Refining Remote Sensing precipitation Datasets in the South Pacific: An Adaptive Multi-Method Approach for Calibrating the TRMM Product

Submitted to Hydrology and Earth System Sciences (HESS)

O.Mirones, J.Bedia, S.Herrera, M.Iturbide & J.Baño-Medina

2023-06-23

Abstract

This notebook provides a comprehensive illustration of the entire adaptive calibration process, including available data download from an open repository and the computation of both standard and adaptive calibration RF scores.

Contents

Used packages. Installing climate4R

For complete reproducibility of the results, it is highly recommended to install the specific versions of the packages used in order to avoid compilation errors. All subsequent operations are carried out using the core packages of climate4R. The appropriate package versions are indicated by their version tags, and the installation can be done using the `install_github` function from the `devtools` package (*Wickham et al., 2020*).

```r
###installation of libraries
library(devtools)
install_github(c("SantanderMetGroup/loadeR.java@1.1.1",
                 "SantanderMetGroup/climate4R.UDG",
                 "SantanderMetGroup/loadeR@1.7.0",
                 "SantanderMetGroup/transformeR@time_res",
                 "SantanderMetGroup/visualizeR",
                 "SantanderMetGroup/downscaleR@3.3.3",
                 "SantanderMetGroup/climate4R.value"))
```

```r
###loading libraries
library(loadeR)
library(transformeR)
library(downscaleR)
library(climate4R.value)
```

Adaptive calibration process. An illustrative example at Kolopelu station.

Loading and harmonizing data

First, we load the required dataset resulting from the study in *Mirones et al.(2022)* located at Zenodo open repository [https://zenodo.org/rec ord/7014397/files/South_Pacific_precipitation.zip].

```
###download of the dataset from Zenodo repository
temp <- tempfile(fileext = ".zip")
download.file("https://zenodo.org/record/7014397/files/South_Pacific_precipitation.zi
```

The downloaded dataset comprises a collection of stations located in the South Pacific region, obtained from the **PACRAIN** (Pacific Rainfall Database) [http://pacrain.ou.edu/]. Additionally, precipitation series from the **TRMM** (Tropical Rainfall Measuring Mission) satellite [https://trmm.gsfc.nasa.gov/] and the **ERA5** reanalysis [https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5] have been extracted for the following stations: *Kolopelu* (*Wallis and Futuna*), *Alofi* (*Niue*), *Rarotonga* (*Cook Islands*), *Raoul Island* (*New Zealand*), *Port Vila* (*Vanuatu*), *Aoloau*, and *Nuu'uli* (*American Samoa*). Daily variables are included for the stations with the following IDs: *NZ75400*, *NZ82400*, *NZ84317*, *NZ99701*, *SP00646*, *US14000*, and *US14690*.

- TRMM raw satellite precipitation
- ERA5 raw reanalysis precipitation
- PACRAIN rain gauge precipitation
- Weather type associated
- TRMM calibrated (scaling)
- TRMM calibrated (empirical quantile mapping)
- TRMM calibrated (scaling conditioned)

- TRMM calibrated (empirical quantile mapping conditioned)

where the last four are a consequence of the application of 4 different calibration methods.

The required variables are *rain gauge precipitation* (`pr`), *satellite precipitation* (`pp_trmm`) and the *WTs* (`wt`). Thus, we set the corresponding variable names in the argument `var` when calling to `loadStationData` from `loadeR`:

```
###extracting the variables from the dataset
obs <- loadStationData(dataset = temp, var = "pr")
trmm <- loadStationData(dataset = temp, var = "pp_trmm")
wt <- loadStationData(dataset = temp, var = "wt")
```

To select the station for analysis, you can use the `station.id` argument in the `subsetGrid` function from the `transformeR` package. The station IDs can be found in `$Metadata`. We replicate the methodology outlined in the article for the *Kolopelu* station, as the code for implementing the adaptive methodology is the same for all stations. The only difference lies in the choice of the `station.id` argument in the `subsetGrid` function.

```
###selecting the station to be analyzed
obs <- subsetGrid(obs, station.id = obs$Metadata$station_id[1])
trmm <- subsetGrid(trmm, station.id = obs$Metadata$station_id)
wt <- subsetGrid(wt, station.id = obs$Metadata$station_id)
```

Lastly, we perform an intersection of the rain gauge and TRMM data to account for potential differences in their time coverage. To achieve this, we utilize the `intersectGrid` function from the `transformeR` package. The `which.return` argument specifies which grid should be returned, encompassing the overlapping time period. In this case, the observation series is returned as the target period for calibration is based on the TRMM data.

```
obs <- intersectGrid(obs, trmm, which.return = 1)
```

Adaptive calibration methodology and evaluation

The computation of the adaptive calibration for the Kolopelu TRMM series is carried out using a function called `adaptiveCalibration`, which encapsulates the adaptive calibration methodology presented in the article.

For a proper evaluation and comparison of different calibration methods (scaling, eQM, pQM, gpQM-95, and gpQM-75), a consistent scoring methodology is employed, as outlined by *Kotlarski et al. (2019)*.

To ensure code reproducibility, the `./utils` directory has been provided, which contains the auxiliary files `adaptiveCalibration.R`, `RFScore.R`, and `MaxReturnValue.R` necessary for implementing the adaptive calibration methodology.

In this methodology, the calibration methods are applied individually for each weather type (WT), and the final calibrated series is determined by selecting the best calibration method for each WT. This process results in a single time series that covers the entire period.

To determine the best calibration method for each WT, a score calculation *based on Kotlarski et al. (2019)* is utilized. The score computation follows these steps:

1. Climate indices from the VALUE database (http://www.value-cost.eu/validationportal/app/#!indices) are calculated using the climate4R.value package, which serves as a validation reference.

2. The absolute difference between the observed series and the climate indices is computed.

3. The difference in absolute value for each index is normalized across all methods. The resulting score for each method is the average of the normalized values of the climate indices.

This process of the score calculation is encapsulated in a atomic function called `RFScore`. In `RFScore` is possible to assing arbitrary weights through `weights` argument to the indices involved in the RF score, giving more importance to specific precipitation characteristics, such as the representation of extremes. The weighting aims to guide the calibration towards better adjustment in the upper tail of the distribution, thereby achieving improved correction for extreme precipitation events beyond a certain threshold. This allows obtaining a score that focuses more on specific indices, such as extreme precipitation (e.g., *R20p* or *P98WetAmount*).

The cross-validation scheme and calibration methods are implemented within the calibration and empirical statistical downscaling tools provided by the `downscaleR`package *(Bedia et al., 2020)* in the `climate4R` framework.

The validation of the results is based on the VALUE climatic indices. However, the `adaptiveCalibration` function allows the inclusion of other user-defined indices using the `custom_function` argument, such as the *MaxReturnValue* index.

Finally, the adaptive calibration is performed using the scaling, eQM, pQM, and gpQM (with 0.95 and 0.75 thresholds) techniques. These methods are also computed independently to evaluate whether the adaptive calibration results in improved performance compared to the standard calibration methods.

```
###load of auxiliar functions
source("./utils/adaptiveCalibration.R")
source("./utils/RFscore.R")
source("./utils/MaxReturnValue.R")
```

```
###adaptive calibration computations
cal <- adaptiveCalibration(obs = obs, sim = trmm,
                           clustering = wt$Data,
                           methods = c("scaling", "eqm", "pqm",
                                       "gpqm", "gpqm"),
                           index = c("Skewness","Mean","SDII",
                                     "R10","R10p","R20","R20p",
                                     "P98Wet","P98WetAmount"),
                           weights = NULL,
                           custom_function = list(MaxReturnValue),
                           scaling.type = "multiplicative",
                           window = NULL,
                           theta = c(.95, .75))
```

```
###adaptive calibration including weigths
w <- c(.05, .05, .05, .05, .05, .15, .15, .15, .2, .1)
cal_w <- adaptiveCalibration(obs = obs, sim = trmm,
                             clustering = wt$Data,
                             methods = c("scaling", "eqm", "pqm",
                                         "gpqm", "gpqm"),
                             index = c("Skewness","Mean","SDII",
                                       "R10","R10p",
                                       "R20","R20p","P98Wet",
                                       "P98WetAmount"),
                             weights = w,
                             custom_function = list(MaxReturnValue),
                             scaling.type = "multiplicative",
                             window = NULL,
                             theta = c(.95, .75))
```

Computation of standard calibration techniques

After calculating the series calibrated using the adaptive methodology, it is essential to compute the series of standard calibrations to assess their performance and compare them with the adaptive methodology. The `downscaleR` package provides the `biasCorrection` function, which offers specific arguments based on the chosen calibration method:

1. The `multiplicative` argument determines the type of scaling method employed. It can be set to `additive` for unbounded variables like temperature or `multiplicative` for bounded variables like precipitation. If the bias correction method is not set to `scaling`, this argument is disregarded.

2. The `fitdistr.args` argument allows us to specify additional arguments passed to the `fitdistr` function, such as `densfun`, `start`, and others. This argument is only used when applying the `pqm` method.

3. The `theta` argument defines the upper threshold (and lower threshold for the left tail, if necessary) above which values are fitted to a Generalized Pareto Distribution (GPD). Values below this threshold are fitted to a gamma distribution. By default, `theta` corresponds to the 95th percentile (and 5th percentile for the left tail). This argument is exclusive to the `gpqm`method.

Hence, we compute the standard calibration for the scaling, eQM, pQM, and gpQM methods. In the case of the gpQM method, two series are generated: gpQM-95 and gpQM-75. The gpQM-95 series fits the data to a GPD above the 95th percentile, while the gpQM-75 series fits the data above the 75th percentile.

```
obs <- setGridDates.asPOSIXlt(obs,
                              tz = "UTC")
trmm <- setGridDates.asPOSIXlt(trmm,
                               tz = "UTC")
```

```r
scaling <- biasCorrection(y = obs, x = trmm,
                          precipitation = TRUE,
                          method = "scaling",
                          scaling.type = "multiplicative",
                          window = NULL,
                          cross.val = "loo")
```

```r
eqm <- biasCorrection(y = obs, x = trmm,
                      precipitation = TRUE,
                      method = "eqm",
                      cross.val = "loo")
```

```r
pqm <- biasCorrection(y = obs, x = trmm,
                      precipitation = TRUE,
                      method = "pqm",
                      cross.val = "loo")
```

```r
gpqm95 <- biasCorrection(y = obs, x = trmm,
                         precipitation = TRUE,
                         method = "gpqm",
                         theta = .95,
                         cross.val = "loo")
```

```r
gpqm75 <- biasCorrection(y = obs, x = trmm,
                         precipitation = TRUE,
                         method = "gpqm",
                         theta = .75,
                    cross.val = "loo")
```

Ranking Framework (RF) Score and selection of the best calibrated series

To evaluate and compare the performance of different calibration methods for a specific weather type (WT), we extract the relevant section from the `adaptiveCalibration` function where the RF Score is computed. This allows us to assess whether the adaptive methodology improves upon the results obtained from standard calibration methods.

The RF score is calculated using the `RFscore` function. The `obs` argument corresponds to the rain gauge series, while the `series` argument represents the list of calibrations from which the score is computed. In the `index` argument, we specify the validation indices obtained from VALUE, including the `MaxReturnValue`, which can be included as part of the validation indices using the `custom_function` argument.

By employing this approach, we can determine the best calibrated series by comparing their RF scores. This enables us to assess whether the adaptive methodology yields improvements over the standard calibration methods.

```
###computation of RF Score
score <- RFscore(obs = obs,
                 series = list(scaling, eqm, pqm, gpqm95, gpqm75,
                               cal),
                 index = c("Skewness","Mean","SDII","R10","R10p",
                           "R20","R20p","P98Wet","P98WetAmount"),
                 custom_function = list(MaxReturnValue),
                 methods = c("scaling", "eqm", "pqm", "gpqm95",
                             "gpqm75", "adaptive"))
```

```
###computation of RF Score for the weighted calibration
score_w <- RFscore(obs = obs,
                series = list(scaling, eqm, pqm, gpqm95, gpqm75,
                              cal_w),
                index = c("Skewness","Mean","SDII","R10","R10p",
                          "R20","R20p",
                          "P98Wet","P98WetAmount"),
                custom_function = list(MaxReturnValue),
                methods = c("scaling", "eqm", "pqm", "gpqm95",
                            "gpqm75",
                            "adaptive"),
                weights = w)
```

```
print(score)
```

```
##   scaling       eqm       pqm    gpqm95    gpqm75  adaptive
## 0.3715306 0.5870249 0.5321737 0.5169097 0.5628998 0.7317244
```

```
print(score_w)
```

```
##   scaling       eqm       pqm    gpqm95    gpqm75  adaptive
## 0.3249935 0.5027176 0.5314995 0.5017685 0.6817913 0.7805111
```

The results of the evaluation indicate that the adaptive methodology yields a significantly better score compared to the other calibration techniques. This finding reinforces the effectiveness and value of utilizing adaptive calibration in this context. Furthermore, the capability to customize the calibration by applying arbitrary weights to specific indices offers increased flexibility in determining the optimal combination of methods that align with the unique characteristics of each site. This adaptability further enhances the overall calibration process.

References

- Bedia, J., Baño-Medina, J., Legasa, M. N., Iturbide, M., Manzanas, R., Herrera, S., Casanueva, A., San-Martín, D., Cofiño, A. S., and Gutiérrez, J. M.: Statistical downscaling with the downscaleR package (v3.1.0): contribution to the VALUE intercomparison experiment, Geoscientific Model Development, 13, 1711–1735, https://doi.org/10.5194/gmd-13-1711-2020, 2020.
- Maraun, D., Widmann, M., Gutiérrez, J. M., Kotlarski, S., Chandler, R. E., Hertig, E., Wibig, J., Huth, R., and Wilcke, R. A.: VALUE: A framework to validate downscaling approaches for climate change studies, Earth's Future, 3, 1–14, https://doi.org/https://doi.org/10.1002/2014EF000259, 2015
- Kotlarski, S., Szabó, P., Herrera, S., Räty, O., Keuler, K., Soares, P. M., Cardoso, R. M., Bosshard, T., Pagé, C., Boberg, F., Gutiérrez, J. M., Isotta, F. A., Jaczewski, A., Kreienkamp, F., Liniger, M. A., Lussana, C., and Pianko-Kluczynska, K.: Observational un-

certainty and regional climate model evaluation: A pan-European perspective, International Journal of Climatology, 39, 3730–3749, https://doi.org/https://doi.org/10.1002/joc.5249, 2019.

- Mirones, O., Bedia, J., Fernández-Granja, J. A., Herrera, S., Van Vloten, S. O., Pozo, A., Cagigal, L., and Méndez, F. J.: Weather-type-conditioned calibration of Tropical Rainfall Measuring Mission precipitation over the South Pacific Convergence Zone, International Journal of Climatology, pp. 1–18, https://doi.org/https://doi.org/10.1002/joc.7905, 2022.
- Wickham, H., Hester, J. and Chang, W., 2020. devtools: Tools to Make Developing R Packages Easier. R package version 2.3.0. https://CRAN.R-project.org/package=devtools

Session info

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
```

```
##  [1] LC_CTYPE=es_ES.UTF-8          LC_NUMERIC=C
##  [3] LC_TIME=es_ES.UTF-8           LC_COLLATE=es_ES.UTF-8
##  [5] LC_MONETARY=es_ES.UTF-8       LC_MESSAGES=es_ES.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8          LC_NAME=es_ES.UTF-8
##  [9] LC_ADDRESS=es_ES.UTF-8        LC_TELEPHONE=es_ES.UTF-8
## [11] LC_MEASUREMENT=es_ES.UTF-8    LC_IDENTIFICATION=es_ES.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] climate4R.value_0.0.2 VALUE_2.2.2           downscaleR_3.3.3
##  [4] transformeR_2.1.4     loadeR_1.7.0          climate4R.UDG_0.2.0
##  [7] loadeR.java_1.1.1     rJava_0.9-11          devtools_2.3.2
## [10] usethis_1.6.3
##
## loaded via a namespace (and not attached):
##  [1] viridis_0.6.2       pkgload_1.3.2       maps_3.4.1
##  [4] splines_3.6.3       jsonlite_1.8.3      viridisLite_0.4.1
##  [7] foreach_1.5.1       dotCall64_1.0-2     kohonen_3.0.11
## [10] assertthat_0.2.1    sp_1.6-0            deepnet_0.2
## [13] yaml_2.3.6          remotes_2.2.0       sessioninfo_1.1.1
## [16] pillar_1.8.1        lattice_0.20-41     sticky_0.5.6.1
## [19] glue_1.6.2          reticulate_1.26     RcppEigen_0.3.3.9.3
## [22] digest_0.6.30       colorspace_2.1-0    htmltools_0.5.0
## [25] Matrix_1.5-1        pkgconfig_2.0.3     scales_1.2.1
## [28] processx_3.7.0      CircStats_0.2-6     proxy_0.4-26
## [31] dtw_1.22-3          tibble_3.1.8        generics_0.1.3
## [34] ggplot2_3.4.0       ellipsis_0.3.2      withr_2.5.0
## [37] verification_1.42   pbapply_1.5-0       cli_3.6.0
## [40] survival_3.2-7      magrittr_2.0.3      crayon_1.5.1
## [43] memoise_1.1.0       evaluate_0.15       ps_1.7.1
## [46] fs_1.5.0            fansi_1.0.4         MASS_7.3-53
## [49] pkgbuild_1.1.0      tools_3.6.3         prettyunits_1.1.1
## [52] lifecycle_1.0.3     stringr_1.4.0       munsell_0.5.0
## [55] glmnet_4.1-3        callr_3.5.1         signal_0.7-7
## [58] akima_0.6-3.4       compiler_3.6.3      evd_2.3-3
## [61] rlang_1.1.1         grid_3.6.3          RCurl_1.98-1.5
## [64] iterators_1.0.13    rstudioapi_0.14     spam_2.9-1
## [67] bitops_1.0-7        rmarkdown_2.5       boot_1.3-25
## [70] codetools_0.2-16    gtable_0.3.1        abind_1.4-5
## [73] R6_2.5.1            gridExtra_2.3       knitr_1.39
## [76] dplyr_1.0.9         utf8_1.2.3          rprojroot_2.0.3
## [79] shape_1.4.6         desc_1.2.0          stringi_1.5.3
## [82] parallel_3.6.3      Rcpp_1.0.10         fields_14.1
## [85] vctrs_0.5.2         png_0.1-7           tidyselect_1.2.0
## [88] xfun_0.30
```