# hyfo Easy Start

*Yuanchao Xu*

*2015-07-01*

## hyfo

**is designed for hydrology and forecasting anaylasis, containing a number of tools including data extration, data processing and data visulization. There are two main parts in the package, as well as in this mannual:**

1. Hydrology

- Providing tools from raw data extration to final precipitation data used by model.
- e.g., data extraction from file, precipitation gap filler, annual precipitation calculation.

2. Forecasting

- Providing tools from forecasting data visulization and analysis.
- e.g., get spatial maps, data analysis and bias analysis.

### Note

- For the forecasting tools part, `hyfo` mainly focuses on the post processing of the gridData derived from forecasts or other sources. The input is a list file, usually a result from `loadGridData{ecomsUDG.Raccess}` or `loadECOMS{ecomsUDG.Raccess}`. Pacakage`{ecomsUDG.Raccess}`is designed for getting access to different dataset, and also can load grid file (like netcdf file) directly.
- The functions end with `_anarbe` are the functions designed specially for some case in Spain, those functions mostly are about data collection of the anarbe catchment, which will be introduced in the end fo this mannual.

## Content

- 2.2 Add Background (catchment and gauging stations)
  - 2.2.1 Add catchment shape file.
  - 2.2.2 Add station locations
- 2.3 Precipitation Bar Plot
- 2.4 Analysis and Comparison
  - 2.4.1 Spatial Map
  - 2.4.2 Bar Plot

3. Anarbe Case

# 1. Hydrology

**Note** If you are an experienced R user, and know how to read data in R, deal with dataframe, generate date and list, please start from next charpter, "1.2 Rainfall Analysis"

## 1.1 Data Preparation

### 1.1.1 From File

`hyfo` does provide a common tool for collecting data from different type of files, including "txt",

"csv" and "excel", which has to be assigned to the argument `fileType`.

Now let's use internal data as an example.

```
library(hyfo)#load the package.
# get the folder containing different csv (or other type) files.
file <- system.file("extdata", "1999.csv", package = "hyfo")
folder <- strsplit(file, '1999')[[1]][1]

# Extract and combine content from different files and in each file, the extracted zone is
# from row 10 to row 20, Column 1 to column2.
a <- collectData(folder, fileType = 'csv', range = c(10, 20, 1, 2))
```

```
## All the files in the folder should have the same format
```

```
str(a)
```

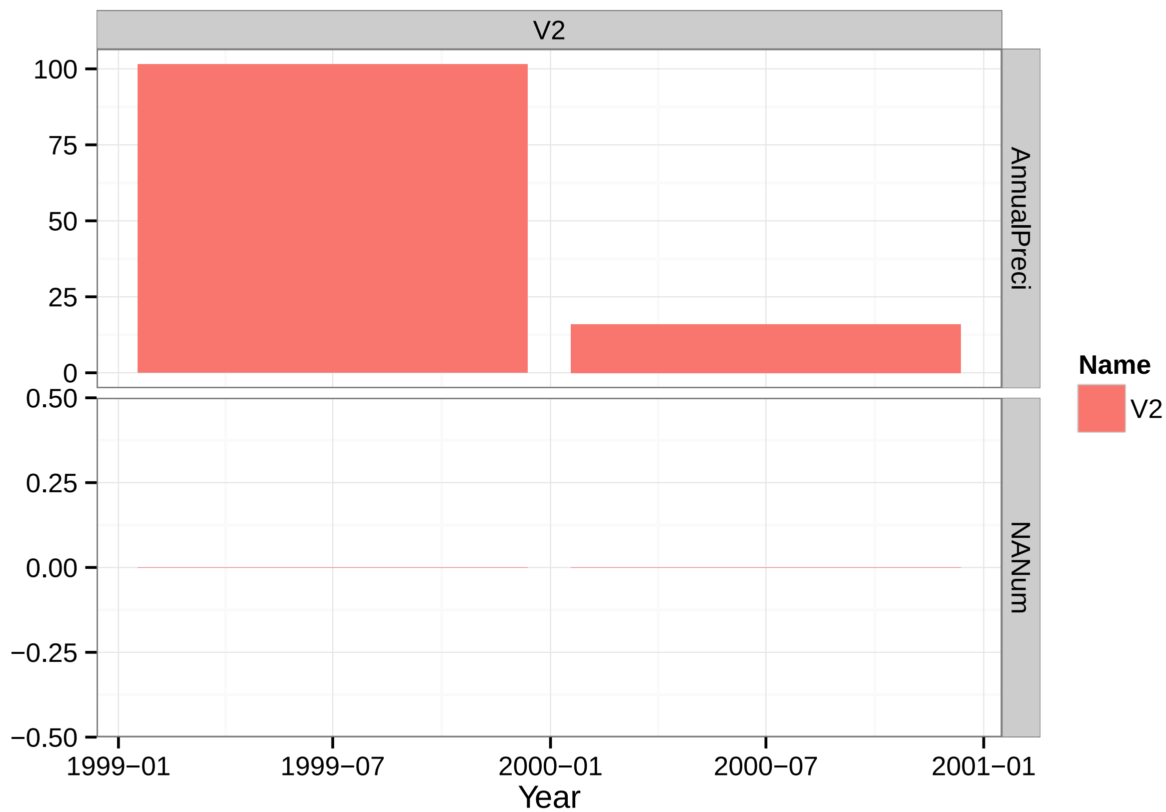
```
## 'data.frame':    22 obs. of  2 variables:
##  $ V1: Factor w/ 722 levels "","01/02/1999",..: 57 69 81 93 105 117 129 141 153 165 ...
##  $ V2: num  0 0 19.7 42.9 4.7 14.5 2 10.9 5.6 0 ...
```

`a` cannot be directly inputed in `hyfo`, it still needs some process.

```
# Check the date to see if it follows the format in ?as.Date(), if not,
# use as.Date to convert.
a <- data.frame(a)
#get date
date <- a[, 1]
```

```r
# The original format is d/m/year, convert to formal format.
date <- as.Date(date, format = '%d/%m/%Y')
a[, 1] <- date

# Now a has become `a` time series dataframe, which is the atom element of the analysis.
#`hyfo` deals with list containing different time series dataframe. In this example,
#there is only one dataframe, and more examples please refer to the following chapter.
datalist <- list(a)

# Use getAnnual as an example, here since `a` is not a complete time series,
# the result is only base on the input.
# getAnnual gives the annual precipitation of each year,
# and will be introduced in the next chapter.
getAnnual(datalist)
```

```
## Using Year, Name as id variables
```



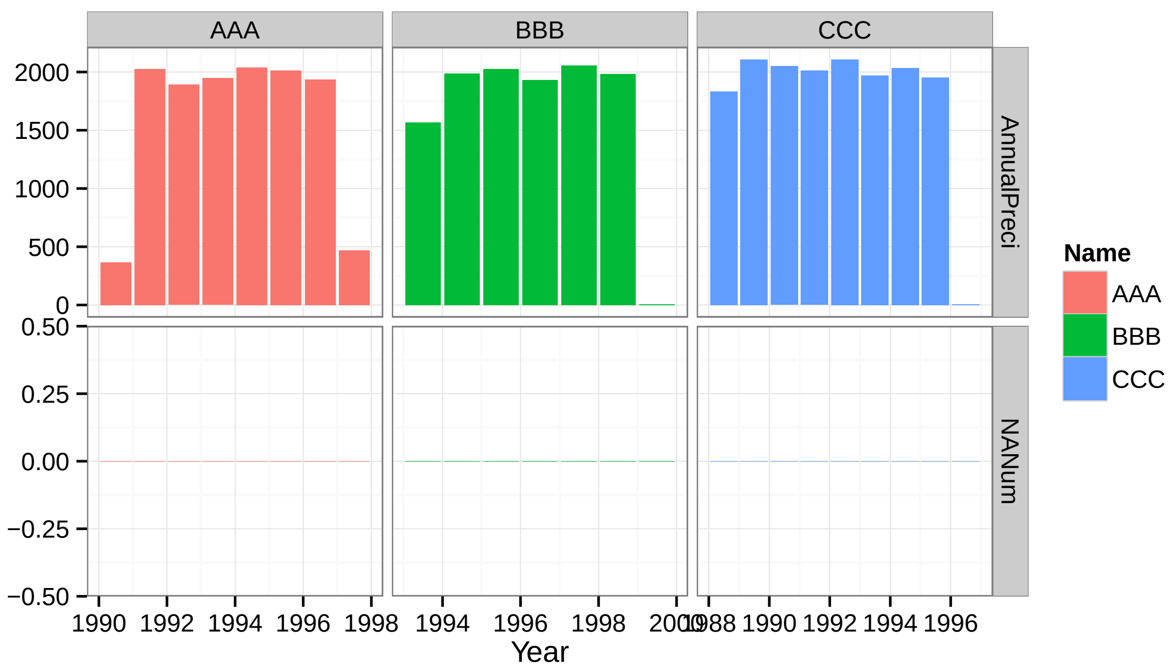

```
##   Year Name AnnualPreci recordNum NANum
## 1 1999   V2       101.5        11     0
## 2 2000   V2        16.0        11     0
```

### 1.1.2 Mannually

Following example shows a simple way to generate dataframe with start date, end date, and the value. Here in the example, `sample()` is used to generate random values, while in real case it will be a vector containing time series values.

```
# Generate timeseries datalist. Each data frame consists of a Date and a value.
library(hyfo)
AAA <- data.frame(
  Date = seq(as.Date('1990-10-28'), as.Date('1997-4-1'), 1), # Date column
  AAA = sample(1:10, length(seq(as.Date('1990-10-28'), # value column
                                as.Date('1997-4-1'), 1)), repl = TRUE))

BBB <- data.frame(
  Date = seq(as.Date('1993-3-28'), as.Date('1999-1-1'),1),
  BBB = sample(1:10, length(seq(as.Date('1993-3-28'),
                                as.Date('1999-1-1'),1)), repl = TRUE))

CCC <- data.frame(
  Date = seq(as.Date('1988-2-2'), as.Date('1996-1-1'),1),
  CCC = sample(1:10, length(seq(as.Date('1988-2-2'),
                                as.Date('1996-1-1'),1)), repl = TRUE))

datalist <- list(AAA, BBB, CCC)# dput() and dget() can be used to save and load list file.
a <- getAnnual(datalist)
```

```
## Using Year, Name as id variables
```

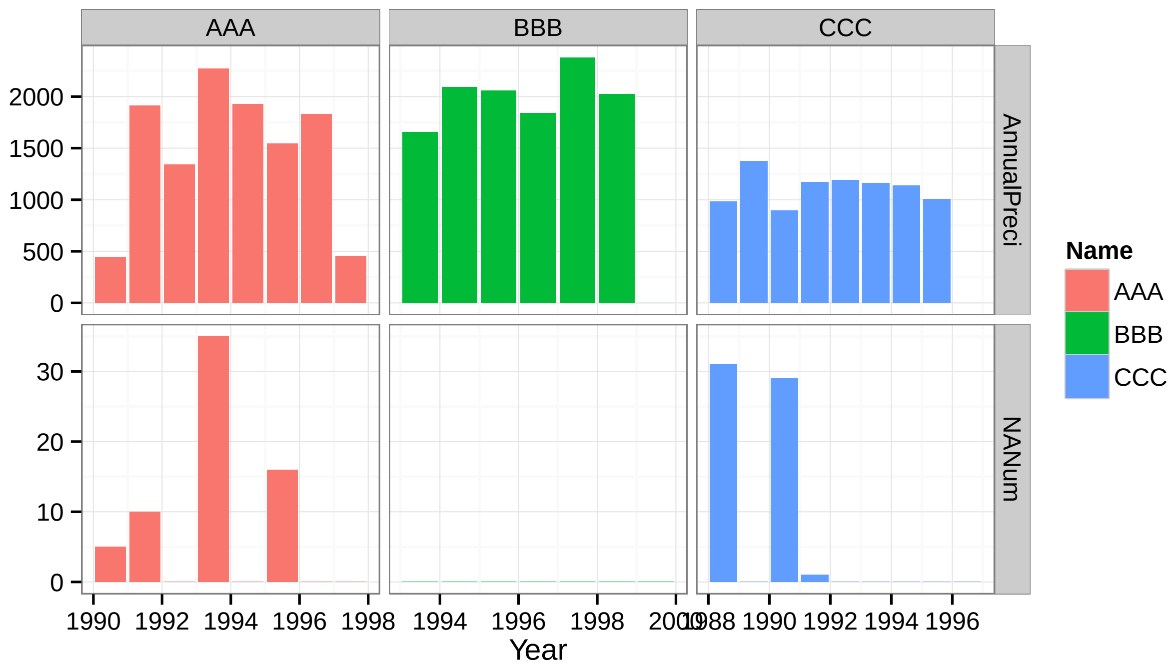


## 1.2 Rainfall Analysis

Assuming we have three gauging stations named “AAA”, “BBB”, “CCC”, the precipitation information can be get by the following:

```
# testdl is a datalist provided by the package as a test.
# It's a list containing different time series.
data(testdl)
a <- getAnnual(testdl)
```

```
## Using Year, Name as id variables
```

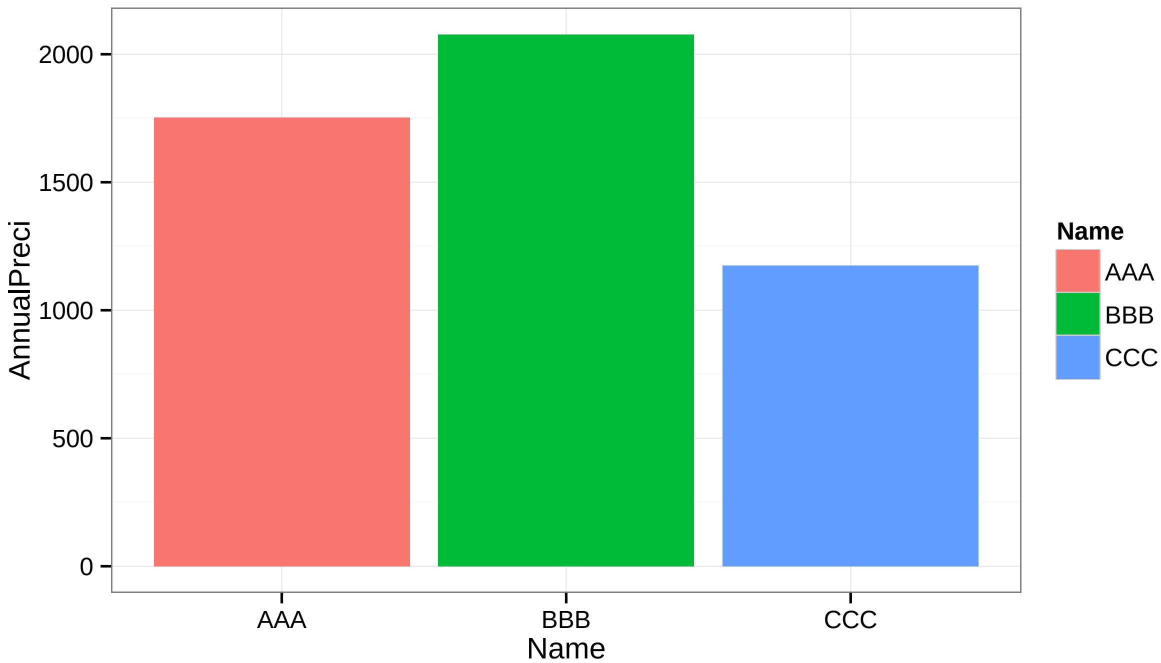



As shown above, the annual precipitation and the number of missing values are shown in the figure. Knowing how many missing values you have is alway important when calculating the mean annual precipitation.

Now we want to get the mean annual precipitation.

```
a <- getAnnual(testdl, output = 'mean')
a
```

```
##   Name AnnualPreci
## 1  AAA    1752.725
## 2  BBB    2078.190
## 3  CCC    1174.540
```



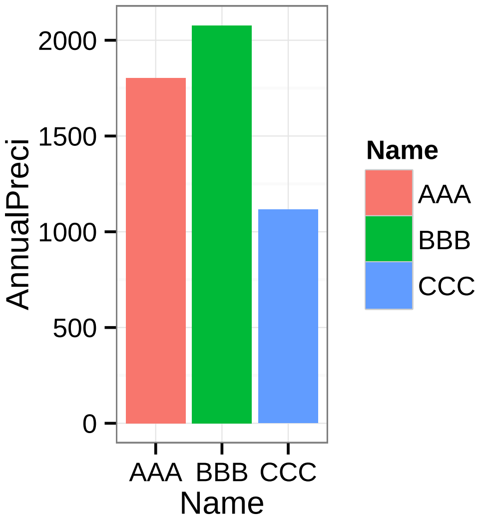

Mean annual precipitation is calculated, but as we can see in the figure before, it's not reliable, since there are a lot of missing values in AAA and CCC, especially in AAA, in 1993, there are more than 30 missing values in a year. So we have to decide which is the threshold for the valid record. the default is 355, which means in a year (355 or 365 days), if the valid records (not missing) exceeds 355, then this year is taken into consideration in the mean annual preicipitation calculation.

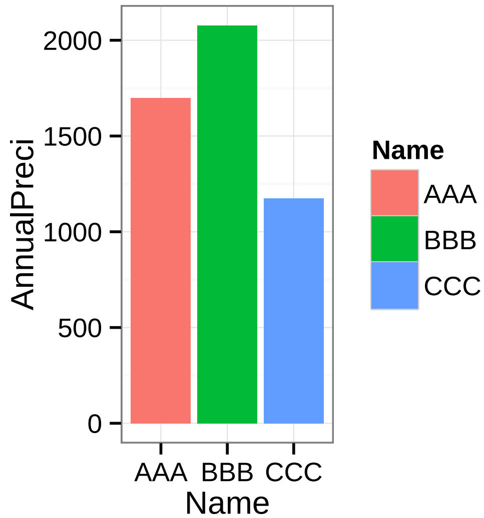
```
getAnnual(testdl, output = 'mean', minRecords = 300)
```

```
##   Name AnnualPreci
## 1  AAA    1804.154
## 2  BBB    2078.190
## 3  CCC    1115.910
```

```
getAnnual(testdl, output = 'mean', minRecords = 365)
```

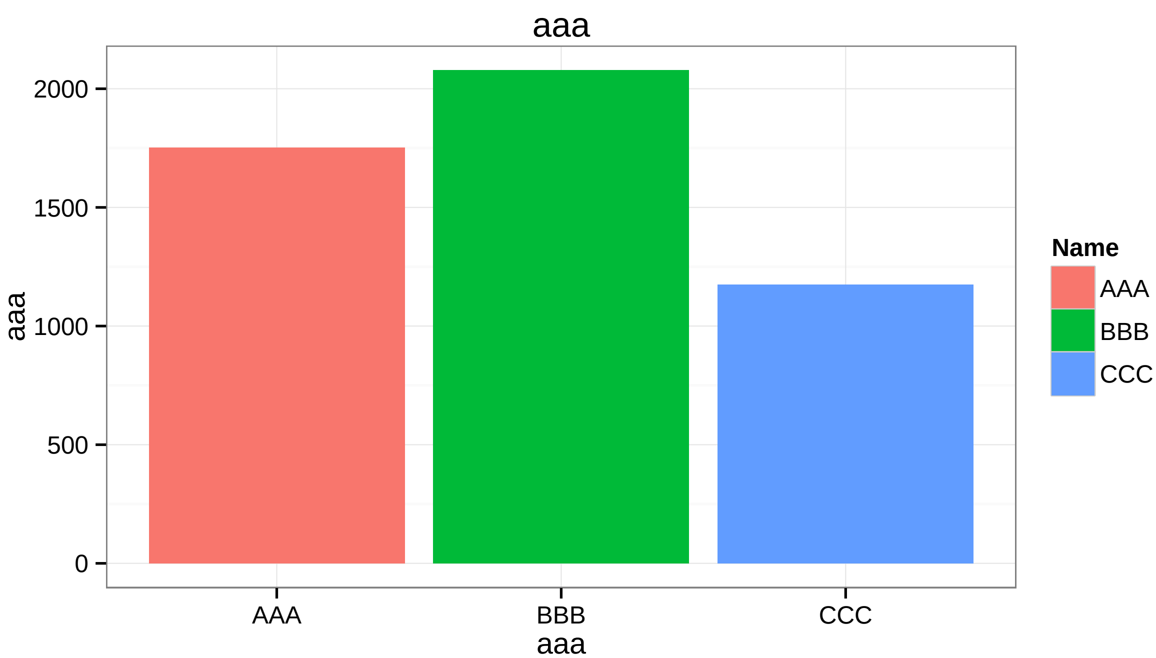
```
##   Name AnnualPreci
## 1  AAA    1699.079
## 2  BBB    2078.190
## 3  CCC    1175.172
```




If you are not satisfied with the title and x axis and y axis, you can assign them yourself.

```
a <- getAnnual(testdl, output = 'mean', title = 'aaa', x = 'aaa', y = 'aaa')
```




If you want to calculate annual rainfall for a single dataframe containing one time series.

```
a <- getAnnual_dataframe(testdl[[1]])
a
```

```
##      Year Name AnnualPreci recordNum NANum
## 1990 1990  AAA     446.772        60     5
## 1991 1991  AAA    1913.661       355    10
```

```
## 1992 1992  AAA    1340.688       366     0
## 1993 1993  AAA    2270.130       330    35
## 1994 1994  AAA    1927.704       365     0
## 1995 1995  AAA    1543.893       349    16
## 1996 1996  AAA    1828.845       366     0
## 1997 1997  AAA     454.863        91     0
```

### 1.3 Extract Common Period from Different Time Series

Now we have the general information of the precipitation, if we want to use them in a model, we have to extract the common period of them, and use the common period precipitation to analyze.

```
testdl_new <- extractPeriod(testdl, commonPeriod = TRUE )
str(testdl_new)
```

```
## List of 3
##  $ AAA:'data.frame': 1010 obs. of  2 variables:
##   ..$ Date: Date[1:1010], format: "1993-03-28" ...
##   ..$ AAA : num [1:1010] NA NA NA NA NA NA NA NA NA NA ...
##  $ BBB:'data.frame': 1010 obs. of  2 variables:
##   ..$ Date: Date[1:1010], format: "1993-03-28" ...
##   ..$ BBB : num [1:1010] 0 1.26 0 0 0 ...
##  $ CCC:'data.frame': 1010 obs. of  2 variables:
##   ..$ Date: Date[1:1010], format: "1993-03-28" ...
##   ..$ CCC : num [1:1010] 0.72 1.56 20.82 18.9 9.54 ...
```

If we want to extract data from a certain period, we can assgin start and end date.

```
# Extract period of the winter of 1994
testdl_new <- extractPeriod(testdl, startDate = '1994-12-01', endDate = '1995-03-01' )
str(testdl_new)
```

```
## List of 3
##  $ AAA:'data.frame': 91 obs. of  2 variables:
##   ..$ Date: Date[1:91], format: "1994-12-01" ...
##   ..$ AAA : num [1:91] 0.837 10.509 0.279 4.185 0 ...
##  $ BBB:'data.frame': 91 obs. of  2 variables:
##   ..$ Date: Date[1:91], format: "1994-12-01" ...
##   ..$ BBB : num [1:91] 0 0.45 0.45 0 0 0 0 0 0 0 ...
##  $ CCC:'data.frame': 91 obs. of  2 variables:
##   ..$ Date: Date[1:91], format: "1994-12-01" ...
##   ..$ CCC : num [1:91] 7.32 0 0 0 19.08 ...
```

### 1.4 Fill Gaps (rainfall data gaps)

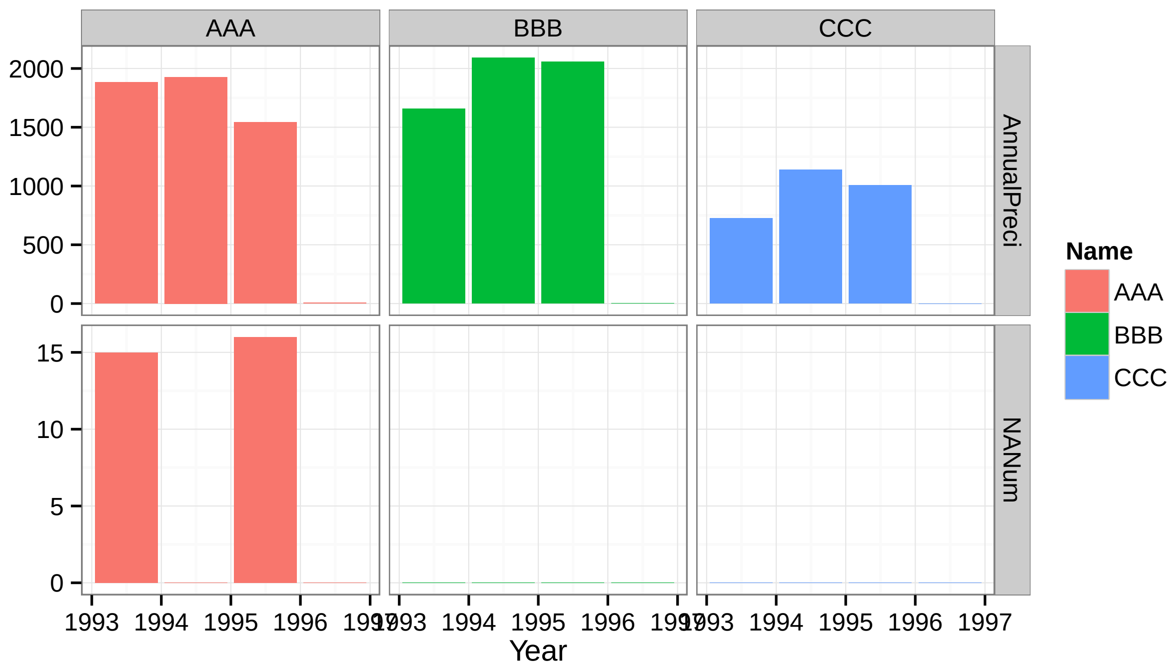
Although we have got the precipitation of the common period, we can still see that there are some missing values inside, which we should fill.

```
testdl_new <- extractPeriod(testdl, commonPeriod = TRUE )
a <- getAnnual(testdl_new)
```

```
## Using Year, Name as id variables
```

```
a
```

```
##    Year Name AnnualPreci recordNum NANum
## 1  1993  AAA    1883.157       264    15
## 2  1994  AAA    1927.704       365     0
## 3  1995  AAA    1543.893       349    16
## 4  1996  AAA       5.394         1     0
## 5  1993  BBB    1657.080       279     0
## 6  1994  BBB    2090.970       365     0
## 7  1995  BBB    2056.230       365     0
## 8  1996  BBB       3.060         1     0
## 9  1993  CCC     724.560       279     0
## 10 1994  CCC    1139.640       365     0
## 11 1995  CCC    1006.260       365     0
## 12 1996  CCC       0.000         1     0
```




First we have to transform the datalist to dataframe, which can be done by the code below:

```
df <- list2Dataframe(testdl_new)
head(df)
```

```
##         Date AAA  BBB   CCC
## 1 1993-03-28  NA 0.00  0.72
## 2 1993-03-29  NA 1.26  1.56
## 3 1993-03-30  NA 0.00 20.82
## 4 1993-03-31  NA 0.00 18.90
## 5 1993-04-01  NA 0.00  9.54
## 6 1993-04-02  NA 0.00  0.00
```

From above, we can see that in the gauging station “AAA”, there are some missing value marked as “NA”. Now we are going to fill these gaps.

The gap filling is based on the correlation and linear regression between each two gauging stations, correlation table, correlation Order and Linear Coefficients are also printed when doing the calculation. Details can be found in `?fillGap`.

```
df_filled <- fillGap(df)
```

```
##
## Correlation Coefficient
##              AAA         BBB         CCC
## AAA  1.000000000 -0.07445112 0.008566204
## BBB -0.074451120  1.00000000 0.039809765
## CCC  0.008566204  0.03980976 1.000000000
##
## Correlation Order
##     1     2
## AAA "CCC" "BBB"
## BBB "CCC" "AAA"
## CCC "BBB" "AAA"
##
## Linear Coefficients
##             1          2
## AAA 0.3308048 0.12015931
## BBB 0.3756172 0.11752878
## CCC 0.1094488 0.09047318
```

```
head(df_filled)
```

```
##         Date   AAA  BBB   CCC
## 1 1993-03-28 0.238 0.00  0.72
## 2 1993-03-29 0.516 1.26  1.56
## 3 1993-03-30 6.887 0.00 20.82
## 4 1993-03-31 6.252 0.00 18.90
## 5 1993-04-01 3.156 0.00  9.54
## 6 1993-04-02 0.000 0.00  0.00
```

Default correlation period is “daily”, while sometimes the daily rainfall correlation of precipitation is not so strong, we can also select the correlation period.

```
df_filled <- fillGap(df, corPeriod = 'monthly')
```

```
##
## Correlation Coefficient
##             AAA         BBB       CCC
## AAA  1.00000000 -0.02020277 0.4980004
## BBB -0.02020277  1.00000000 0.2513406
## CCC  0.49800040  0.25134059 1.0000000
##
## Correlation Order
##     1     2
```

```
## AAA "CCC" "BBB"
## BBB "CCC" "AAA"
## CCC "AAA" "BBB"
##
## Linear Coefficients
##              1         2
## AAA 0.33080477 0.1201593
## BBB 0.37561723 0.1175288
## CCC 0.09047318 0.1094488
```

```
head(df_filled)
```

```
##         Date   AAA  BBB   CCC
## 1 1993-03-28 0.238 0.00  0.72
## 2 1993-03-29 0.516 1.26  1.56
## 3 1993-03-30 6.887 0.00 20.82
## 4 1993-03-31 6.252 0.00 18.90
## 5 1993-04-01 3.156 0.00  9.54
## 6 1993-04-02 0.000 0.00  0.00
```

```
df_filled <- fillGap(df, corPeriod = 'yearly')
```

```
##
## Correlation Coefficient
##            AAA       BBB        CCC
## AAA 1.00000000 0.1894243 0.02040045
## BBB 0.18942426 1.0000000 0.97659734
## CCC 0.02040045 0.9765973 1.00000000
##
## Correlation Order
##     1     2
## AAA "BBB" "CCC"
## BBB "CCC" "AAA"
## CCC "BBB" "AAA"
##
## Linear Coefficients
##             1          2
## AAA 0.1201593 0.33080477
## BBB 0.3756172 0.11752878
## CCC 0.1094488 0.09047318
```

```
head(df_filled)
```

```
##         Date   AAA  BBB   CCC
## 1 1993-03-28 0.000 0.00  0.72
## 2 1993-03-29 0.151 1.26  1.56
## 3 1993-03-30 0.000 0.00 20.82
## 4 1993-03-31 0.000 0.00 18.90
## 5 1993-04-01 0.000 0.00  9.54
## 6 1993-04-02 0.000 0.00  0.00
```

## 1.5 Seasonal and Monthly Precipitation

Sometimes we need to know not only the annual precipitation, but also the precipitation of a certain month or certain season.

```
data(testdl)
# year and mon can be extracted from date.
TS  <- testdl[[1]]
year = as.numeric(format(TS[, 1], '%Y'))
month = as.numeric(format(TS[, 1], '%m'))
```

Get the mean spring precipitation.

```
a <- getMeanPreci(TS[, 2], method = 'spring', yearIndex = year, monthIndex = month)
a
```

```
## [1] 450.8082
```

Get the series of spring precipitation, set `fullResults = TRUE`.

```
a <- getMeanPreci(TS[, 2], method = 'spring', yearIndex = year, monthIndex = month,
                  fullResults = TRUE)
a
```

```
##    1990    1991    1992    1993    1994    1995    1996    1997
##      NA 437.565 461.187      NA 445.005 458.583 451.701      NA
```

If missing value is excluded, set omitNA = TRUE.

```
a <- getMeanPreci(TS[, 2], method = 'winter', yearIndex = year, monthIndex = month,
                  omitNA = TRUE, fullResults = TRUE)
a
```

```
##    1990    1991    1992    1993    1994    1995    1996    1997
##      NA 521.916 375.627 403.155 942.555 428.637 623.565 437.565
```

Get special month precipitation, e.g. march.

```
a <- getMeanPreci(TS[, 2], method = 3, yearIndex = year, monthIndex = month,
                  fullResults = TRUE)
a
```

```
##    1990    1991    1992    1993    1994    1995    1996    1997
##      NA 127.968 136.989      NA 140.058 179.769  81.561 265.887
```

We can also get annual precipitation.

```
a <- getMeanPreci(TS[, 2], method = 'annual', yearIndex = year, monthIndex = month,
                  fullResults = TRUE)
```

# 2. Forecasting

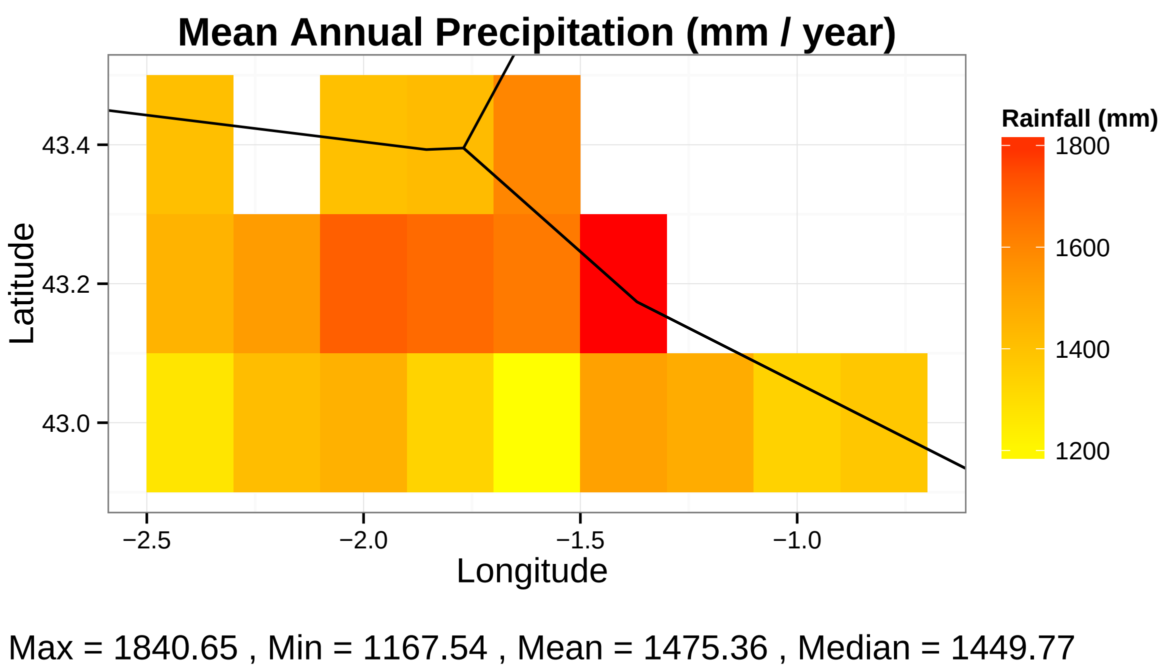
**Note** If an ensemble forecasting data is loaded, there will be one dimension called "member", by default, `hyfo` will calculate the mean of different members. If you want to see a special member, add `member` argument to `getSpatialMap`, e.g., `getSpatialMap(tgridData, method = 'meanAnnual', member = 3)`, `getPreciBar(tgridData, method = 'annual', member = 14)`

## 2.1 Plot Spatial Map

As described at the start of the mannual, `hyfo` is mainly in charge of the post processing of the forecast data. Input of `hyfo` should be the result from `loadGridData{ecomsUDG.Raccess}` or `loadECOMS{ecomsUDG.Raccess}`. An example is included in the package.

If we want to see the mean daily precipitation.

```
data(tgridData)
a <- getSpatialMap(tgridData, method = 'meanAnnual')
```

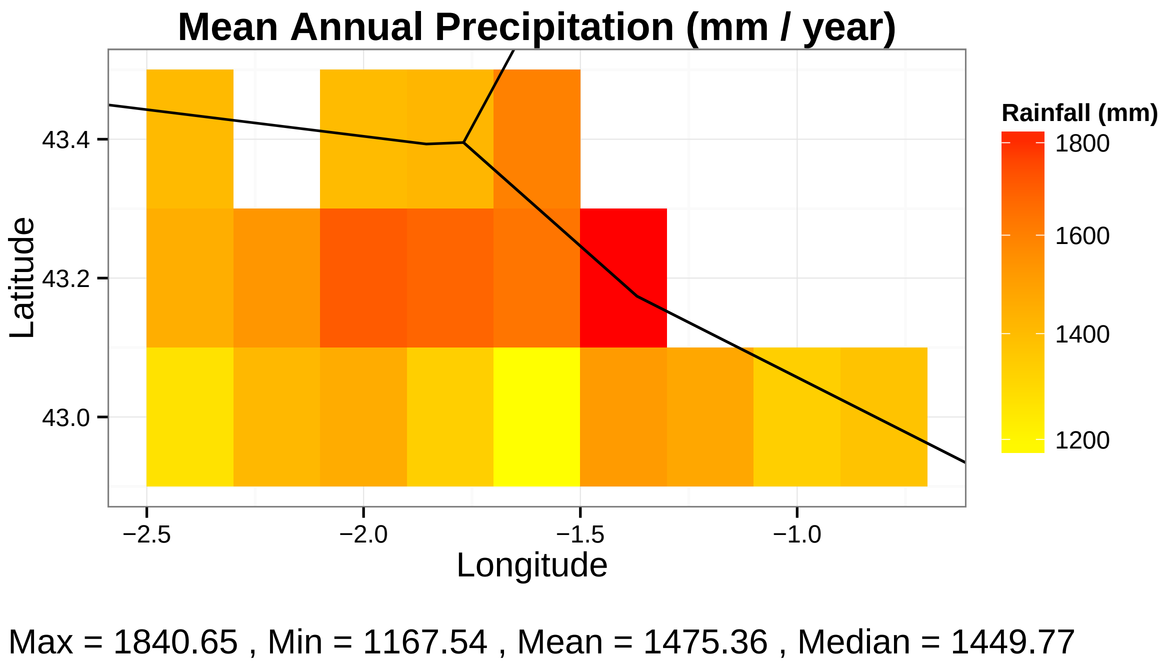



There are several methods to be seleted in the function, details can be found by `?getSpatialMap`.

Sometimes there exists a great difference in the whole map, e.g., the following value, `c(100, 2, 2,6, 1,7)`, since the maximum value is too large, so in the plot, by normal plot scale, we can only recognize value 100 and the rest, it's hard for us to tell the difference between 2, 2.6, and 1.7 from the plot. In this situation, the value needs to be processed before plotting. Here `scale` provides a way to decide the plot scale.

`scale` passes the arguments to the `trans` argument in `ggplot2`. The most common scale is "sqrt" and "log10", which focus more on the minutiae. Default is "identity", which means no change to the plot scale.

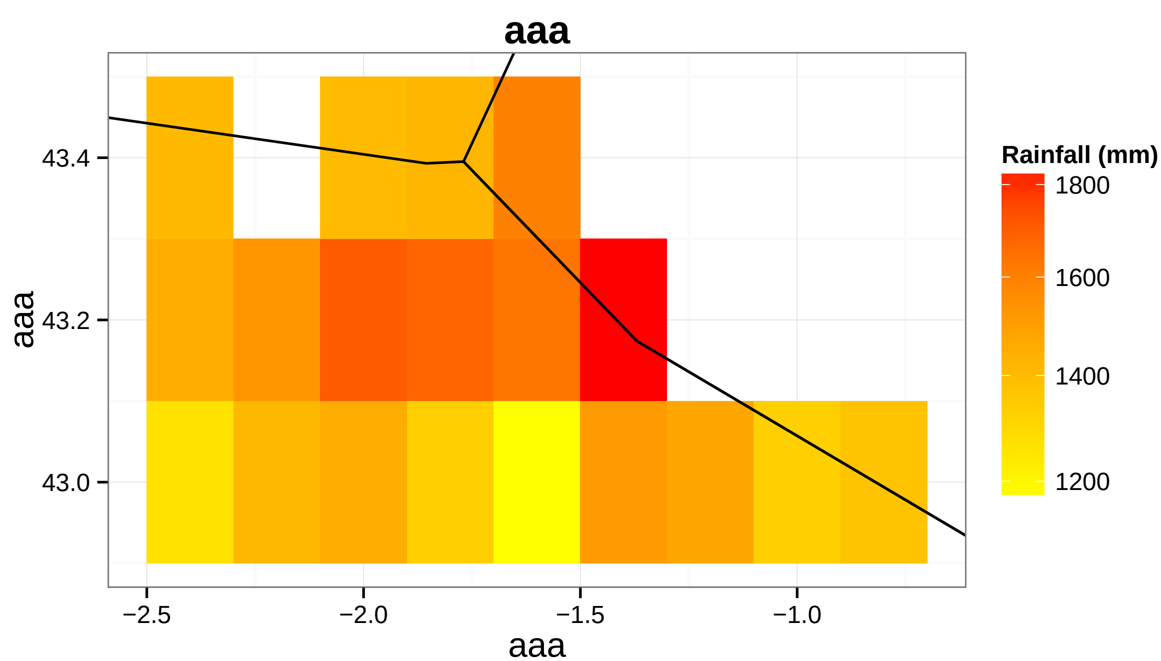
```
a <- getSpatialMap(tgridData, method = 'meanAnnual', scale = 'sqrt')
```




Here in our example, because the region is too small, and the differences is not so big, so it's not so obvious to tell from the plot. But if in a map, both dry region and wet region is included, that will be more obvious to see the difference between the plot scales.

Also, if you are not satisfied with the title, x axis and y axis, you can assgin yourself.

```
a <- getSpatialMap(tgridData, method = 'meanAnnual', scale = 'sqrt',
                   title = 'aaa', x = 'aaa', y = 'aaa')
```



## 2.2 Add Background (catchment and gauging stations)

The default background is the world map, while if you have other backgrounds like catchment shape file and station location file, you are welcome to import them as background.

### 2.2.1 Add catchment shape file

Catchment shape file needs to be processed with a very simple step. It's based on the package `rgdal`, details can be found by `?shp2cat`

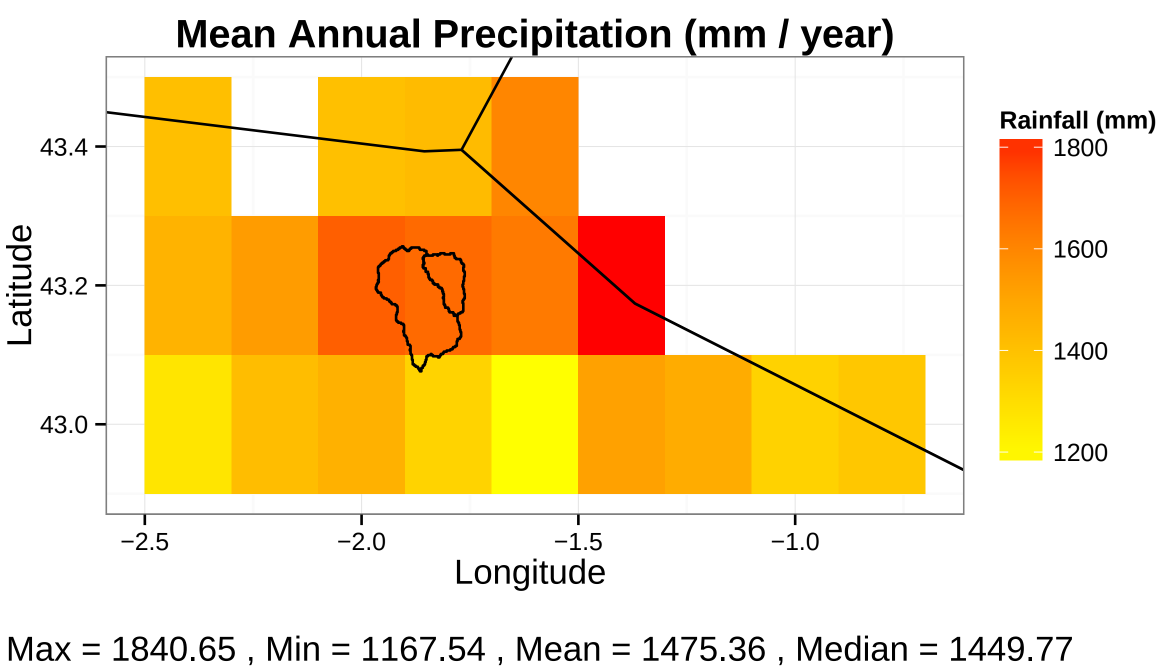
```
# Use the test file provided by hyfo
file <- system.file("extdata", "testCat.shp", package = "hyfo")
cat <- shp2cat(file)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:/data/hyfo/inst/extdata", layer: "testCat"
## with 2 features
## It has 4 fields
```

```
# cat is the catchment file.
```

Then the catchment file `cat` can be inputed as background.

```
a <- getSpatialMap(tgridData, method = 'meanAnnual', catchment = cat)
```

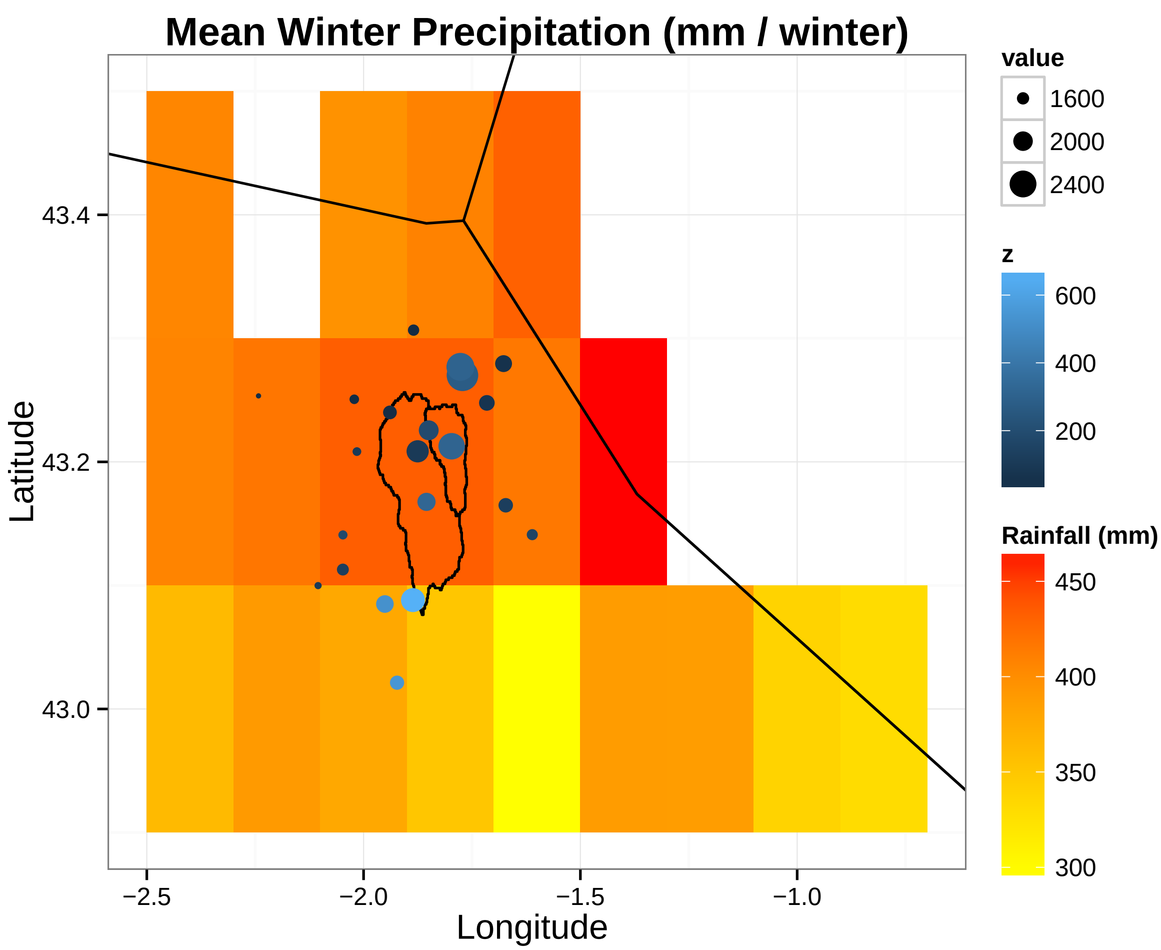



### 2.2.2 Add station locations

Points file needs to be read into dataframe, and special column has to be assigned, details can be found by `?getSpatialMap_mat`

```r
# Use the points file provided by hyfo
file <- system.file("extdata", "points.txt", package = "hyfo")
points <- read.table(file, header = TRUE, sep = ',' )
getSpatialMap(tgridData, method = 'winter', points = points, catchment = cat)
```

```
##          -2.4     -2.2       -2     -1.8     -1.6     -1.4     -1.2
## 43   361.4828 390.5670 377.9069 350.8193 297.0026 388.4785 387.3985
## 43.2 407.4378 417.6438 433.4606 433.5258 416.7791 465.6031       NA
## 43.4 406.1767       NA 396.6478 409.2967 431.8485       NA       NA
## 43.6       NA       NA       NA       NA       NA       NA       NA
## 43.8       NA       NA       NA       NA       NA       NA       NA
##            -1     -0.8 -0.6 -0.4
## 43   338.2406 330.1733   NA   NA
## 43.2       NA       NA   NA   NA
## 43.4       NA       NA   NA   NA
## 43.6       NA       NA   NA   NA
## 43.8       NA       NA   NA   NA
```




Max = 465.6 , Min = 297 , Mean = 391.6 , Median = 396.65

As can be seen above, the color of the points represents the elevation, the size of the points represents the value, e.g., rainfall value.

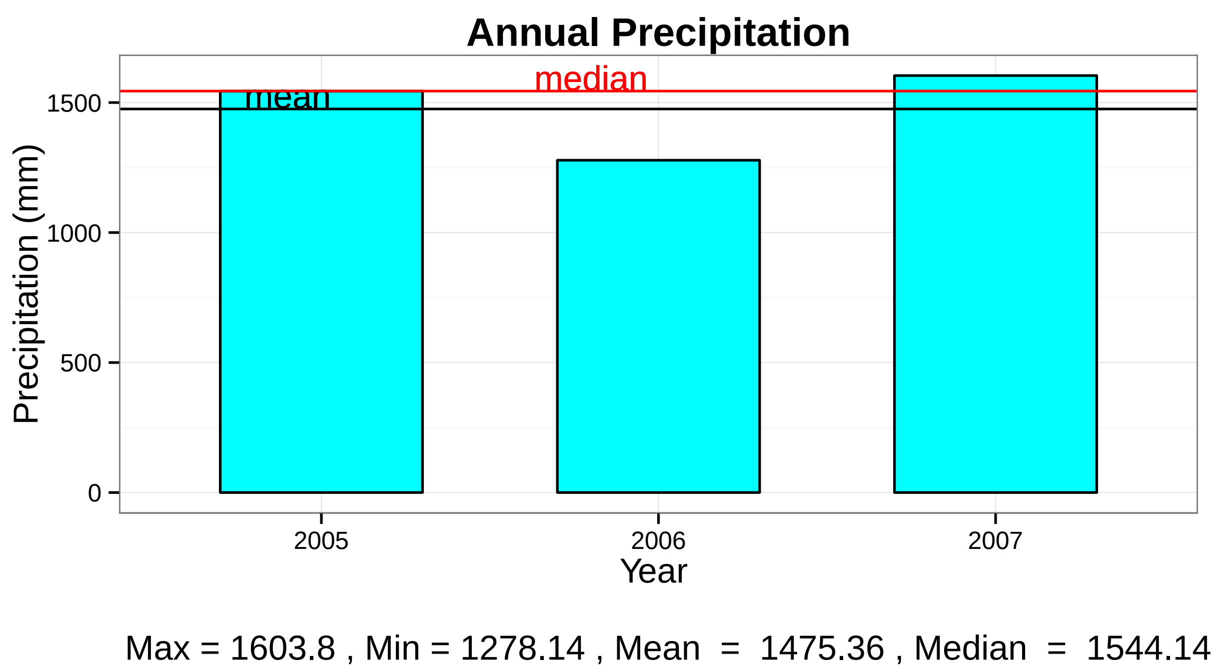
### 2.3 Precipitation Bar Plot

Bisides spatial map, bar plot can also be plotted. The value in the bar plot is spatially averaged, i.e. the value in the bar plot is the mean value over the region.

Annual precipitation.

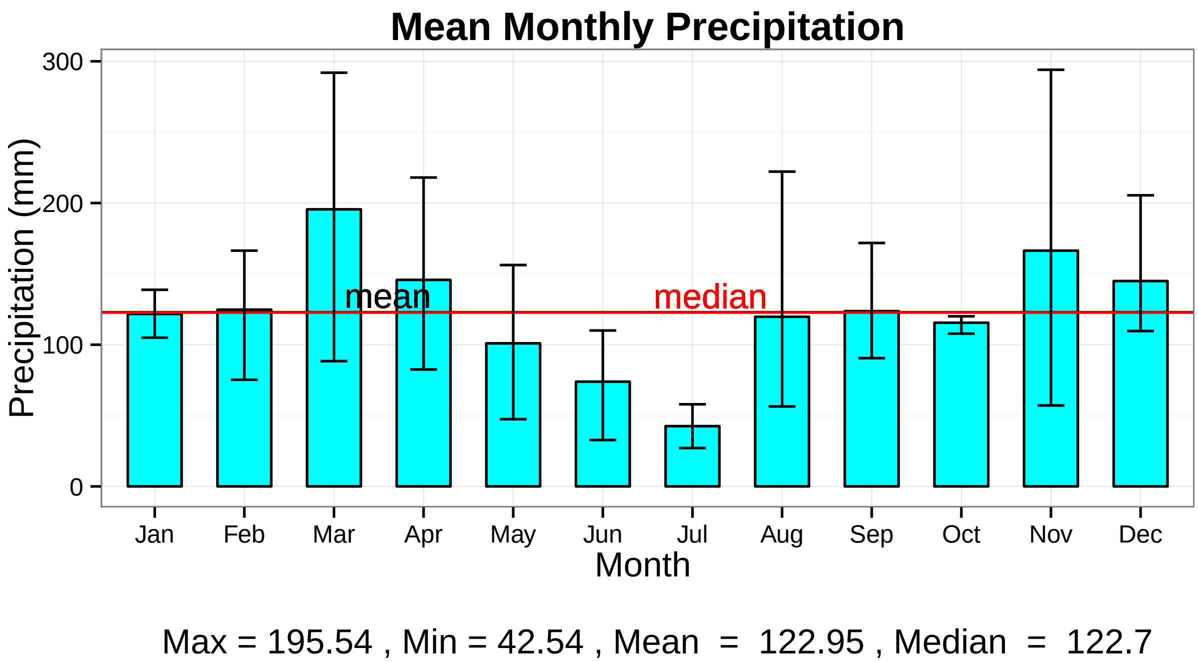
```r
data(tgridData)
a <- getPreciBar(tgridData, method = 'annual')
```

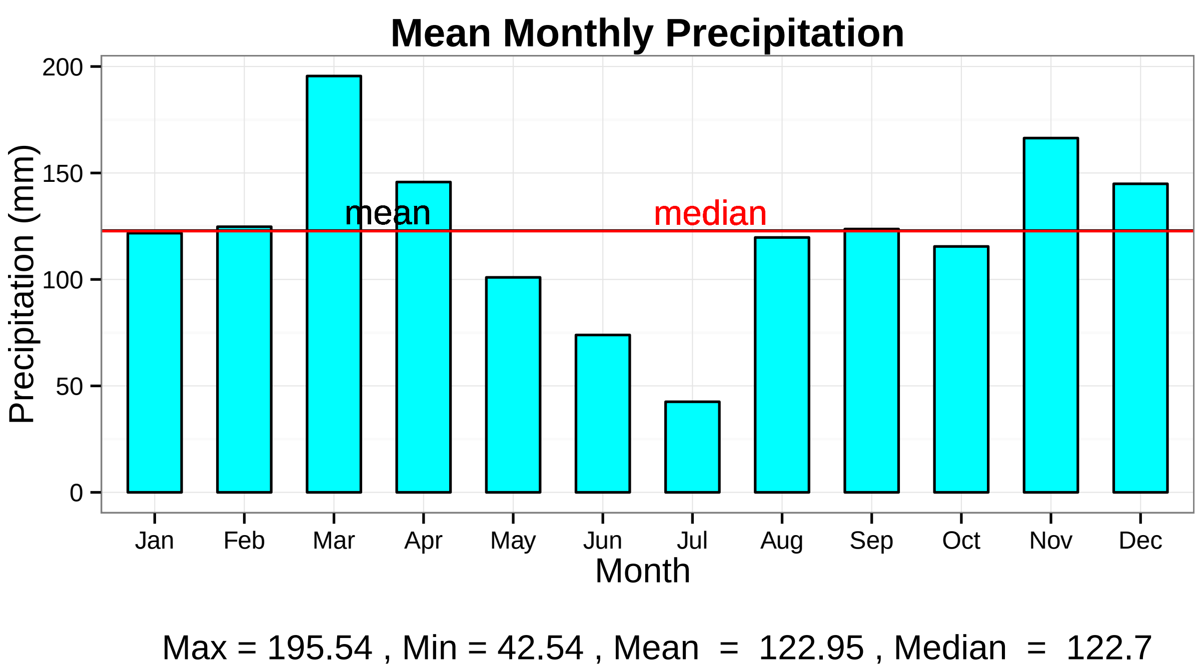
```
## There is no plotRange for this method
```




Mean monthly precipitation over the whole period, with the ranges for each month. But not all kinds of bar plot have a plot range.

```r
a <- getPreciBar(tgridData, method = 'meanMonthly')
a <- getPreciBar(tgridData, method = 'meanMonthly', plotRange = FALSE)
```



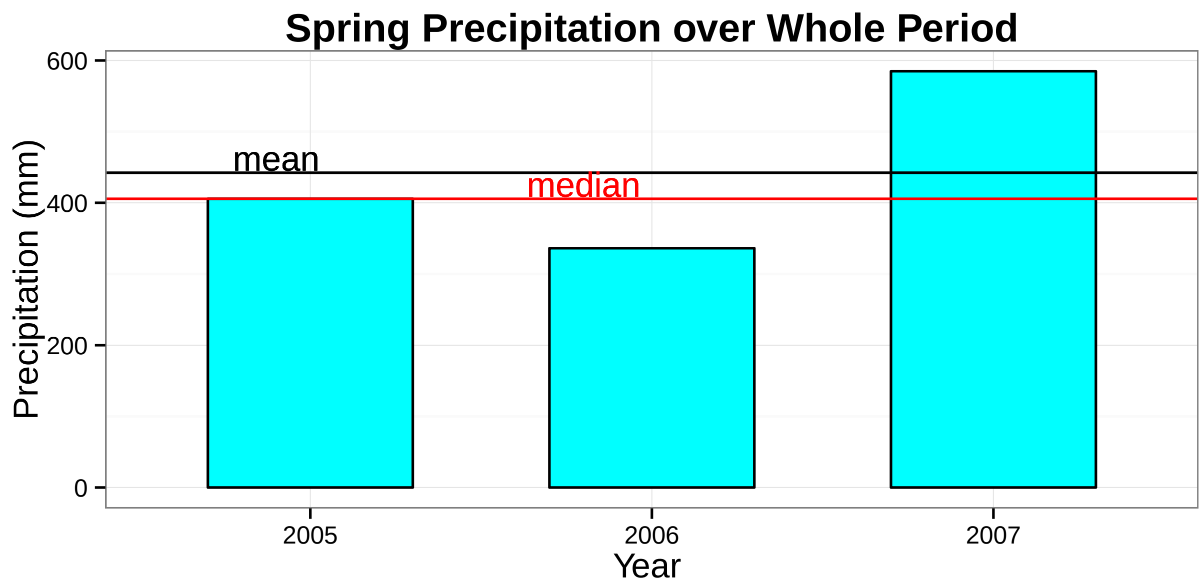

Seasonal precipitation, and monthly precipitation can also be plotted.

```
a <- getPreciBar(tgridData, method = 'spring')# spring precipitation for each year
```

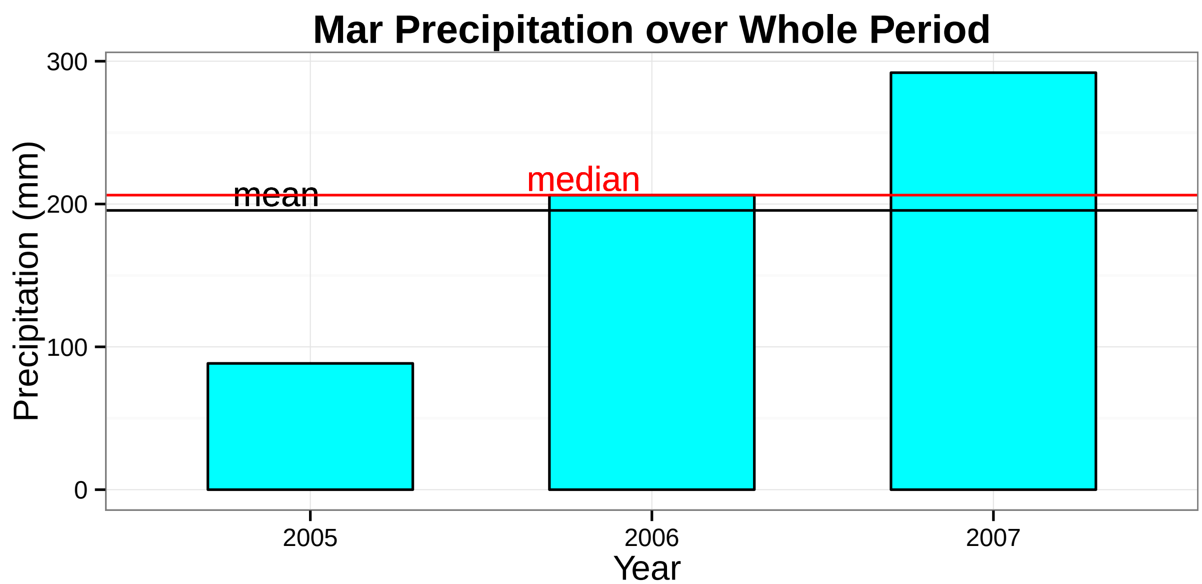
```
## There is no plotRange for this method
```

```r
a <- getPreciBar(tgridData, method = 3) # march precipitation for each year
```

```
## There is no plotRange for this method
```




Max = 584.88 , Min = 336.23 , Mean = 442.25 , Median = 405.64




Max = 291.97 , Min = 88.41 , Mean = 195.54 , Median = 206.23

### 2.4 Analysis and Comparison

For some cases, analysis and comparison are necesssary, which are also provided by `hyfo`.

There are three different kinds of output from `getSpatialMap` and `getPreciBar`, respectively, `output = 'data'`, `output = 'ggplot'` and `output = 'plot'`.

`output = 'data'` is default in the function and do not need to be declare when input. It is mainly used in analyzing and replot the results.

`output = 'ggplot'` is used when combining different plots.

`output = 'plot'` is used when a layer output is needed. the output can be directly printed, and can be mannually combined by the plot arrange functions, e.g., `grid.arrange()`

**Note: All the comparisons must be comparable, e.g.,**

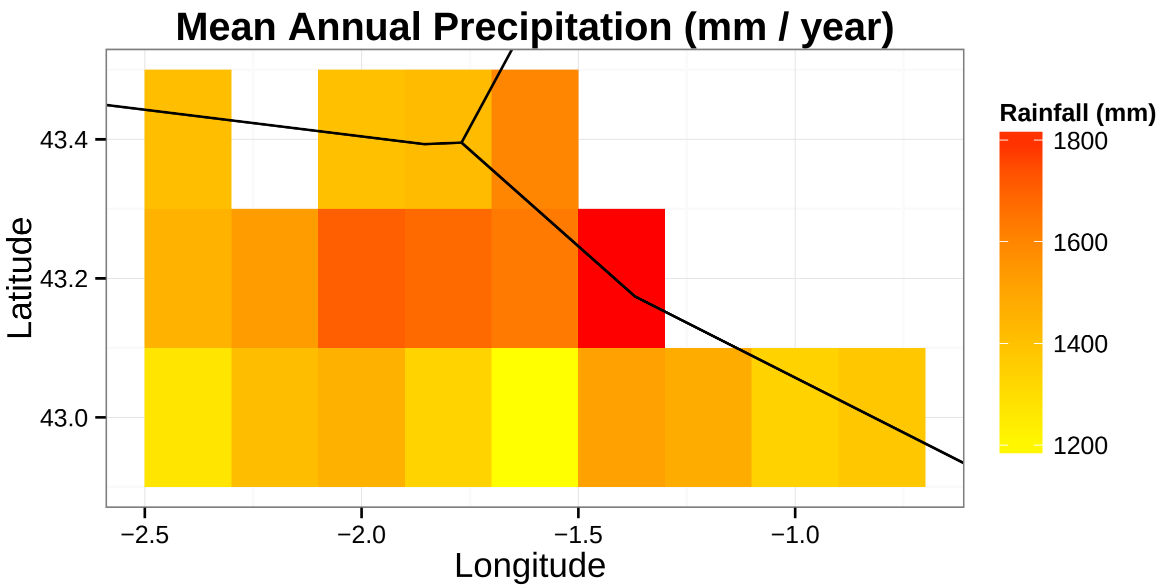
- For `getSpatialMap_comb`, the maps to be compared should be with same size and resolution, in other words, they should be fully overlapped by each other. Check `?getSpatialMap_comb` for details.
- For `getPreciBar_comb`, the bar plots to be compared should belong to the same kind, e.g., spring and winter, January and December, and couldn't be spring and annual. Details can be found by `?getPreciBar_comb`

### 2.4.1 Spatial Map

The default "data" output provides a matrix, representing the raster information of the spatial map.

```
a <- getSpatialMap(tgridData, method = 'meanAnnual')
a
```

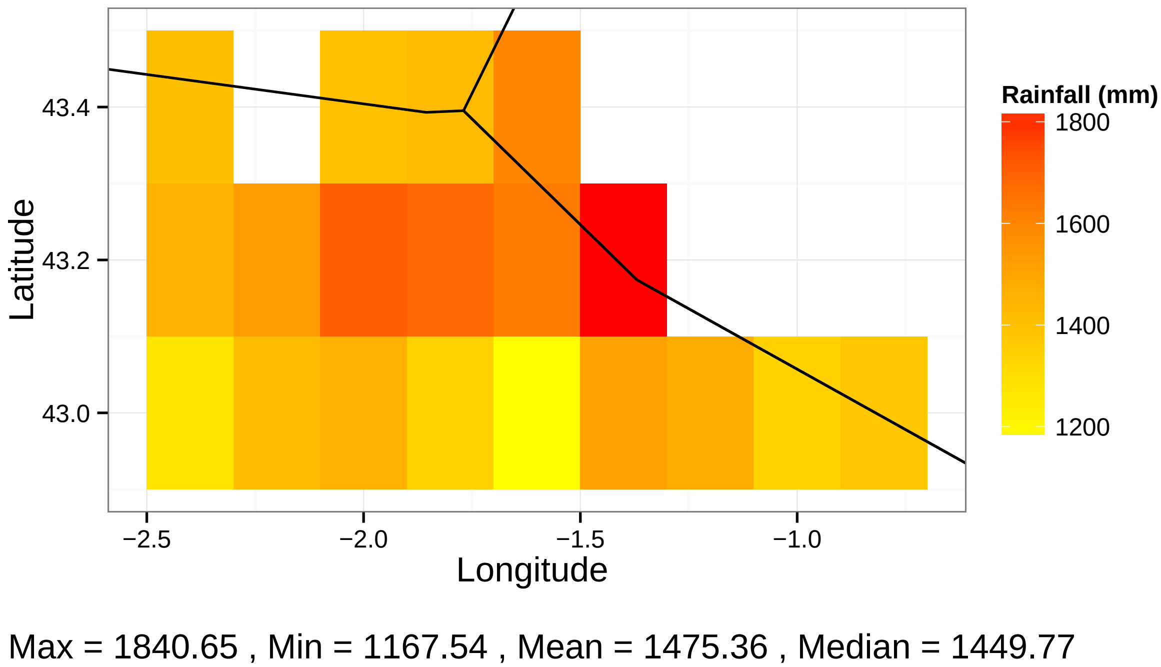
```
##          -2.4     -2.2       -2     -1.8     -1.6     -1.4     -1.2
## 43   1265.663 1414.792 1459.179 1331.803 1167.537 1515.733 1476.697
## 43.2 1449.765 1533.385 1711.032 1683.085 1638.575 1840.649       NA
## 43.4 1406.753       NA 1404.264 1420.504 1601.129       NA       NA
## 43.6       NA       NA       NA       NA       NA       NA       NA
## 43.8       NA       NA       NA       NA       NA       NA       NA
##            -1     -0.8 -0.6 -0.4
## 43   1334.274 1377.036   NA   NA
## 43.2       NA       NA   NA   NA
## 43.4       NA       NA   NA   NA
## 43.6       NA       NA   NA   NA
## 43.8       NA       NA   NA   NA
```

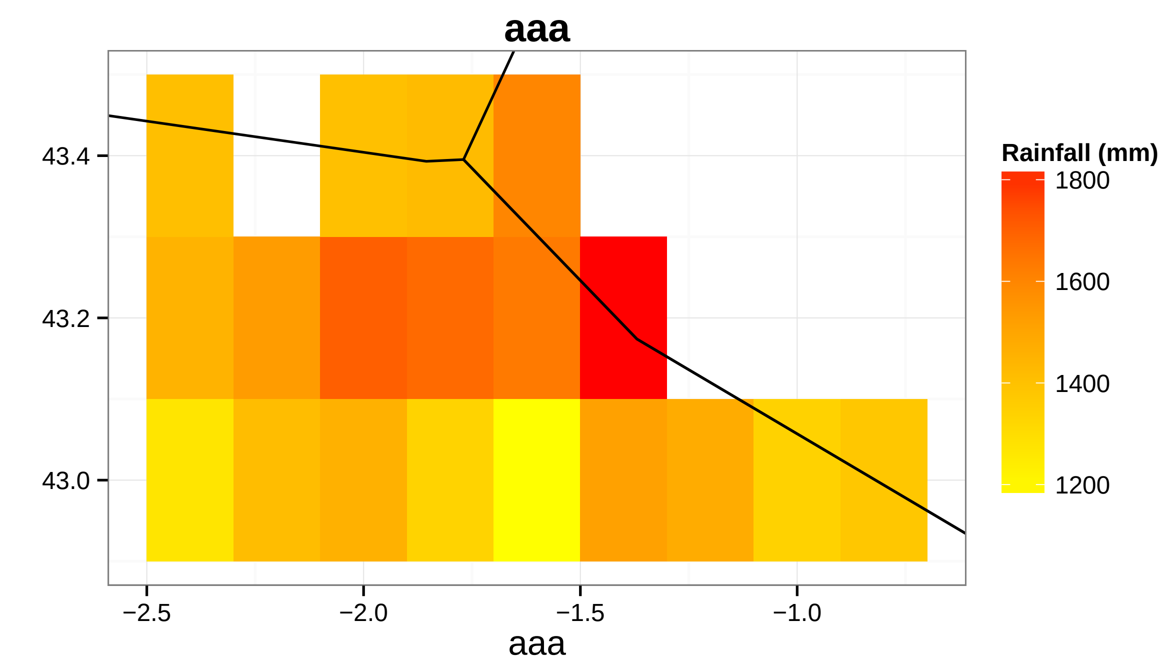



Max = 1840.65 , Min = 1167.54 , Mean = 1475.36 , Median = 1449.77

This matrix is upside down from what you can see from the plot. **DO NOT try to change this matrix.** `hyfo` can deal with it.

```
# For re-plot the matrix
b <- getSpatialMap_mat(a)

# Without title and x and y, also you can assign yourself.
b <- getSpatialMap_mat(a, title = 'aaa', x = 'aaa', y = '')
```



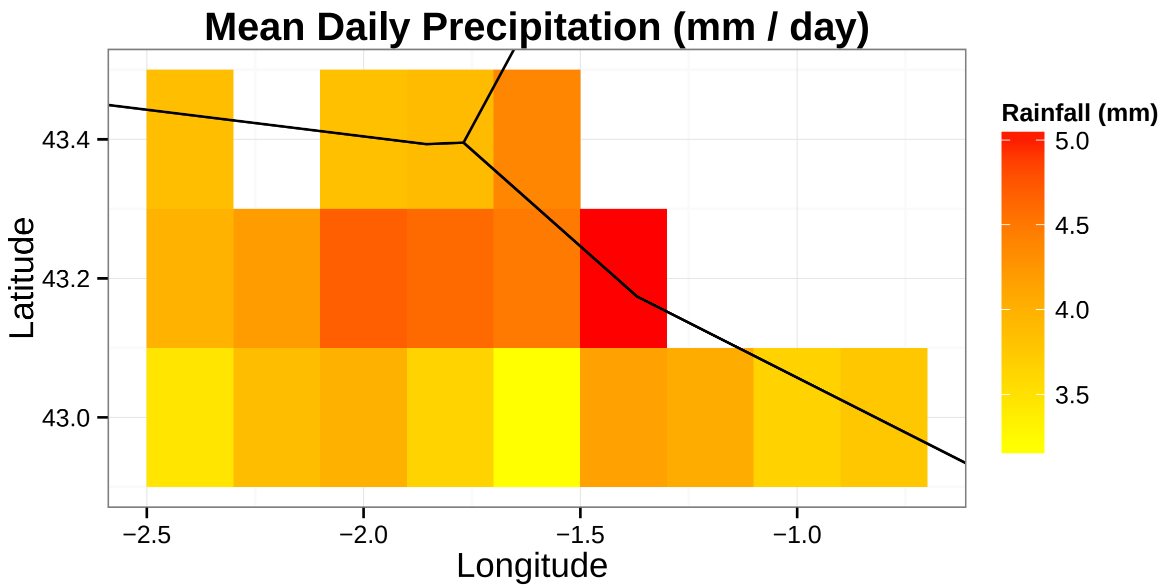




The matrix can be used to make different analysis and plot again.

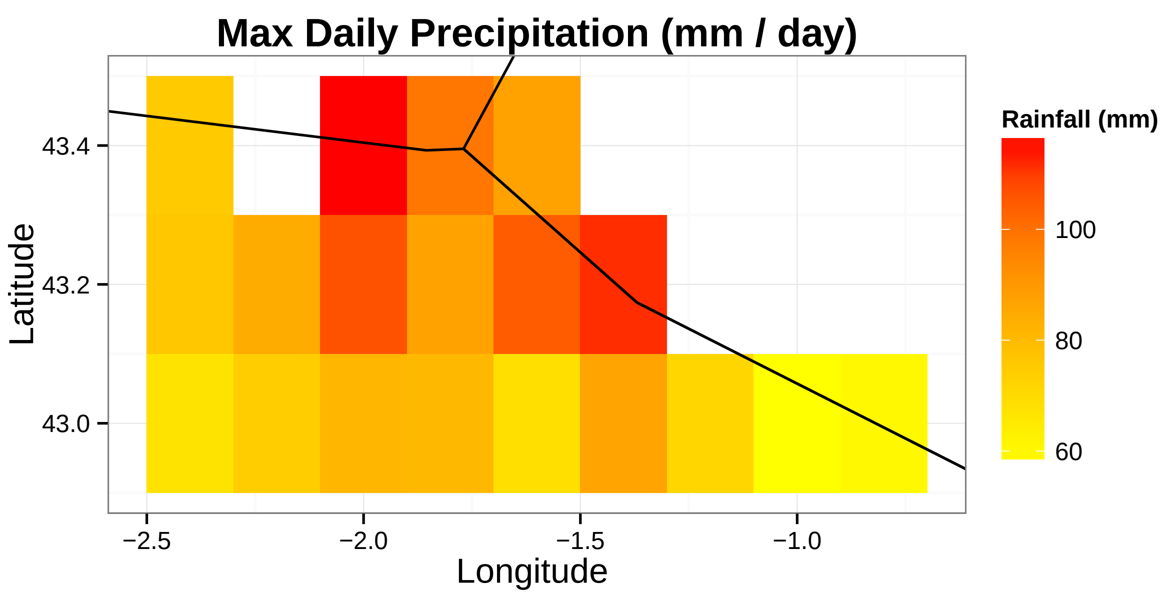
**Note If the matrix doesn't come from `getSpatialMap`, dimension name of longitude and latitude needs to be provided to the matrix, in order to be plotted.**

```
a1 <- getSpatialMap(tgridData, method = 'mean')
a2 <- getSpatialMap(tgridData, method = 'max')
```

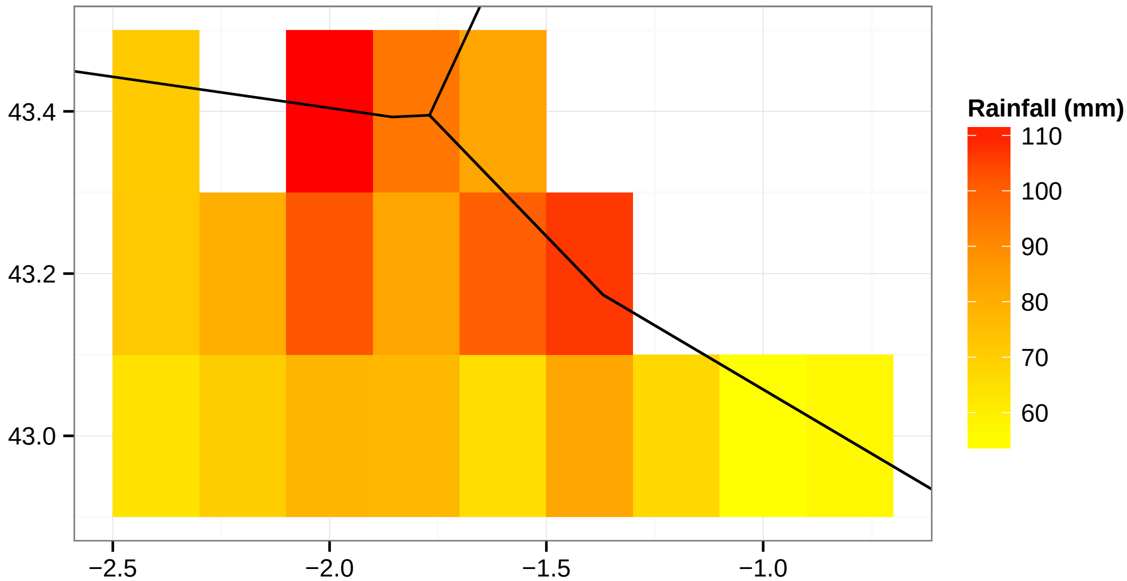
```r
# To see the difference between mean value and maximum value.
b <- a2 - a1
getSpatialMap_mat(b, title = '', x = '', y = '')

# To make some changes to mean value.
b <- a1 * 3 -1
getSpatialMap_mat(b, title = '', x = '', y = '')

# Bias, variation and other analysis can also be processed
# the same way.
# Just apply the analysis to the matrix and
# use getSpatialMap_mat to plot.
```

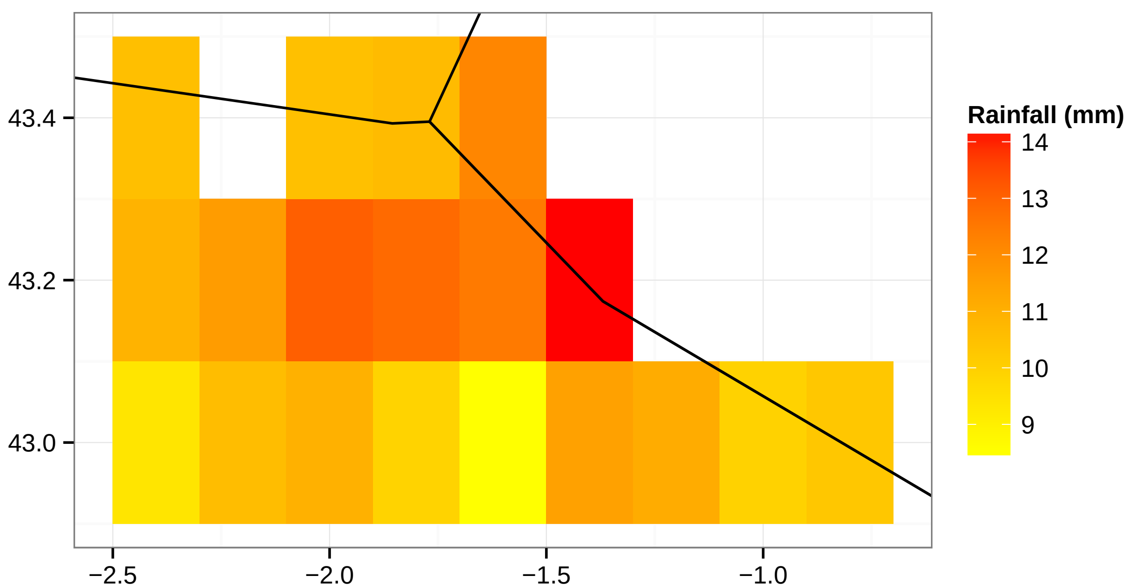



Max = 5.04 , Min = 3.2 , Mean = 4.04 , Median = 3.97




Max = 115.81 , Min = 57.68 , Mean = 84.01 , Median = 81.4




If multi-plot is needed, `hyfo` can also combine different plots together. Use `output = ggplot`, which gives back the a special format that can be easily used by `ggplot2`

```
a1 <- getSpatialMap(tgridData, method = 'spring', output = 'ggplot')
```

```
a2 <- getSpatialMap(tgridData, method = 'summer', output = 'ggplot')
```

```r
a3 <- getSpatialMap(tgridData, method = 'meanAnnual', output = 'ggplot')
```

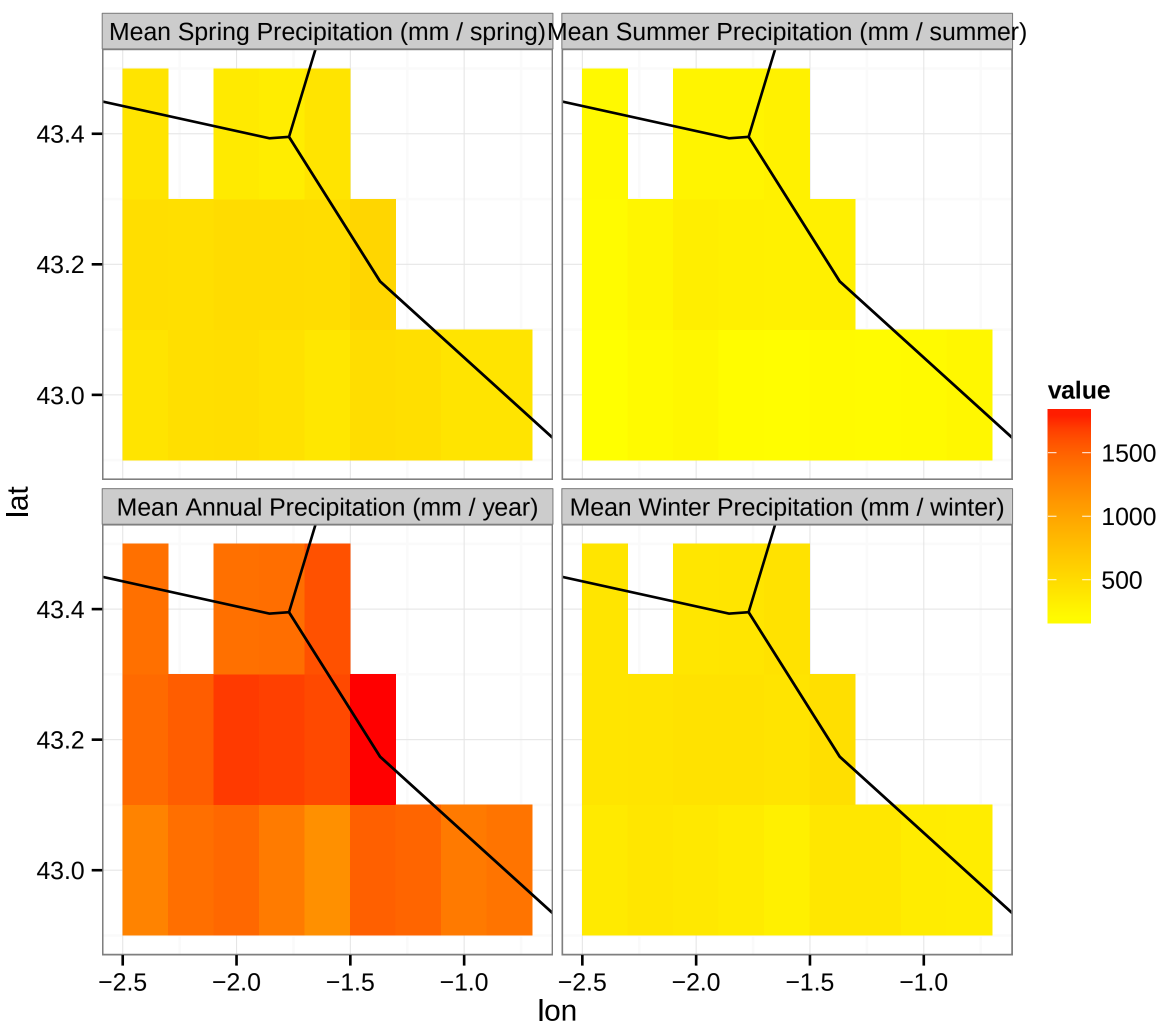
```r
a4 <- getSpatialMap(tgridData, method = 'winter', output = 'ggplot')
```

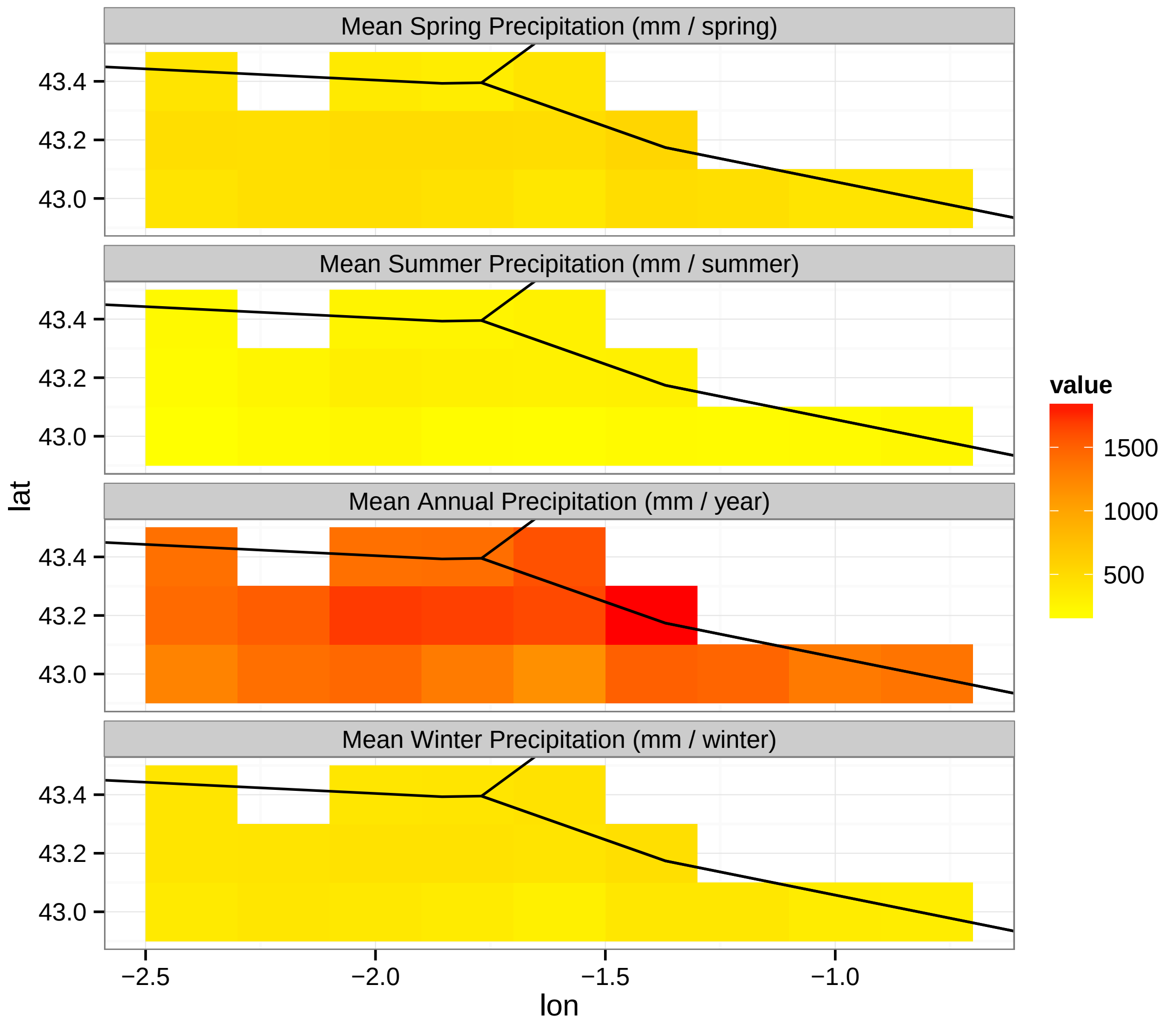
```r
getSpatialMap_comb(a1, a2, a3, a4, nrow = 2)# you cannot assign title
```

```
## Warning in `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels)
## else paste0(labels, : duplicated levels in factors are deprecated
```

```r
getSpatialMap_comb(a1, a2, a3, a4, nrow = 4)
```

```
## Warning in `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels)
## else paste0(labels, : duplicated levels in factors are deprecated
```





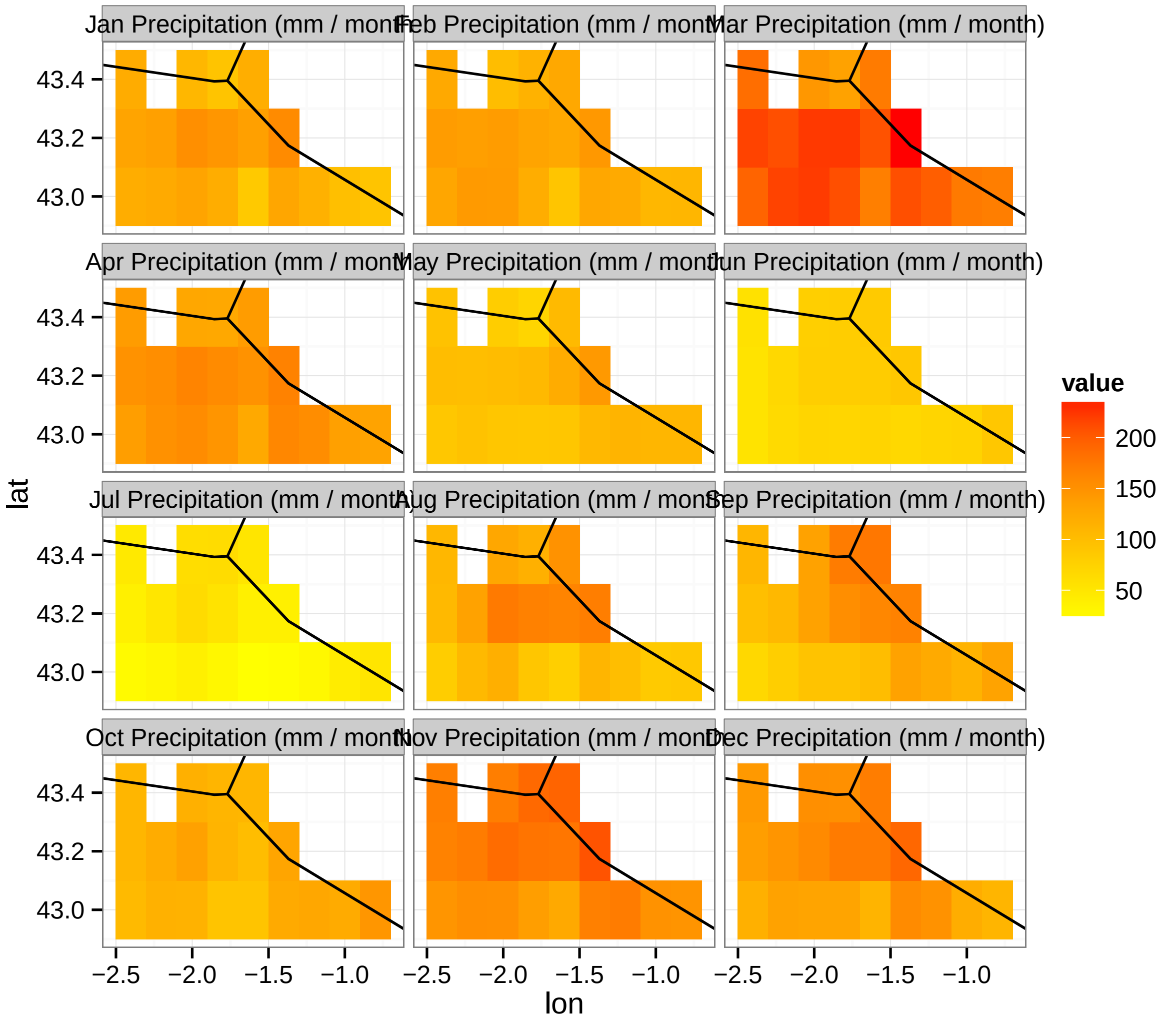

`getSpatialMap_comb` accepts list (using `list =`) object too, which is easier for multi-plot. First list of 12 months are got.

```
c <- lapply(1:12, function(x) getSpatialMap(tgridData, method = x, output = 'ggplot') )
```

Then they are combined.

```
getSpatialMap_comb(list = c, nrow = 4)
```

```
## Warning in `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels)
## else paste0(labels, : duplicated levels in factors are deprecated
```




### 2.4.2 Bar Plot

Basically, bar plot follows the same rule as part 2.4.1 spatial map, only a few cases that needs to pay attention.

```
b1 <- getPreciBar(tgridData, method = 'spring', output = 'ggplot')
```

```
## There is no plotRange for this method
```

```
b2 <- getPreciBar(tgridData, method = 'summer', output = 'ggplot')
```

```
## There is no plotRange for this method
```

```
b3 <- getPreciBar(tgridData, method = 'autumn', output = 'ggplot')
```

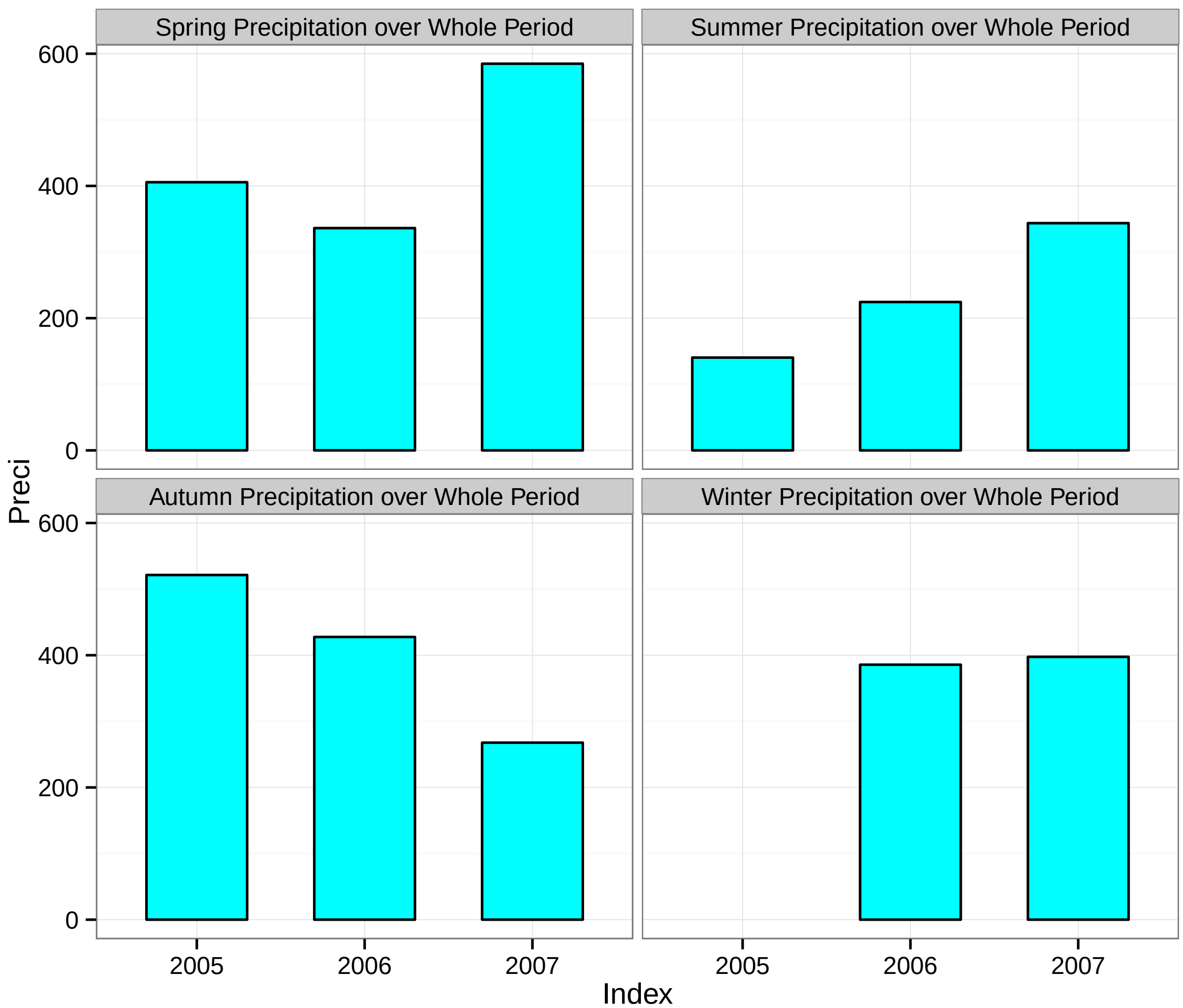
```
## There is no plotRange for this method
```

```r
b4 <- getPreciBar(tgridData, method = 'winter', output = 'ggplot')
```

```
## There is no plotRange for this method

## Warning: Removed 1 rows containing missing values (position_stack).
```

```r
getPreciBar_comb(b1, b2, b3, b4, nrow = 2)
```

```
## Warning in `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels)
## else paste0(labels, : duplicated levels in factors are deprecated
```




```r
c <- lapply(1:12, function(x) getPreciBar(tgridData, method = x, output = 'ggplot') )
```

```
## There is no plotRange for this method
## There is no plotRange for this method

## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
## There is no plotRange for this method
```

```
getPreciBar_comb(list = c, nrow = 4)
```

```
## Warning in `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels)
## else paste0(labels, : duplicated levels in factors are deprecated
```


## 3. Anarbe Case

The functions with anarbe case end with `_anarbe`, all of them are used to collect different available published data in anarbe catchment in Spain. The data comes from two website: linked phrase and linked phrase, there are precipitation or discharge data on those website, and can be downloaded directly.

Since the available files on those website are arranged by a year or five years, for long term data collection, a tools is necessary for collecting data from different files.

**Note:** For excel files, if you have access to the dam regulation excel file of the dam anarbe, you can use `collectData_excel_anarbe` in the package, but this function is commented in the original code, cannot be used directly. Go to original file in the library or go to github linked phrase, copy the original code.

There are two csv files and txt files included in the package, which can be used as examples.

```
file <- system.file("extdata", "1999.csv", package = "hyfo")
folder <- strsplit(file, '1999')[[1]][1]

a <- collectData_csv_anarbe(folder, output = TRUE)
```

```
## C:/data/hyfo/inst/extdata/1999.csv
## C:/data/hyfo/inst/extdata/2000.csv
```

```r
str(a)
```

```
## 'data.frame':    731 obs. of  2 variables:
##  $ Date: Date, format: "1999-01-01" "1999-01-02" ...
##  $ inst: num  0 1.4 0 0 0 0 0 19.7 42.9 4.7 ...
```

```r
b <- collectData_txt_anarbe(folder, output = TRUE)
```

```
## Warning in anarbe_txt(dataset = a, x1, x2): NAs introduced by coercion

## Warning in anarbe_txt(dataset = a, x1, x2): NAs introduced by coercion

## Warning in anarbe_txt(dataset = a, x1, x2): NAs introduced by coercion

## C:/data/hyfo/inst/extdata/1999.TXT

## Warning in anarbe_txt(dataset = a, x1, x2): NAs introduced by coercion

## C:/data/hyfo/inst/extdata/2004.TXT
## new file should be located a different location than the excel folder,
##          in order to avoid error.
##          At least 2 excels should be in the folder
```

```r
str(b)
```

```
## 'data.frame':    3353 obs. of  2 variables:
##  $ Date: Factor w/ 3353 levels "1999-10-28","1999-10-29",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ inst: num  0 0 0.4 0 0.2 46.6 1.8 0 0 31.2 ...
```