Tutorial for merging satellite-based precipitation datasets with ground observations using `RFmerge`

Oscar M. Baez-Villanueva[*] Mauricio Zambrano-Bigiarini[†]
Juan Diego Giraldo-Osorio Ian McNamara

07 January 2020

1 About

This vignette describes a basic application of the `RFmerge` function to create an improved precipitation dataset by combining two different satellite-based precipitation products with ground-based observations and user-selected covariates (i.e., digital elevation model and Euclidean distances to rain gauge stations).

We use as case study Valparaiso (Chile), as an example of how to generate this product at daily temporal scale and at 0.05° spatial resolution, from January to August 1983. This example requires the following data from the user:

i) time series of rainfall observations,
ii) metadata describinf the spatial coordiantes of the rain gauges,
iii) the Climate Hazards Group InfraRed Precipitation with Station data version 2.0 (CHIRPSv2),
iv) the Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks - Climate Data Record (PERSIANN-CDR), and
v) the Shuttle Radar Topography Mission (SRTM-v4) digital elevation model (DEM).

In addition, Euclidean distances are also used as covariates, but they are automatically computed within `RFmerge` as the Eucalidean distances from each rain gauge station to every grid-cell within the study area.

2 Installation

Install the latest stable version (from CRAN):

```
install.packages("RFmerge")
```

Alternatively, you can also try the under-development version from Github:

```
if (!require(devtools)) install.packages("devtools")
library(devtools)
install_github("hzambran/RFmerge")
```

[*]obaezvil@th-koeln.de
[†]mauricio.zambrano@ufrontera.cl

3 Setting up the environment

1. Load other packages that will be used in this analysis:

```
library(zoo)
library(sf)
library(rgdal)
library(raster)
```

2. Load the `RFmerge` package, which contains the main function used in the analysis and required datasets:

```
library(RFmerge)
```

4 Loading input data

First, daily time series of rainfall observations in 34 rain gauges located in Valparaiso will be used for this example, from *1983-01-01* to *1983-08-31*, which are available in the **ValparaisoPPts** dataset provided in the `RFmerge` package (for your own application, this dataset might be read from a CSV file or a `zoo` file). In addition, the **ValparaisoPPgis** dataset contains information about the IDs and spatial coordinates of each one of the rain gauges (for your own application, this dataset might be read from a CSV file). Finally, **ValparaisoSHP** stores the `sf` polygon defining the outer borders of the study area (for your own application, this dataset might be read from an ESRI shapefile).

```
data(ValparaisoPPts)
data(ValparaisoPPgis)
data(ValparaisoSHP)
```

Secondly, we need to load the satellit-based precipitation datasets and other covariates. For this example, CHIRPSv2 (Funk et al., 2015) and PERSIANN-CDR (Ashouri et al.,2015), at a spatial resolution of 0.05°, are used as dynamic covariates (in this context, *dynamic* means time-varying covariates). The Digital Elevation Model SRTM-v4 (DEM), also with a spatial resolution of 0.05°, is used as a static covariate to account for the impact of elevation on precipitation(in this context, *static* means that this covariate does not change in time).

```
chirps.fname   <- system.file("extdata/CHIRPS5km.tif",package="RFmerge")
prsnncdr.fname <- system.file("extdata/PERSIANNcdr5km.tif",package="RFmerge")
dem.fname      <- system.file("extdata/ValparaisoDEM5km.tif",package="RFmerge")

CHIRPS5km        <- brick(chirps.fname)
PERSIANNcdr5km   <- brick(prsnncdr.fname)
ValparaisoDEM5km <- raster(dem.fname)
```

5 Basic exploratory data analysis

The two multi-band geotiff files provided in the `RFmerge` package do not store the date of the precipitation estimate as name of the corresponding layer. Therefore, before any exploratory analysis, we would give meaningful names to each band (layer) in `CHIRPS5km` and `PERSIANNcdr5km`:

```
#ldates                  <- hydroTSM::dip("1983-01-01", "1983-08-31")
ldates                 <- seq(from=as.Date("1983-01-01"), to=as.Date("1983-08-31"), by="days")
names(CHIRPS5km)       <- ldates
names(PERSIANNcdr5km) <- ldates
```

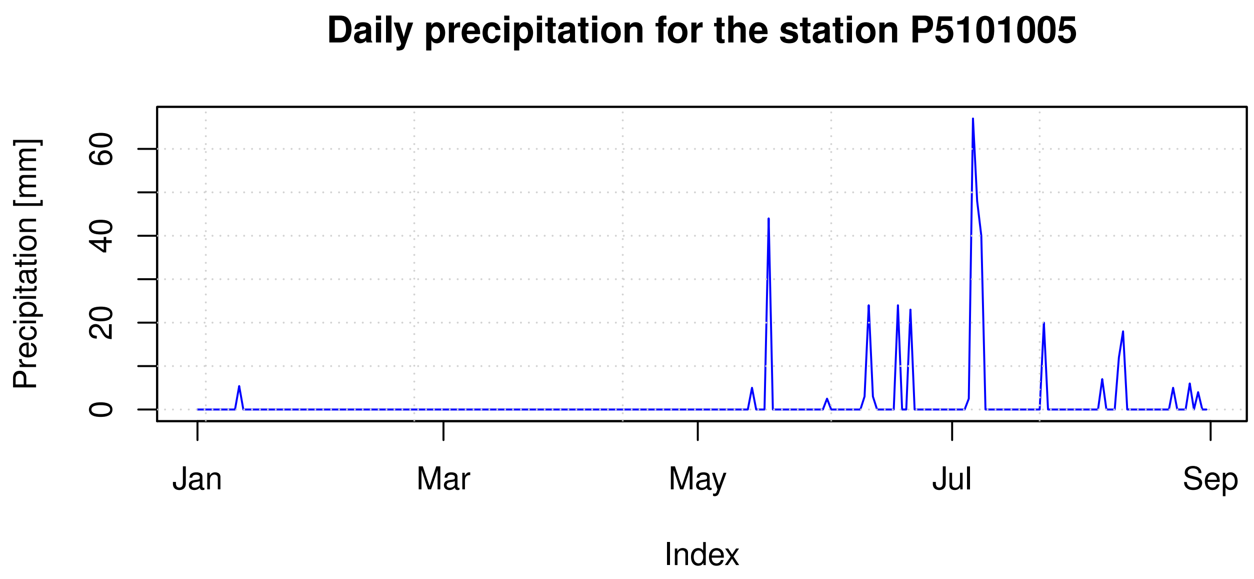
Then, we want to visualise the first six rows of the spatial metadata:

```
head(ValparaisoPPgis)
```

```
##       Code      lon      lat            NOM_REG  NOM_PROV COD_COM     NOM_COM
## 1 P5101005 -70.8000 -32.0836 Regin de Valparaso Los Andes    5304 San Esteban
## 2 P5111002 -71.0300 -32.1567 Regin de Valparaso Los Andes    5304 San Esteban
## 3 P5101006 -70.7833 -32.1836 Regin de Valparaso Los Andes    5304 San Esteban
## 4 P5100006 -70.7839 -32.2261 Regin de Valparaso Los Andes    5304 San Esteban
## 5 P5100005 -70.7100 -32.2286 Regin de Valparaso Los Andes    5304 San Esteban
## 6 P5111001 -71.1386 -32.2528 Regin de Valparaso Los Andes    5304 San Esteban
```

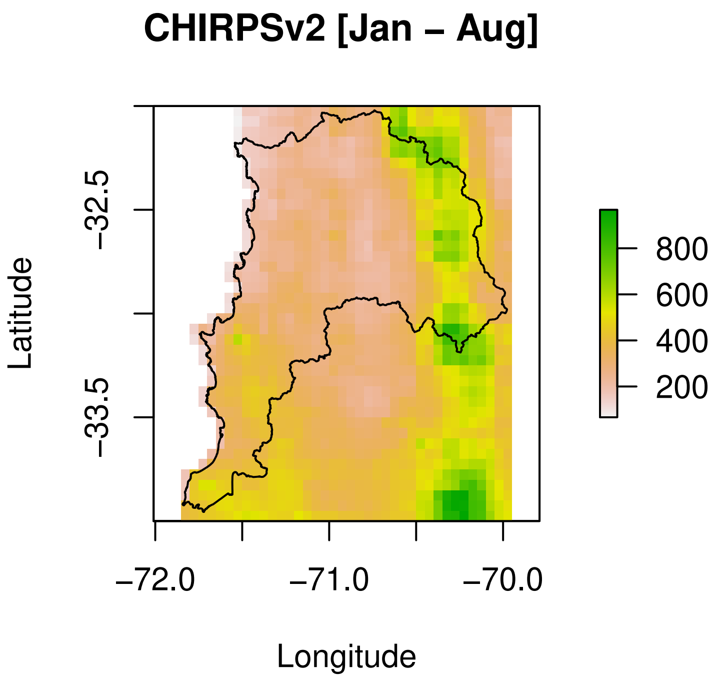
Plotting the daily precipitation time series for the first station (code: *P5101005*).

```
main <- paste("Daily precipitation for the station", ValparaisoPPgis$Code[1])
ylab <- "Precipitation [mm]"
x.ts <- ValparaisoPPts[,1]

#hydroTSM::hydroplot(x.ts, pfreq="o", main=main, ylab= ylab)
plot(x.ts, main=main, ylab= ylab, col="blue")
grid()
```

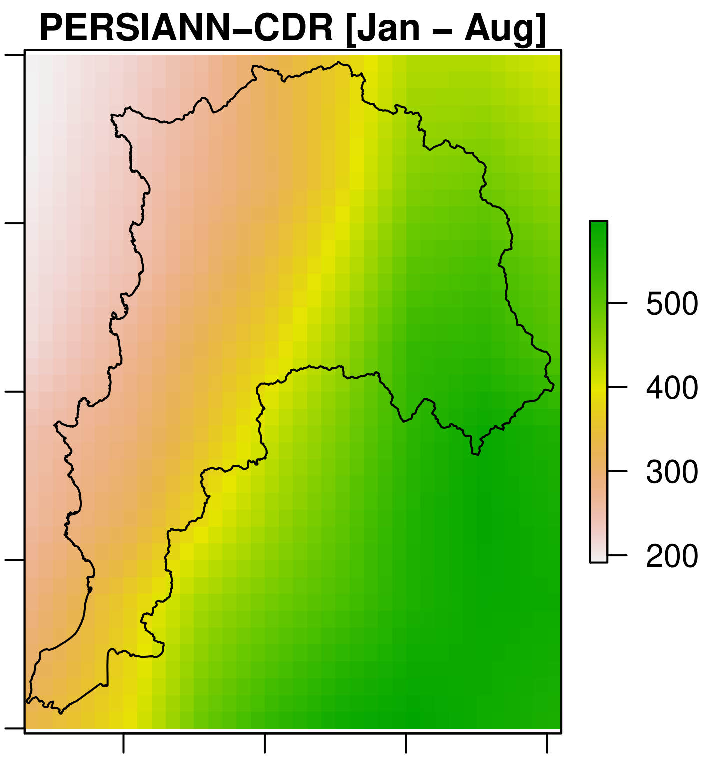


Plotting the accumulated precipitation estimates for the first eight months of 1983 from CHIRPS and PERSIANN-CDR, and overlying the boundaries of the study area (only its first attribute):

```
chirps.total   <- sum(CHIRPS5km, na.rm= FALSE)
persiann.total <- sum(PERSIANNcdr5km, na.rm= FALSE)


plot(chirps.total, main = "CHIRPSv2 [Jan - Aug] ", xlab = "Longitude", ylab = "Latitude")
plot(ValparaisoSHP[1], add=TRUE, col="transparent")
```



```r
plot(persiann.total, main = "PERSIANN-CDR [Jan - Aug]", xlab = "Longitude", ylab = "Latitude")
plot(ValparaisoSHP[1], add=TRUE, col="transparent")
```



6 Preparing input data

6.1 Spatial metadata

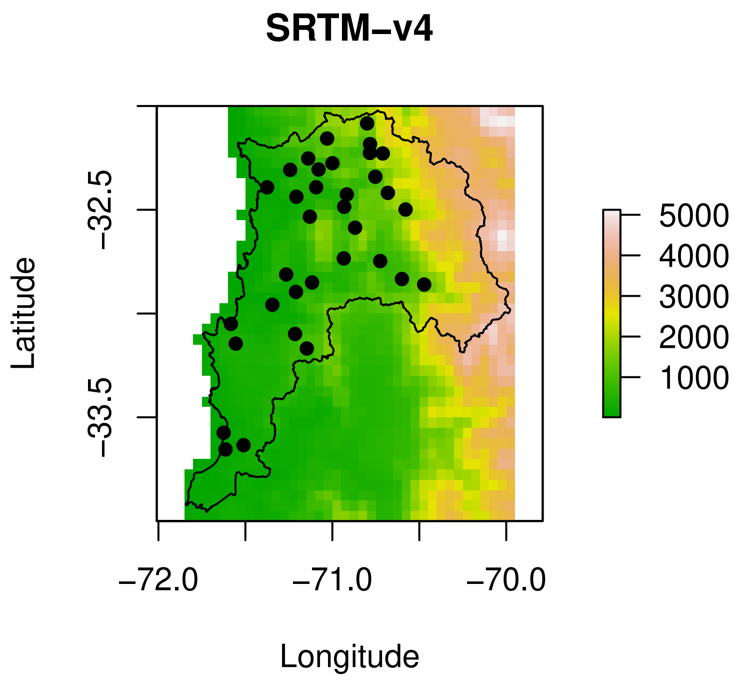
In order to use the spatial information stored in `ValparaisoPPgis`, we first need to convert it into a `SpatialPointsDataFrame`, using the latitude and longitude fields, stored in the `lat` and `lon` columns:

```
stations <- ValparaisoPPgis
( stations <- st_as_sf(stations, coords = c('lon', 'lat'), crs = 4326) )
```

```
## Simple feature collection with 34 features and 5 fields
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: -71.625 ymin: -33.655 xmax: -70.4719 ymax: -32.0836
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## First 10 features:
##        Code            NOM_REG  NOM_PROV COD_COM     NOM_COM
## 1  P5101005 Regin de Valparaso Los Andes    5304 San Esteban
## 2  P5111002 Regin de Valparaso Los Andes    5304 San Esteban
## 3  P5101006 Regin de Valparaso Los Andes    5304 San Esteban
## 4  P5100006 Regin de Valparaso Los Andes    5304 San Esteban
## 5  P5100005 Regin de Valparaso Los Andes    5304 San Esteban
## 6  P5111001 Regin de Valparaso Los Andes    5304 San Esteban
## 7  P5110003 Regin de Valparaso Los Andes    5304 San Esteban
## 8  P5111004 Regin de Valparaso Los Andes    5304 San Esteban
## 9  P5120004 Regin de Valparaso Los Andes    5304 San Esteban
## 10 P5200006 Regin de Valparaso Los Andes    5304 San Esteban
##                       geometry
## 1     POINT (-70.8 -32.0836)
## 2    POINT (-71.03 -32.1567)
## 3  POINT (-70.7833 -32.1836)
## 4  POINT (-70.7839 -32.2261)
## 5    POINT (-70.71 -32.2286)
## 6  POINT (-71.1386 -32.2528)
## 7  POINT (-70.9997 -32.2756)
## 8  POINT (-71.0792 -32.3058)
## 9  POINT (-71.2425 -32.3075)
## 10 POINT (-70.7528 -32.3408)
```

Now, we can plot the DEM with the locations of the rain gauge stations that will be used in the computation of RF-MEP, and overlying the boundaries of the study area and the stations (only their first attribute):

```
plot(ValparaisoDEM5km, main="SRTM-v4", xlab="Longitude", ylab="Latitude", col=terrain.colors(255))
plot(ValparaisoSHP[1], add=TRUE, col="transparent")
plot(stations[1], add=TRUE, pch = 16, col="black")
```



6.2 Verification of covariates

Note that for this example we want to produce a merged precipitation product only from January 1st to August 31th in 1983, i.e., 243 days; therefore, the precipitation products used as covariates have 243 layers each (one for each day in the time period used ofr the analysis), which is the **same number of rows** of the `ValparaisoPPts` object.

```
nlayers(CHIRPS5km)
```

```
## [1] 243
```

```
( nlayers(CHIRPS5km) == nlayers(PERSIANNcdr5km) )
```

```
## [1] TRUE
```

```
( nlayers(CHIRPS5km) == nrow(ValparaisoPPts) )
```

```
## [1] TRUE
```

Now, we have to verify that the precipitation products and the DEM have the **same spatial extent**:

```
extent(CHIRPS5km)
```

```
## class       : Extent
## xmin        : -71.85
## xmax        : -69.95
## ymin        : -34
## ymax        : -32
```

```
( extent(CHIRPS5km) == extent(PERSIANNcdr5km) )
```

```
## [1] TRUE
```

```
( extent(CHIRPS5km) == extent(ValparaisoDEM5km) )
```

```
## [1] TRUE
```

and the **same spatial resolution**:

```
res(CHIRPS5km)
```

```
## [1] 0.05 0.05
```

```
( res(CHIRPS5km) == res(PERSIANNcdr5km) )
```

```
## [1] TRUE TRUE
```

```
( res(CHIRPS5km) == res(ValparaisoDEM5km) )
```

```
## [1] TRUE TRUE
```

6.2.1 Optional: Reprojection into another CRS

If you would like to test the use of Euclidean distances as covariates in `RFmerge` you need to be sure that Euclidean distances will be correctly computed from your datasets.

Because `ValparaisoPPgis`, `CHIRPS5km`, and `PERSIANNcdr5km` all use geographical coordinates, we need first to project their values into a projected coordinate reference system (CRS), i.e., one defined on a flat, two-dimensional surface.

First, we reproject the satellite products from geographic coordinates into WGS 84 / UTM zone 19S (EPSG:32719):

```
utmz19s.p4s <- CRS("+init=epsg:32719") # WGS 84 / UTM zone 19S

CHIRPS5km.utm        <- projectRaster(from=CHIRPS5km, crs=utmz19s.p4s)
PERSIANNcdr5km.utm   <- projectRaster(from=PERSIANNcdr5km, crs=utmz19s.p4s)
ValparaisoDEM5km.utm <- projectRaster(from=ValparaisoDEM5km, crs=utmz19s.p4s)
```

Second, we reproject the rainfall observations from geographic coordinates into WGS 84 / UTM zone 19S (EPSG:32719):

```
stations.utm <- sf::st_transform(stations, crs=32719) # for 'sf' objects
```

Third, we reproject the polygon with the boundaries used to define the study area from geographic coordinates into WGS 84 / UTM zone 19S (EPSG:32719):

```
ValparaisoSHP.utm <- sf::st_transform(ValparaisoSHP, crs=32719)
```

Fourth, we create a new data.frame with the expected metadata, i.e., at least, ID, lat, lon:

```
st.coords <- st_coordinates(stations.utm)
lon       <- st.coords[, "X"]
lat       <- st.coords[, "Y"]

ValparaisoPPgis.utm <- data.frame(ID=stations.utm[["Code"]], lon=lon, lat=lat)
```

You might want to skip (**at your own risk**) this reprojection step when the study area is small enough to neglect the impact of using geographic coordinates in the computation of Euclidean distances.

7 Running RFmerge

7.1 Covariates

Now, we can create the covariates object to be used in the RFmerge function. For doing this, we will create a list object with the different covariates. Please note that **the order and name of the covariates in the list is not important**.

```
covariates.utm <- list(chirps=CHIRPS5km.utm, persianncdr=PERSIANNcdr5km.utm,
                       dem=ValparaisoDEM5km.utm)
```

7.2 Setup

Finally, if you want the resulting merged files be written into disk you need to define the output directory (drty.out) before running RFmerge. Then, you can run the RFmerge function as follows:

Without using parallelisation (default option):

```
drty.out <- file.path(tempdir(), "Test.nop")
rfmep <- RFmerge(x=ValparaisoPPts, metadata=ValparaisoPPgis.utm, cov=covariates.utm,
                 id="ID", lat="lat", lon="lon", mask=ValparaisoSHP.utm,
                 training=0.8, write2disk=TRUE, drty.out=drty.out)
```

Detecting if your OS is Windows or GNU/Linux, and setting the 'parallel' argument accordingly:

```
onWin <- ( (R.version$os=="mingw32") | (R.version$os=="mingw64") )
ifelse(onWin, parallel <- "parallelWin", parallel <- "parallel")
```

```
## [1] "parallel"
```

Using parallelisationin GNU/Linux machines, with a maximum number of nodes/cores to be used equal to 2:

```
par.nnodes <- min(parallel::detectCores()-1, 2)
drty.out <- file.path(tempdir(), "Test.par")
rfmep <- RFmerge(x=ValparaisoPPts, metadata=ValparaisoPPgis.utm, cov=covariates.utm,
                 id="ID", lat="lat", lon="lon", mask=ValparaisoSHP.utm,
                 training=0.8, write2disk=TRUE, drty.out=drty.out,
                 parallel=parallel, par.nnodes=par.nnodes)
```

```
## [ Creating the training (80%) and evaluation (20%) datasets ... ]

## [ Computing the Euclidean distances to each observation of the training set ...]

##

## [ Parallel initialization ... ]

## [ Number of cores/nodes detected: 4 ]

## [ Number of cores/nodes used    : 2 ]

## [ Parallelisation finished ! ]
```

Using parallelisation in Window$ machines, with a maximum number of nodes/cores to be used equal to 2:

```
par.nnodes <- min(parallel::detectCores()-1, 2)
drty.out <- file.path(tempdir(), "Test.par")
rfmep <- RFmerge(x=ValparaisoPPts, metadata=ValparaisoPPgis.utm, cov=covariates.utm,
                 id="Code", lat="lat", lon="lon", mask=ValparaisoSHP.utm,
                 training=0.8, write2disk=TRUE, drty.out=drty.out,
                 parallel=parallelWin, par.nnodes=par.nnodes)
```

7.3 Expected outputs

If `RFmerge` run without problems, and `write2disk=TRUE`, the following output files will be stored in your user-defined `drty.out` directory:

i) the rain gauge stations used as training and evaluation datasets; and

ii) the final merged product (individual *GeoTiff* files).

The aforementioned resulting objects will be stored within `drty.out` as follows:

- `Ground_based_data/Training/`: In this directory, time series and metadata used to train RF-MEP will be stored as a `zoo` file (`Training_ts.txt`), and a `text` file (`Training_metadata.txt`), respectively.
- `Ground_based_data/Evaluation/`: This directory will store time series and metadata not used in training the Random Forest model, but that are available for evaluating the performance of the results in an independent evaluation dataset. The `Evaluation_ts.txt` and `Evaluation_metadata.tx` files are stored as `zoo` and `CSV` files, respectively)
- `RF-MEP/`: This directory will store the individual *GeoTiff* files produced by the RF-MEP algorithmn, using the same spatial resolution as the selected covariates.

7.4 Evaluation

After running `RFmerge` we will use the evaluation dataset of rain gauge observations to evaluate the performance of RF-MEP at observation not used in the merging procedure. For this purpose, we will create a stack with the obtained merged product and will import the time series and metadata from the evaluation set:

```
ts.path       <- paste0(drty.out, "/Ground_based_data/Evaluation/Evaluation_ts.txt")
metadata.path <- paste0(drty.out, "/Ground_based_data/Evaluation/Evaluation_metadata.txt")

eval.ts  <- read.zoo(ts.path, header = TRUE)
eval.gis <- read.csv(metadata.path)

# promoting 'eval.gis' into a spatial object (in order to be plotted)
( eval.gis.utm <- st_as_sf(eval.gis, coords = c('lon', 'lat'), crs = 32719) )
```

Visualisation of one day of precipitation using RF-MEP, and overlying the boundaries of the study area (only its first attribute)::

```
plot(rfmep[[11]], main="RF-MEP precipitation for 1983-01-11", xlab="Longitude", ylab="Latitude")
plot(ValparaisoSHP.utm[1], add=TRUE, col="transparent")
plot(eval.gis.utm, add=TRUE, pch = 16, col="black")
```

The total amount of precipitation over Valparaiso for January to August 1983 according to RF-MEP, and overlying the boundaries of the study area (only its first attribute)::

```
rfmep.total <- sum(rfmep, na.rm= FALSE)

plot(rfmep.total, main = "RF-MEP [Jan - Aug]", xlab = "Longitude", ylab = "Latitude")
plot(ValparaisoSHP.utm[1], add=TRUE, col="transparent")
```

First, we will compare RF-MEP (and the two precipitation products used as covariates) with the rain gauge data from the evaluation set. To extract the RF-MEP precipitation values at the gauge locations, we will use the `extract` function from the `raster` package:

```r
coordinates(eval.gis) <- ~ lon + lat

rfmep.ts    <- t(raster::extract(rfmep, eval.gis))
chirps.ts   <- t(raster::extract(CHIRPS5km.utm, eval.gis))
persiann.ts <- t(raster::extract(PERSIANNcdr5km.utm, eval.gis))
```

To evaluate the performance of RF-MEP and the products used in its computation, the NAsh-Sutcliffe efficiency (NSE) will be used. The optimal value for the NSE is one.

```r
# Defining a function to compute the Nash-Sutcliffe efficiency(NSE)
NSE <- function(sim, obs) return( 1 - (sum((obs - sim)^2)/ (sum((obs - mean(obs))^2)) ) )

sres        <- list(chirps.ts, persiann.ts, rfmep.ts)
nsres       <- length(sres)
nstations   <- ncol(eval.ts)
tmp         <- rep(NA, nstations)
nse.table   <- data.frame(ID=eval.gis[["ID"]], CHIRPS=tmp, PERSIANN_CDR=tmp, RF_MEP=tmp)

# Computing the NSE between the observed rainfall measured in each one of the raingauges
# of the training dataset and CHIRPSv2, PERSIANN-CDR, the merged product `rfmep`:

for (i in 1:nsres) {
  ldates <- time(eval.ts)
  lsim   <- zoo(sres[[i]], ldates)
  nse.table[, (i+1)] = NSE(sim= lsim, obs= eval.ts)
} # FOR end
```

Finally, a boxplot comparing the performance, in terms of *KGE'*, of the merged product in comparison to the original SREs used as covariates will be produced:

```r
# Boxplot with a graphical comparison
sres.cols <- c("powderblue", "palegoldenrod", "mediumseagreen")
boxplot(nse.table[,2:4], main = "NSE evaluation for Jan - Aug 1983",
        xlab = "P products", ylab = "NSE'", ylim = c(0, 1), # horizontal=TRUE,
        col=sres.cols)
legend("topleft", legend=c("CHIRPS", "PERSIANN-CDR", "RF-MEP"), col=sres.cols, pch=15, cex=1.5, bty="n"
grid()
```

8 Full vignette

In order to reduce the package dependencies for CRAN, this vignette was built with reduced functionality. The full vignette can be found here.

9 Software details

This tutorial was built under:

```
## [1] "x86_64-pc-linux-gnu (64-bit)"
```

```
## [1] "R version 3.6.2 (2019-12-12)"
## [1] "RFmerge 0.1-6"
```

10 References

1. Ashouri, H., Hsu, K.-L., Sorooshian, S., Braithwaite, D. K., Knapp, K. R., Cecil, L. D., Nelson, B. R., and Prat, O. P. (2015). PERSIANN-CDR: Daily Precipitation Climate Data Record from Multisatellite Observations for Hydrological and Climate Studies, Bulletin of the American Meteorological Society, 96, 69–83, doi:10.1175/BAMS-D-13-00068.1.

2. Baez-Villanueva, O. M.; Zambrano-Bigiarini, M.; Beck, H.; McNamara, I.; Ribbe, L.; Nauditt, A.; Birkel, C.; Verbist, K.; Giraldo-Osorio, J.D.; Thinh, N.X. (2020). RF-MEP: a novel Random Forest method for merging gridded precipitation products and ground-based measurements, Remote Sensing of Environment, 239, 111610. doi:10.1016/j.rse.2019.111606.

3. Funk, C., Peterson, P., Landsfeld, M., Pedreros, D., Verdin, J., Shukla, S., Husak, G., Rowland, J., Harrison, L., Hoell, A., and Michaelsen, J. (2015) The climate hazards infrared precipitation with stations-a new environmental record for monitoring extremes, Sci Data, 2, 150 066, doi:10.1038/sdata.2015.66.

4. Hengl, T., Nussbaum, M., Wright, M. N., Heuvelink, G. B., & Gr"{a}ler, B. (2018). Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. PeerJ, 6, e5518.