dynamic. Instead of having a list of `install.packages()` commands followed by a list of `library()` commands, now it reads the list of packages from the XML file as a list and cycles over this list. Also the repository can be selected from the initial form. In case of failure, a default mirror is selected. It is worth noting that this version check if a package is already installed and loaded before adding it to the list of packages to be downloaded and installed.

```
#MSG#Checking installed R packages...
InP<-installed.packages()
NeededPackages<-list(@packages@)
Repos<-"@repos@"
#JRC:: To remove packages
#for(test in NeededPackages)
# if (test %in% InP[ ,1]) {remove.packages(test)} else {}
packageList <-list()
for (NeedP in NeededPackages)
 if (NeedP %in% InP[ ,1]) {} else packageList<-c(packageList,NeedP)
if (length(packageList)>0)
 for (newpkg in packageList) {
  if (Repos != "LOCAL") {
    # Uncomment to download a local copy
    # download.packages(newpkg,"C:/path/to/folder",repos=Repos)
    install.packages(newpkg,repos=Repos)
  } else {
install.packages(list.files("@rpkgdir@",pattern=paste(newpkg,".*",sep=''),
                            full.names=TRUE,ignore.case=TRUE),repos=NULL)
  } }
```

It is worth paying attention to the order of packages in the list because some of them overwrite shared functions. As an example, packages `lmomco`, `lmom` and `lmomRFA` must be listed in this order.

Furthermore, if REFRAN-CV is running locally without an Internet connection, this script takes the needed packages from a local folder rather than from the Internet mirror.

This template has also been equipped with a couple of commands (commented by default) useful to remove all the installed packages and download again a new version of them from the repository. This functionality is especially needed when a new version of R is published thus requiring to update the entire execution environment.

3.1.2. Module1.template

In this module the main changes introduced have been related to the file format check we already discussed in Section 2.4.3. The core parts of the format check are the `f.checkFiles` function and the queries over the two input files in order to find orphaned entries.

```
f.checkFiles <- function(fileToCheck) {
  tmpFileName <- read.csv(fileToCheck,
                 sep=",",na.strings="NA",header=FALSE,nrows=1)
  if (dim(tmpFileName)[2] == 1) { # separator is not ','
    tmpFileName <- read.csv(fileToCheck,
                   sep=";",na.strings="NA",nrows=1,header=FALSE)
    if (dim(tmpFileName)[2] == 1) { #separator is not ',' neither ';'
      stop("Please check your file and provide it only
            in allowed CSV or CSV2 format")
    } else { # separator IS ';'
      checkNumFields<-count.fields(fileToCheck,sep=";",quote="\"",skip=1)
      checkWrongFields<-count.fields(fileToCheck,sep=".",quote="\"",skip=1)
      for (i in 1:length(checkNumFields)){
        if (checkNumFields[[i]]!=dim(tmpFileName)[2])
          stop(paste("Check the file: number of fields is different
                      at line: ",i+1,sep=''))
        if (checkWrongFields[[i]] > 1)
          stop(paste("Check the file: wrong field separator found
                      at line: ",i+1,sep=""))
      }
      tmpFileName <- read.csv(fileToCheck, sep=";",na.strings="NA",
                     quote = "\"", dec = ",")
    }
  } else { # separator IS ','
    checkNumFields<-count.fields(fileToCheck,sep=",",quote="\"",skip=1)
    checkWrongFields<-count.fields(fileToCheck,sep=";",quote="\"",skip=1)
    for (i in 1:length(checkNumFields)){
```

```
      if (checkNumFields[[i]] != dim(tmpFileName)[2])
        stop(paste("Check the file: number of fields is different
                   at line: ",i+1,sep=''))
      if (checkWrongFields[[i]] > 1)
        stop(paste("Check the file: wrong field separator found
                   at line: ",i+1,sep=""))
    }
    tmpFileName <- read.csv(fileToCheck, sep=",",na.strings="NA",
                   quote = "\"", dec = ".")
  }
  which(duplicated(tmpFileName)==TRUE) #EXISTEN FILAS REPETIDAS?
  tmpFileName<-unique(tmpFileName) # ELIMINAR FILAS REPETIDAS
  return(tmpFileName)
}

BaseDatosEstaciones<-f.checkFiles("@BaseDatosEstaciones@")
BaseDatosRegistros<-f.checkFiles("@BaseDatosRegistros@")
```

Furthermore, it is also responsible for the translation of files having the header in English, as already presented in Section 2.4.3.

3.1.3. Module2.template

The content of this module changed a lot from version to version. Nowadays, it is the final merge among the CAZALAC original script, the last version of the script Jorge Nuñez sent to us in March 2013 (mainly regarding implemented checks on data correctness and cleaning from NA values), the code we checked and improved with the Colombian partner and our contribution.

It is possible to say that while the analysis part more or less maintained the same structure, main changes have been introduced in Regions creation algorithm. In fact, instead of having a long list of single queries, we opted for a cycle after a single query executed on the `BaseCompletaReducida` database: `select Region from BaseCompletaReducida where Region>0 group by Region`. We opted for this choice after having noticed how both the Colombian partner and the script from Jorge Nuñez went in the direction of creating regions by using above all the value of the `Region` field as index (instead of using PMA values or Station ID). So we came out with the code in the box below.

```
numEstaciones<-sqldf("select Region from BaseCompletaReducida
                     where Region>0 group by Region")
regionTotal<-dim(numEstaciones)[[1]]
regionList<-list()
regNameList<-c()
for (regionNumber in 1:regionTotal) {
  regionName<-sqldf(paste("select id_estacion, SumaLluviaAnual from
```

```
                          BaseCompletaReducida where Region = '",
                          numEstaciones$Region[[regionNumber]],"'",
                          sep=''))
  regionName_dat<-regionName["SumaLluviaAnual"][,]
  regionName_fac<-factor(regionName["id_estacion"][,])
  isReg<-split(regionName_dat,regionName_fac)
  assign(paste("Region",numEstaciones$Region[[regionNumber]],sep=''),
               regionName)
  assign(paste("Region",numEstaciones$Region[[regionNumber]],"_dat",
               sep=''),regionName_dat)
  assign(paste("Region",numEstaciones$Region[[regionNumber]],"_fac",
               sep=''),regionName_fac)
  assign(paste("Reg",numEstaciones$Region[[regionNumber]],sep=''),isReg)
  regionList[[length(regionList)+1]]<-isReg
  regNameList[[length(regNameList)+1]]<-paste("Reg",
                     numEstaciones$Region[[regionNumber]],sep='')
}
#JRC:: To save Regions CSV
for (i in 1:length(regNameList)) {
  tmpRegName<-regNameList[[i]]
  make.names(tmpRegName)
  RegVal<-get(tmpRegName)
  max.len <- max(sapply(RegVal, length))
  corrected.list <- lapply(RegVal, function(x) {c(x,
                                   rep(NA, max.len - length(x)))})
  Reg_csv <- do.call(rbind, corrected.list)
  write.csv(t(Reg_csv),paste(tmpRegName,".csv",sep=''),
            row.names=FALSE,fileEncoding="UTF-8")
}
BaseRegiones<-regionList
NombreRegiones<-regNameList
```

3.1.4. Module3.template

Main changes in this module from the original one pertain to the merge of the Regions CSV files in order to obtain the `BaseRegiones` data structure. Then, all the analyses done in `Module2.template` are executed once more on each Region.

```
regNameList<-NombreRegiones
regionList<-list()
for (i in 1:length(regNameList)) {
  tmpCSV<-read.csv(paste("output/",regNameList[[i]],".csv",sep=""),
```

```
          fileEncoding="UTF-8",check.names=FALSE)
  colnames(tmpCSV)<-gsub("[.]","-",colnames(tmpCSV,do.NULL=FALSE))
  TtmpCSV<- t(tmpCSV)
  RegionTMP<- split(TtmpCSV,row.names(TtmpCSV))
  RegionTMP<- lapply(RegionTMP,function (x) x[!is.na(x)])
  regionList[[length(regionList)+1]]<-RegionTMP }
BaseRegiones<-regionList
```

After the execution of this module, users could decide to create also a geographical map to view stations' L-moments and 4 plots representing precipitations values (as presented in Section 2.4.5); the scripts responsible for these computations are discussed in Sections 3.1.9 and 3.1.10.

3.1.5. Module4.template

In an intermediate version of this R template we offered users the possibility to choose among three different optimization methods: `DEoptim`, `NLM`, `NLS` (as discussed in Section 2.4.6) but with the last version of the script only the `DEoptim` option works properly. However, `NLM` and `NLS` options are still present whilst commented as well as the code added to select the preferred method: `MinMethod<-"@minimizationMethod@"`. In the original version of the script we also noted how both `NLM` and `NLS` always requires `DEoptim` to be executed, because `NLM` and `NLS` need `DEoptim` parameters `paLCV`, `paLSk` and `paLKurt`.

3.1.6. Module5.template

Last version of this module includes three main changes from the original script: the `idw` computation, the 'No Data' to `NA` base map values conversion and the multi-band map creation using the `brick` function. For a detailed explanation of the `idw`, please refer to Section 2.4.7. The code we added can be seen in the following box.

```
OtherBaseMap<-readGDAL("@MapaBase@")
MapaBase<-raster(OtherBaseMap)
coordinates(BaseConsolidada_sin_NA) <- ~ Long + Lat
proj4string(BaseConsolidada_sin_NA)<- CRS("@projection@")
projection(OtherBaseMap)<- "@projection@"
fullgrid(OtherBaseMap) = FALSE
idw.out <- idw(LongRegistro ~1, BaseConsolidada_sin_NA,
               OtherBaseMap, idp=2)
idwmap<-raster(idw.out)
projection(idwmap) <- "@projection@"
xmin(idwmap)<-xmin(MapaBase)
xmax(idwmap)<-xmax(MapaBase)
ymin(idwmap)<-ymin(MapaBase)
ymax(idwmap)<-ymax(MapaBase)
```

At the beginning of the code in the box above it is possible to notice a double assignment. This is not redundant, because this module uses the base map in two different ways. Thus, we would need the base map as read from the `readGDAL` function and as raster. Indeed, the former is used by the `idw` calculation, while the latter is needed by the `brick` function (apart from being used as reference for min and max coordinates).

Before using the raster base map as a parameter for the `brick` function in order to generate the multi-band map, we need to turn all the map values 'No Data' (as passed by the user as parameter) to `NA` because certain values used as 'No Data' give problems in Module 6 for the `pixel_moments_Reg1<-as.regdata(data.frame(i,as.integer(yrs),cap1))` instruction. The code in the box below shows how we implemented this step.

```
s<-raster()
fun<-function(x) { x[x<=@noData@] <- NA; return(x) } s<-calc(MapaBase, fun)
MapaBase<-s
projection(MapaBase) <- "@projection@"
```

The creation of the multi-band GeoTIFF map was a challenging task. In the end we succeeded using the `brick` function of the `raster` package, which creates a `RasterBrick` object - a `RasterBrick` is a multi-layer raster object, typically created from a multi-layer (band) file.

```
b<-brick(MapaBase,lcvmap,lsmap,lkmap,idwmap,values=TRUE,nl=5)
bf<-writeRaster(b, filename="MergedMaps.tif", format="GTiff",
                options=c("TFW=YES","PHOTOMETRIC=RGB","INTERLEAVE=BAND"),
                overwrite=TRUE,NAflag=-999)
```

3.1.7. Module6.template

This piece of R code has been completely rewritten. The previous code to create frequency / return period maps has been discarded and substituted with a loop which creates the final maps. The code of the loop can be seen in the box below.

```
mergedMaps@data[mergedMaps@data==@noData@]=NA
fullgrid(mergedMaps) = FALSE
ndim = dim(mergedMaps)[1]
cdat = as.matrix(mergedMaps@data)
for (i in 1:ndim) { #start loop to calculate the L-moments for each pixel
cap1 = cdat[i,1:4, drop = FALSE]
yrs = cdat [i,5]
pixel_moments_Reg1<-as.regdata(data.frame(i,as.integer(yrs),cap1))
fit_test<-regtst(pixel_moments_Reg1, nsim=100)
if (i%%1000 == 0) {
  f1<-regtst(pixel_moments_Reg1, nsim=100)
  f2<-regtst(pixel_moments_Reg1, nsim=300)
  f3<-regtst(pixel_moments_Reg1, nsim=1000)
```

```
  f4<-regtst(pixel_moments_Reg1, nsim=3000)
  f5<-regtst(pixel_moments_Reg1, nsim=5000)
  f6<-regtst(pixel_moments_Reg1, nsim=10000) }
min_z<-min(abs(fit_test$Z))#finds Z values closest to Zero
best_pdf<-match(min_z,  abs(fit_test$Z))  #order:  1=glo  2=gev  3=gno  4=pe3
5=gpa
cdat[i,6]=best_pdf
cap1[2] = cap1[2]*cap1[1]
if (best_pdf==1){
  pdf_par<-pelglo(cap1) #generalized logistic
  if (createMaps) {extrem_precip<-quaglo(Pn, pdf_par)}
  GenLogis<-par.genlogis(cap1[1],cap1[2],cap1[3])
  if (mmRain!=0)
    freq_precip<-F.genlogis(mmRain, GenLogis$xi, GenLogis$alfa, GenLogis$k)
  if (Quant!=1)
    freq_precip2<-F.genlogis(Quant*cdat[i,1], GenLogis$xi, GenLogis$alfa,
                                                          GenLogis$k)
}else if (best_pdf==2){
  pdf_par<-pelgev(cap1) #generalized extreme-value
  if (createMaps) {extrem_precip<-quagev(Pn, pdf_par)}
  GEV<-par.GEV(cap1[1],cap1[2],cap1[3])
  if (mmRain!=0)
    freq_precip<-F.GEV(mmRain, GEV$xi, GEV$alfa, GEV$k)
  if (Quant!=1)
    freq_precip2<-F.GEV(Quant*cdat[i,1], GEV$xi, GEV$alfa, GEV$k)
}else if (best_pdf==3){
  pdf_par<-pelgno(cap1) #generalized normal
  if (createMaps) {extrem_precip<-quagno(Pn, pdf_par)}
  LogNorm<-par.lognorm(cap1[1],cap1[2],cap1[3])
  if (mmRain!=0)
    freq_precip<-F.lognorm(mmRain, LogNorm$xi, LogNorm$alfa, LogNorm$k)
  if (Quant!=1)
    freq_precip2<-F.lognorm(Quant*cdat[i,1], LogNorm$xi, LogNorm$alfa,
                                                        LogNorm$k)
}else if (best_pdf==4){
  pdf_par<-pelpe3(cap1) #three-parameter lognormal
  if (createMaps) {extrem_precip<-quape3(Pn, pdf_par)}
  P3<-as.matrix(pdf_par)
  P3<-t(P3)
  Pearson3<-mom2par.gamma(P3[1],P3[2],P3[3])
  if (mmRain!=0)
```

```
    freq_precip<-F.gamma(mmRain, Pearson3$xi, Pearson3$beta, Pearson3$alfa)
  if (Quant!=1)
    freq_precip2<-F.gamma(Quant*cdat[i,1], Pearson3$xi, Pearson3$beta,
                                                        Pearson3$alfa)
}else if (best_pdf==5){
  pdf_par<-pelgpa(cap1) #generalized Pareto
  if (createMaps) {extrem_precip<-quagpa(Pn, pdf_par)}
  GenPar<-par.genpar(cap1[1],cap1[2],cap1[3])
  if (mmRain!=0)
    freq_precip<-F.genpar(mmRain,GenPar$xi, GenPar$alfa, GenPar$k)
  if (Quant!=1)
    freq_precip2<-F.genpar(Quant*cdat[i,1],GenPar$xi, GenPar$alfa,
                                                      GenPar$k)
}
for (yy in 1:length(listOfYears))
  cdat[i,6+yy] = extrem_precip[yy]
if (mmRain!=0) cdat[i,11] = 1.0/freq_precip[1]
if (Quant!=1) cdat[i,12] = 1.0/freq_precip2[1]
}
```

It is necessary to go through some notes on the code.

- We changed the structure to use with `pelglo`, `pelgev`, `pelgno`, `pelpe3`, `pelgpa` which are part of the `lmom` package, and not `lmomRFA` (which commands like `regtst` and `regdata` belong to)
- This change makes the line `cap1[2] = cap1[2]*cap1[1]` necessary, as `lmom` and `lmomRFA` seems slightly incompatible with each other: the former uses the coefficient of L-variation (L-CV = λ_2/λ_1) as the second moment, while the latter use L-Sigma (or L_scale, λ_2). Data provided with the `MergedMap` passed through the `regdata` function, and thus belonging `regdata` to the `lmomRFA` package, the second column of the `MergedMap` data (an so the `cap1[2]` values) are L-CV. After this multiplication we obtain the L_scale value required by the `pel`-functions (part of the `lmom` package).
- The `lmom` package `qua`-functions (`quagpa`, `quaglo`, `quape3`, and so on) compute the quantile function (inverse cumulative distribution function) of the distribution, but since the probabilities were calculated based on the return periods, they return the precipitations associated with the respective return periods. The first parameter `Pn` is a list of estimations of the probability of non-exceedance as a function of the return period set by the user.
- Then the script executes the `par.`-functions[28] to obtain the parameters `xi`, `alfa`, `k` to be passed to the `F.`-functions. These functions are used to create the return period maps if

[28] The documentation of `nsRFA` package version 0.7-10 (which comes with the R version 3.0.1) says that the

users pass desired precipitation and/or quantile values to the script. It is worth noting that in case of too high input values for the `mmOfRain` field, this algorithm could produce `NA` values which will be displayed in the map as empty pixels. This is due because log-based functions (e.g. GEV) will execute the `log(1-k(x-ξ)/α)` formula with k, ξ and α parameters computed on the original map values while `x` is the new desired precipitation value set by the user and it should satisfy the following conditions for every pixel[29] to give a real value:

- $-\infty < x \leq \xi + \alpha / k$ if $k>0$;
- $-\infty < x < \infty$ if $k=0$;
- $\xi + \alpha / k \leq x < \infty$ if $k<0$.

- Using a parameterized `for` cycle, all computed values are saved into the `cdat` matrix.
- As already stated, it could be possible in a future release to slice the merged map in order to compute the final calculation on smaller sets of data and then merge the results together again.

3.1.8. moduleImg.template

This template has been developed from scratch to fulfil the requirement of providing the customization of the generated maps. It takes all the parameters specified by the user and plots a new map. An excerpt of the code can be seen in the box below for convenience.

```
Mapa<-readGDAL("output/@inputmap@") # Load GeoTIFF
tmpMap<-raster(Mapa)
IMG_Projection <- "@img_projection@" # Set projection
if (IMG_Projection == "NA") {
IMG_Projection <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs" }
projection(tmpMap) <- IMG_Projection # Set projection
jpeg(filename="@img_title@.jpg", width = @img_resolution@,
  height = @img_resolution@, pointsize=5, quality = 100, units = "px",
  bg="white", res = @img_ppi@, type = "cairo")
lwd<-0.5 # line width
```

`par.gamma` function now "gives the parameters (mu, sigma, gamm)" and it introduces the function `mom2par.gamma` which "returns the parameters α, β and ξ, given the parameters (moments) μ, σ, γ". Thus we changed the code executing first `mom2par.gamma` passing λ_1, λ_2, τ_3 and then executing `par.gamma` with the `mom2par.gamma` output. However, we discovered that `par.gamma` gives actually α, β, ξ, μ, and σ (and γ is missing) but α, β, ξ are actually `NA`. In the end we put `pdf_par` (which has the μ, σ, γ values) in a matrix and we used its values as input for `mom2par.gamma` function to obtain α, β, ξ. It is clear this is not a good solution but for the time being it seems the only working solution. We also sent an e-mail to the `nsRFA` package maintainer to inform him about this problem.

[29] However, it implies also to execute again most of the functions and this is not in the scope of this version of REFRAN-CV; indeed, we aim just to give an estimation.

```
par(mar=c(5, 5, 4, 3) + 0.1)
if ((maxValue(tmpMap)) < 100) {
     maxCol<-maxValue(tmpMap)+100} else {maxCol<-maxValue(tmpMap)}
maxCol<-round(maxCol)
plot(tmpMap,main="@img_title@",col=@img_palette@(maxCol),
  tcl=1,xlab="@xlab@",ylab="@ylab@",cex.main=2,cex.lab=2,cex.axis=2,
  legend=T,legend.args=list(text="@img_legend@",
  side=4, font=3, line=2.5, cex=1.2))
xaxe<-seq(floor(xmin(tmpMap)),ceiling(xmax(tmpMap)),0.5)
xaxe2<-seq(floor(xmin(tmpMap)),ceiling(xmax(tmpMap)),0.1)
axis(1, at=xaxe,tcl=0.4,labels=F)
axis(1, at=xaxe2,tcl=0.2,labels=F)
axis(2, at=seq(floor(ymin(tmpMap)),ceiling(ymax(tmpMap)),0.5),tcl=0.4)
axis(2, at=seq(floor(ymin(tmpMap)),ceiling(ymax(tmpMap)),0.1),tcl=0.2)
Grid<-"@img_grid@"
if (Grid == "yes") {
 abline(v=(seq(floor(xmin(tmpMap)),ceiling(xmax(tmpMap)),0.5)),
           col="lightgrey", lty="dotted", lwd=lwd)
 abline(h=(seq(floor(ymin(tmpMap)),ceiling(ymax(tmpMap)),0.5)),
           col="lightgrey", lty="dotted", lwd=lwd)
}
dev.off()
```

3.1.9. displayGraphs.template

This template has been developed from scratch to fulfil the requirement of providing the generation of four different plots taking as input all the historical values of stations as saved in their registers. In the box below we present the main part of the script.

```
ListOfStations<-c(@stationsList@) # The stations to be represented
numCol <- length(ListOfStations)
regiony <- read.csv("@BaseCompletaFile@")
MeanMonthPrecTOT <- sqldf("select id_estacion, avg(ENE) ENE, avg(FEB) FEB,
                    avg(MAR) MAR, avg(ABR) ABR, avg(MAY) MAY, avg(JUN) JUN,
                    avg(JUL) JUL, avg(AGO) AGO, avg(SEP) SEP, avg(OCT) OCT,
                    avg(NOV) NOV, avg(DIC) DIC
                    from regiony group by id_estacion")
MeanMonthPrec <- subset(MeanMonthPrecTOT, id_estacion %in% ListOfStations)
MeanMonthPrec$id_estacion <- factor(MeanMonthPrec$id_estacion)
MeanMonthPrec.m<-as.matrix(MeanMonthPrec[2:13])
rownames(MeanMonthPrec.m)<-MeanMonthPrec[,1]
maxMeanValue<-ceiling(range(MeanMonthPrec.m)[2])+50
```

```
barplot(MeanMonthPrec.m, beside=TRUE,col=rainbow(numCol),
        ylim=c(0,maxMeanValue),
        ylab="mm of Rain",
        xlab="Months",
        main=paste("Average Monthly Precipitation for ",iso_pais,sep=''))

for (numST in 1:length(ListOfStations)) {
   dataSt<-as.zoo(subset(regiony,id_estacion==ListOfStations[numST],
           select=c('Anio','ENE','FEB','MAR','ABR','MAY','JUN',
                    'JUL','AGO','SEP','OCT','NOV','DIC')))
   for (years in minYSt:maxYSt)
     valuesSt <- c(valuesSt,subset(dataSt, Anio == years, select=ENE:DIC))
   valuesStTS <- as.zoo(ts(valuesSt, start=c(minYSt, 1),
                         end=c(maxYSt, 12), frequency=12))
   maxvalues <- if (max(dataSt[,2:13])>maxvalues) max(dataSt[,2:13])
                else maxvalues
   tmpData[[numST]]<-valuesStTS
}

finalData<-do.call(merge,tmpData)
colnames(finalData)<-ListOfStations

split.screen(c(ncol(finalData),1))
par(oma=c(4,2,3,4))
for(i in 1:ncol(finalData)) {
  screen(i)
  par(mar = c(0, 4, 0, .5))
  plot.ts(finalData[,i], axes=FALSE, type = "l",
          col=i+1, ylim=c(0,maxvalues+10), ylab="",xlab="")
}
close.screen(all.screens = TRUE)
plot(finalData,xlab="Year",
     main="Stations comparison (mm of Rain)",col='darkblue')
plot(finalData, plot.type="single", col = 2:(ncol(finalData)+1),
     xlab="Years",
     ylab="mm of Rain",
     main="Graph Overlay Comparison")
```

3.1.10. displayMaps.template

This template has been developed from scratch to fulfil the requirement of providing the geographical positioning on a map the stations belonging to the selected regions and their L-moments. In the box below we present the main part of the script.

```
tm_world <- readShapePoly("tm_world2.shp",proj4string=CRS("+proj=longlat"))
if (country == "XXX") {
  c2 <- tm_world
} else {
  c2 = tm_world[tm_world$iso3 %in% country,]
}
proj4string(c2)<- CRS("+proj=latlong +ellps=WGS84")
#####Data base to be plotted#######
regionx <- read.csv("@BaseDatosTotal@")
inter <- sqldf("select * from regionx where Region in (@regionsList@)")
coordinates(inter) <- ~ Long + Lat
proj4string(inter)<- CRS("+proj=latlong +ellps=WGS84")
#####Number of classes#########
nn=length(inter)
if(nn<=1) {
  stop("Wrong number of Stations")
} else if(nn<=10) {
  nclr<-nn
} else if(nn<=20) {
  nclr<-4
} else {
  nclr <- 8
}
numcols<-nlevels(inter$Region)
thisRegionStations <- cbind(as.character(inter$Region),
                      as.character(inter$id_estacion))
colnames(thisRegionStations)<-c("Region","id_estacion")
thisRegionStations<-thisRegionStations[ order(thisRegionStations[,1],
                                        thisRegionStations[,2]), ]
sl1 <- list ('sp.polygons', c2)
sl2 <- list ('sp.pointLabel',inter, label=inter$id_estacion,
              cex=0.5,fontface=2)

if (country != "XXX") {
  jpeg("module3_StationsMap.jpg", width = 3600, height = 3600, ...
} else {
```

```
  jpeg("module3_StationsMap.jpg", width = 7200, height = 7200, ...
}
print(spplot(inter, c("PMA"), cuts=nclr, sp.layout=list(sl1,sl2),
             pch = 16, cex = .5,
             col.regions=rainbow(nclr,start=.7,end=.1),
             xlim=c(x2.min,x2.max),ylim=c(y2.min,y2.max),
             main="PMA Map"),position=c(0,0,1,.75),
             split = c(1,1,2,2), more=TRUE)
print(spplot(inter, c("L_CV"), cuts=nclr,sp.layout=list(sl1,sl2),
             pch = 16, cex = .5,
             col.regions=rainbow(nclr,start=.7,end=.1),
             xlim=c(x2.min,x2.max),ylim=c(y2.min,y2.max),
             main="L-CV Map"),position=c(0,0,1,.75),
             split = c(2,1,2,2), more=TRUE)
print(spplot(inter, c("L_Kurtosis"), cuts=nclr,sp.layout=list(sl1,sl2),
             pch = 16, cex = .5,
             col.regions=rainbow(nclr,start=.7,end=.1),
             xlim=c(x2.min,x2.max),ylim=c(y2.min,y2.max),
             main="L-Kurt Map"),position=c(0,0,1,.75),
             split = c(1,2,2,2), more=TRUE)
print(spplot(inter, c("L_Skewness"), cuts=nclr,sp.layout=list(sl1,sl2),
             pch = 16, cex = .5,
             col.regions=rainbow(nclr,start=.7,end=.1),
             xlim=c(x2.min,x2.max),ylim=c(y2.min,y2.max),
             main="L-Skew Map"),position=c(0,0,1,.75),
             split = c(2,2,2,2), more=TRUE)
print(spplot(inter, c("Region"), sp.layout=list(sl1),
             pch = 16, cex = .5,
             col.regions=rainbow(numcols,start=.1,end=.9),
             xlim=c(x.min,x.max),ylim=c(y.min,y.max),
             main="Region"),position=c(0,.75,1,1), more=FALSE)
```

It is worth noting in this template the algorithm to choose the right number of classes into which split all the values. For less than 10 values, all of them are considered separately. For a number of values between 11 and 20, they are grouped in 4 classes; while for regions with more than 20 stations 8 classes of values are created. We decided to fix the maximum number of visualization classes to 8, because a higher number of classes will start losing in readability and meaning.

Furthermore, it is worth notice the check on the country code and the differences in creating the stations map.

Part 4: Roadmap and Future Works

4.1. Roadmap and future development

In this Section we summarize on one hand the features still missing to fulfil all the initial requirements and desiderata and on the other hand some improvements and suggestions for further developing.

4.1.1. The core PHP code

The PHP code changed a lot during the development phase. We had not enough time to clean and optimize it. Therefore, all the functions should be checked as well as the PHP code embedded into the front-end pages. There are few orphaned pages that should be deleted and the whole structure of the application should be renewed (e.g., by moving all the code from the `mockup` folder to another more appropriate location). Furthermore, some folder name should be changed (e.g., the `html` folder now includes only PHP files). There are pieces of code which are repeated almost identical in different places: they could be rewritten in order to use a function with parameters or, even better, PHP classes.

We opted for using PHP sessions instead of a RDBMS in order to gain a simplified data persistence mechanism on one side and no need to create and save user credentials on the other side. However, we suggest checking if this solution is still the more appropriate for the aims of the REFRAN-CV application, keeping in mind the REFRAN-CV web application could be released in a near future as a module to be integrated in a web-portal (like Aquaknow). This means it could inherit the user management system of its containing web-portal.

This first version of the REFRAN-CV application has been coded following the functional programming methodology. We strongly suggest taking into account the opportunity to migrate toward an object-oriented approach. More in details, some PHP classes could be introduced in order to handle logs, working directories, input forms creation and management, filesystem operations, interactions with the R environment. Moreover, most common patterns could be adopted upon specific needs (e.g., Strategy, Singleton, Factory, and so forth).

At last, logs have been introduced as requested but log writing could be improved by removing or adding more logged events, changing the content of a log or the type of a log.

Finally, add the possibility to run subset of modules as requested. In this case only subsequent modules (e.g. from 1 to 3, from 3 to 6) can be run.

4.1.2. The R templates

The current R code is actually a patchwork which took pieces from many different sources and mixed them up with new lines of code. The resulting R script has been then sliced into 6 template files. Therefore, the R code should now be checked and optimized. In particular:

- there may be present unused variables;
- concordance of names should be checked;
- variables and data structures should be deleted after their use;

- functions could be created to group together repeated lines of code.

The localization in English of the REFRAN-CV application is foreseen so many variable names should be changed accordingly.

We strongly discourage the use of external tools (like RStudio) to make changes to the R templates, also because they cannot be executed singularly: each module needs input values and the substitution of the placeholders. Tools like RStudio can be used but we suggest checking the resulting R code by hand before executing it in REFRAN-CV.

Input file format check as defined in `module1.template` needs further refinement. It could be useful not only to check few fields but the whole header line, both words and special characters like "`-`" and "`_`". Furthermore, add a check into `module1.template` to control whether the input files are localized correctly. Finally, add a stop warning message to warn users if wrong format is found.

It could be useful if REFRAN-CV could also check if the installed R version is the last available. It is important (especially if a user chooses to use its own R engine) that the R version in use is at least the same of the one used to test and execute REFRAN-CV. Using an older version (or even a newer version) of the R executable could raise errors and warnings and versions mismatches.

Regarding module 6, it could be possible in a future release to slice the merged map in order to compute the final calculation on smaller sets of data and then merge the results together again. This way there could be possible to execute the REFRAN-CV also on computer with fewer hardware resources.

4.1.3. The (Graphical) User Interface

REFRAN-CV has been developed starting from a mock-up. One of the objectives was to deploy the REFRAN-CV web application into another framework as a module. Therefore, we put no particular effort in changing the general layout but rather we focused on the controller and model tiers. Now we suggest restructuring the graphical user interface, both in case REFRAN-CV will be deployed into another framework (e.g. Aquaknow) and in case it will be published as a stand-alone application. The input forms should be fixed and cleaned. Moreover, they could become more interactive.

One of the main future steps concerns the software localization, both the graphical user interface and the back-end (the PHP code, XML files, R templates and the log content). All the variable names into the R script should be turned in English as well as into the XML and PHP functions accordingly. It could be worth providing a way to easily add more languages and translations, e.g. by providing a set of arrays to translate input and output from/to English as we already did for the module 1 input files.

Check also the Cascading Style Sheets of the application (especially those referring to the forms style) and test them on major browsers (e.g., labels alignment in forms differs from one browser to the other). Each piece of code used to develop REFRAN-CV came with its own CSS file: they should be now reviewed and their number decreased.

Similarly, all the Javascript files should be reviewed: some of them could be deleted while others could be merged together or at least cleaned from worthless code.

4.1.4. Management of the project

During the REFRAN-CV development we decided to use a public Internet GIT repository as versioning system. For the moment, we kept the repository access restricted. Nonetheless, we strongly suggest from now on choosing a shared repository with versioning system in order to maintain a centralized development environment. By setting up a common versioned repository users could be able to commit changes both to R and to PHP code or at least to gain access to the last version of the application.

Conclusions

The REFRAN-CV development allowed us to prove the effectiveness of creating a PHP web-application upon an R environment. REFRAN-CV is at this moment a complete portable environment enough mature and stable to be tested with different datasets, even without an Internet connection available. Its architecture could also be adapted to host different R scripts. The final result achieved so far could be considered as a third alpha version. There is a clear need of refinements and tests. We did not meet all the initial requirements because of many changes occurred also during the final phase, forcing us to implement changes to the structure instead of adding last remaining minor features. However, REFRAN-CV passed a big on-field test being presented to some Latin America partners which successfully created final result maps using their previously untested datasets. We strongly suggest finishing the development of a first stable and complete release of REFRAN-CV before going further. It becomes ever more important to set-up a common versioned environment to focus resources from all over the World instead of wasting time and capabilities in creating different partial and personalized versions. In the end, we encourage to carefully analyse the next steps to follow, especially whether to release REFRAN-CV as a framework module or as a stand-alone application.

European Commission
Joint Research Centre – Institute for Environment and Sustainability

Title: The REFRAN-CV Web Application:Technical Documentation, Install Instructions, Admin and User Guide.

Author(s): Michele Chinosi, Juan Arevalo Torres, César Carmona Moreno

2013 – 64 pp. – 21.0 x 29.7 cm
EUR – Scientific and Technical Research series

Abstract

This document is intended to be complete technical reference documentation for the REFRAN-CV web application. It is divided into 4 parts. Part 1 describes the requirements for installing and executing the web application (also considering different Operating Systems) as well as results of the compatibility tests performed. It also offers a Quick Execution section to start using the web application in a minute. Part 2 presents the web application roadmap, its design principles stressing on compatibility with previous version and the reference description document, the structure of the application (with a detailed description of each module) and of the folders. Part 3 is completely dedicated to the R code, highlighting all the changes, the criticisms and corrections done with respect to the original version, with a Section dedicated to the test data used while building the application. Part 4 is for the future works list, being the application at this stage only an incomplete prototype, with a future suggested roadmap and other remarks. Conclusions close the document.

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new standards, methods and tools, and sharing and transferring its know-how to the Member States and international community.

Key policy areas include: environment and climate change; energy and transport; agriculture and food security; health and consumer protection; information society and digital agenda; safety and security including nuclear; all supported through a cross-cutting and multi-disciplinary approach.