

# Package ‘EmiStatR’

September 28, 2021

**Type** Package

**Title** Emissions and Statistics in R for Wastewater and Pollutants in  
Combined Sewer Systems

**Version** 1.2.3.0

**Date** 2021-09-14

**Author** J.A. Torres-Matallana [aut, cre]  
K. Klepiszewski [aut, cre]  
U. Leopold [ctb]  
G. Schutz [ctb]  
G.B.M. Heuvelink [ctb]

**Maintainer** J.A. Torres-Matallana <arturo.torres@list.lu>

**Description** Provides a fast and parallelised calculator to estimate combined wastewater emissions.  
It supports the planning and design of urban drainage systems, without the requirement of  
extensive simulation tools. The ‘EmiStatR’ package implements modular R methods. This enables  
to add new functionalities through the R framework.

**License** GPL (>= 3)

**Depends** R (>= 2.10), methods

**Imports** utils, grDevices, graphics, stats, xts, zoo, foreach,  
parallel, lattice, doParallel

**NeedsCompilation** no

**StagedInstall** yes

**Repository** CRAN

**Date/Publication** 2021-09-28 07:30:06 UTC

## R topics documented:

EmiStatR-package . . . . .	2
Agg . . . . .	3
CInp2TS . . . . .	4
Delay . . . . .	6
E1 . . . . .	7

E2	8
E3	9
EmiStatR-methods	10
Esch_Sure2010	12
inf	13
input-class	13
IsReg	16
Level2Volume	18
P1	19
P1_20111216	19
pe_factor	20
qs_factor	21
qs_factor_ATV	21
rw	22
Volume2Level	23
ww	23

<b>Index</b>	<b>25</b>
--------------	-----------

---

EmiStatR-package	<i>Emissions and Statistics in R for Wastewater and Pollutants in Combined Sewer Systems.</i>
------------------	---

---

## Description

Provides a fast and parallelised calculator to estimate combined wastewater emissions. It supports the planning and design of urban drainage systems, without the requirement of extensive simulation tools. The 'EmiStatR' package implements modular R methods. This enables to add new functionalities through the R framework.

## Details

The DESCRIPTION file:

```

Package:   EmiStatR
Type:      Package
Version:   1.2.3.0
Date:      2021-09-13
License:   GPL (>= 3)
Depends:   R (>= 2.10), methods, shiny
Imports:   utils, grDevices, graphics, stats, xts, zoo, foreach, parallel, lattice, doParallel

```

## Author(s)

J.A. Torres-Matallana [aut, cre] K. Klepiszewski [aut, cre] U. Leopold [ctb] G. Schutz [ctb] G.B.M. Heuvelink [ctb]

Maintainer: J.A. Torres-Matallana

## References

J. A. Torres-Matallana, K. Klepiszewski, U. Leopold, and G.B.M. Heuvelink. EmiStatR: a simplified and scalable urban water quality model for simulation of combined sewer overflows. *Water*, 10(6)(782):1-24, 2018. <https://www.mdpi.com/2073-4441/10/6/782/htm>

## See Also

See also the class the method [EmiStatR](#)

---

Agg

*Temporal aggregation of environmental variables*

---

## Description

Function for temporal aggregation of environmental variables. Agg is a wrapper function of aggregate from stats package.

## Usage

```
Agg(data, nameData, delta, func, namePlot)
```

## Arguments

data	A data.frame that contains the time series of the environmental variable to be aggregated, e.g. precipitation. This data.frame should have at least two columns: the first one, Time [y-m-d h:m:s]; the second one, a numeric value equal to the magnitude of the environmental variable. If the environmental variable is different than precipitation, then the column name of the values should be named as value.
nameData	A character string that defines the name of the environmental variable to be aggregated.
delta	A numeric value that specifies the level of aggregation required in minutes.
func	The name of the function of aggregation e.g. mean, sum.
namePlot	A character string that defines the title of the plot generated. Use "" to avoid writing the plot locally.

## Value

A data.frame with two columns:

time	the date-time time series of the aggregated variable
value	time series with the magnitude of the aggregated variable.

## Author(s)

J.A. Torres-Matallana

## Examples

```
## temporal aggregation
library(EmiStatR)
data(P1)
colnames(P1) <- c("time", "value")
head(P1)
tail(P1)

P1.agg <- Agg(data = P1, nameData = "value", delta = 120 , func = sum,
              namePlot = "")

head(P1.agg)
tail(P1.agg)
```

---

CInp2TS

*Function to convert Constant Input to Time Series*

---

## Description

Given daily, weekly and monthly factors, this function converts from a constant numeric input to a time series.

## Usage

```
CInp2TS(cinp, prec, cinp.daily.file, cinp.weekly, cinp.seasonal)
```

## Arguments

<code>cinp</code>	a numeric object that defines the mean constant input to be converted in time series.
<code>prec</code>	A data.frame object with observations on the following 2 variables: <code>time</code> a POSIXct vector <code>'P [mm]'</code> a numeric vector
<code>cinp.daily.file</code>	the path and file name of the comma separated value (csv) file that contains the daily factors. The first column of this file should be the time in format "H:M:S" and should span for 24 hours. The second column should contain the factors as numeric class for the specified time. These factors must average to 1.
<code>cinp.weekly</code>	a "list" that contains the factors per day of the week with 7 elements called "mon" for Monday, "tue" for Tuesday, "wed" for Wednesday, "thu" for Thursday, "fri" for Friday, "sat" for Saturday, and "sun" for Sunday. These factors must average to 1. See example about the definition of this argument.

`cinp.seasonal` a "list" that contains the factors per month of the year with 12 elements called with the three first lower case letters of the month from "jan" for January to "dec" for December. These factors must average to 1. See example about the definition of this argument.

### Details

The generated time series has number of rows equal to the number of rows of the `prec` data.frame. The time index of the time series generated is the same that `prec`.

### Value

Object of class "list". This object contains 3 elements:

<code>time</code>	a POSIXct vector of length <code>n</code> , where <code>n</code> is the number of rows of the <code>prec</code> data.frame.
<code>data</code>	a numeric matrix of size [1:n, 1:4], where columns are: "factor", the daily factor time series based on <code>cinp.daily.file</code> ; "cinp", the daily time series of the variable defined by <code>cinp</code> according to the factors defined by "factor"; "cinp.week", the weekly time series of the variable defined by <code>cinp</code> according to the factors defined by <code>factor</code> and <code>cinp.weekly</code> ; and "cinp.seasonal", the monthly time series of the variable defined by <code>cinp</code> according to the factors defined by <code>factor</code> , <code>cinp.weekly</code> , and <code>cinp.seasonal</code> .
<code>xts</code>	An "xts" object containing the 4 same columns that <code>data</code> .

### Author(s)

J.A. Torres-Matallana; U. Leopold

### Examples

```
library(EmiStatR)
library(zoo)

data("Esch_Sure2010")
data("qs_factor")

cinp      <- 150 # water consumption [m3/h]
prec      <- Esch_Sure2010[1:1000,] # selecting just the first 1,000 rows

cinp.daily.file <- qs_factor
cinp.weekly    <- list(mon=1, tue=.83, wed=.83, thu=.83, fri=1, sat=1.25, sun=1.25)
# factors average to 1
cinp.seasonal  <- list(jan=.79, feb=.79, mar=1.15, apr=1.15, may=1.15, jun=1.15,
                      jul=1.15, aug=1.15, sep=1.15, oct=1.15, nov=.79, dec=.79)
# factors average to 1
ts1 <- CInp2TS(cinp, prec, cinp.daily.file, cinp.weekly, cinp.seasonal)
str(ts1)

head(ts1[["xts"]])
summary(ts1[["xts"]])
```

```

dev.new()
par(mfrow = c(4,1))
plot(index(ts1[["xts"]][,1]), ts1[["xts"]][,1], type = "l",
      xlab = "", ylab = "Daily factor [-]",
      main="Daily factor time series")
plot(index(ts1[["xts"]][,1]), ts1[["xts"]][,2], type = "l",
      xlab = "", ylab = "Water consumption [l/(PE d)]",
      main="Daily water consumption time series")
plot(index(ts1[["xts"]][,1]), ts1[["xts"]][,3], type = "l",
      xlab = "", ylab = "Water consumption [l/(PE d)]",
      main="Weekly water consumption time series")
plot(index(ts1[["xts"]]), ts1[["xts"]][,4], type = "l",
      xlab = "Time", ylab = "Water consumption [l/(PE d)]",
      main="Yearly water consumption time series")

```

---

Delay

*Delay function for time series*


---

### Description

This function allows to create a n-time steps delayed time series, where n is the number of time steps defined by argument x. Henceforth, it is possible to calibrate this argument (parameter) x.

### Usage

```
Delay(P1, x)
```

### Arguments

- |    |  |
|----|--|
| P1 | A data.frame that contains the time series of the environmental variable to be delayed, e.g. precipitation. This data.frame should have at least two columns: the first one, Time [y-m-d h:m:s]; the second one, a numeric value equal to the magnitude of the environmental variable. If the environmental variable is different than precipitation, then the column name of the values should be named as value. |
| x  | A numeric value that specifies the delayed time in time steps.   |

### Value

A data.frame with two columns:

- |       |  |
|-------|--|
| time  | the date-time time series of the delayed variable  |
| value | time series with the magnitude (equal to the original, P1, time series) of the delayed variable. |

### Author(s)

J.A. Torres-Matallana

**Examples**

```

library(EmiStatR)
data(P1)

P1_delayed <- Delay(P1 = P1, x = 500)

head(P1_delayed)

dev.new()
par(mfrow = c(2, 1))
plot(P1[,1], P1[,2], typ = "l", col = "blue")
plot(P1_delayed[,1], P1_delayed[,2], typ = "l", col = "red")

```

---

E1	<i>Dataset: An example of combined sewer overflow (CSO) input for Goesdorf structure (CSO chamber, CSOC).</i>
----	---

---

**Description**

This dataset is an example of input data for the CSOC at Goesdorf, Grand-Duchy of Luxembourg.

**Usage**

```
data("E1")
```

**Format**

A list of 18 elements. This object contains the first structure (CSOC) to simulate.

**id:** numeric, identification number [-];  
**ns:** character, name of the structure [-];  
**nm:** character, name of the municipality [-];  
**nc:** character, name of the catchment [-];  
**numc:** numeric, number of the catchment [-];  
**use:** character, use of the soil [-];  
**Atotal:** numeric, total area [ha];  
**Aimp:** numeric, impervious area [ha];  
**tfS:** numeric, time flow structure [time step];  
**pe:** numeric, population equivalent [PE];  
**V:** numeric, volume [m3];  
**lev2vol:** list of 2, lev and vol, defining the curve lev (level [m]) to vol (volume [m3]);  
**lev.ini:** numeric, initial level in the CSOC [m];  
**Qd:** numeric, maximum throttled outflow [l/s];  
**Dd:** numeric, orifice diameter [m];

**Cd:** numeric, orifice coefficient [-];  
**Cimp:** numeric, coefficient for impervious area [-];  
**Cper:** numeric, coefficient for pervious area [-].

### Examples

```

data("E1")

str(E1)

```

---

E2	<i>Dataset: An example of combined sewer overflow (CSO) input for Kaundorf structure (CSO chamber, CSOC).</i>
----	---

---

### Description

This dataset is an example of input data for the CSOC at Kaundorf, Grand-Duchy of Luxembourg.

### Usage

```
data("E2")
```

### Format

A list of 18 elements. This object contains the first structure (CSOC) to simulate.

**id:** numeric, identification number [-];  
**ns:** character, name of the structure [-];  
**nm:** character, name of the municipality [-];  
**nc:** character, name of the catchment [-];  
**numc:** numeric, number of the catchment [-];  
**use:** character, use of the soil [-];  
**Atotal:** numeric, total area [ha];  
**Aimp:** numeric, impervious area [ha];  
**tfS:** numeric, time flow structure [time step];  
**pe:** numeric, population equivalent [PE];  
**V:** numeric, volume [m3];  
**lev2vol:** list of 2, lev and vol, defining the curve lev (level [m]) to vol (volume [m3]);  
**lev.ini:** numeric, initial level in the CSOC [m];  
**Qd:** numeric, maximum throttled outflow [l/s];  
**Dd:** numeric, orifice diameter [m];  
**Cd:** numeric, orifice coefficient [-];  
**Cimp:** numeric, coefficient for impervious area [-];  
**Cper:** numeric, coefficient for pervious area [-].



**Examples**

```
data("E2")

str(E2)
```

---

E3	<i>Dataset: An example of combined sewer overflow (CSO) input for Nocher-Route structure (CSO chamber, CSOC).</i>
----	---

---

**Description**

This dataset is an example of input data for the CSOC at Nocher-Route, Grand-Duchy of Luxembourg.

**Usage**

```
data("E3")
```

**Format**

A list of 18 elements. This object contains the first structure (CSOC) to simulate.

**id:** numeric, identification number [-];  
**ns:** character, name of the structure [-];  
**nm:** character, name of the municipality [-];  
**nc:** character, name of the catchment [-];  
**numc:** numeric, number of the catchment [-];  
**use:** character, use of the soil [-];  
**Atotal:** numeric, total area [ha];  
**Aimp:** numeric, impervious area [ha];  
**tfs:** numeric, time flow structure [time step];  
**pe:** numeric, population equivalent [PE];  
**V:** numeric, volume [m3];  
**lev2vol:** list of 2, lev and vol, defining the curve lev (level [m]) to vol (volume [m3]);  
**lev.ini:** numeric, initial level in the CSOC [m];  
**Qd:** numeric, maximum throttled outflow [l/s];  
**Dd:** numeric, orifice diameter [m];  
**Cd:** numeric, orifice coefficient [-];  
**Cimp:** numeric, coefficient for impervious area [-];  
**Cper:** numeric, coefficient for pervious area [-].

**Examples**

```
data("E3")

str(E3)
```

**Description**

S4 methods for function EmiStatR. Given the inputs either from the Shiny applications "EmiStatR\_input - Shiny" and "EmiStatR\_inputCSO - Shiny" or user-defined, the methods invoke the main core of the tool and writes the output files in the specified folder.

**Usage**

```
EmiStatR(x)
```

**Arguments**

x                    An object of class input

**Value**

Object of class "list". This object contains N lists, where N is the number of structures to simulate. Each list contains a list with three elements: a data.frame named "out1", a data.frame named "out2", a vector named "lista". "out1" contains n observations of 24 variables, where n is the length of the precipitation time series. The 24 variables are the following time series:

1. id, identification number
2. Time [y-m-d h:m:s]
3. P [mm], precipitation
4. i [mm/(min)], intensity (if available)
5. V\_r [m3], rain water volume
6. V\_dw [m3], dry weather volume
7. cs\_mr [-], combined sewage mixing ratio
8. o\_tfyn [yes=1/no=0], status variable to know when the Combined Sewer Overflow Tank (CSOT) is filling up
9. V\_Tank [m3], volume of CSOT filling up
10. V\_Ov [m3], overflow volume
11. B\_COD\_Ov [kg], Chemical Oxygen Demand (COD) overflow load
12. B\_NH4\_Ov [kg], ammonium (NH4) overflow load
13. C\_COD\_Ov [mg/l], COD overflow concentration
14. C\_NH4\_Ov [mg/l], NH4 overflow concentration
15. d\_Ov [min], total duration of overflows
16. f\_Ov [occurrence], frequency of overflows (just an approximation)
17. V\_InTank [m3], volume at entrance of the CSOT
18. B\_COD\_InTank [Kg], COD load at entrance of the CSOT
19. B\_NH4\_InTank [Kg], NH4 load at entrance of the CSOT
20. C\_COD\_InTank [mg/l], COD concentration at entrance of the CSOT
21. C\_NH4\_InTank [mg/l], NH4 concentration at entrance of the CSOT
22. Q\_Ov [l/s], overflow flow
23. pe.season [PE], population equivalents in the catchment

24 qs.season [l/PE/d], water consumption in the catchment

The summary of the overflow data, "out2", contains 15 observations of 2 variables. The 15 observations are:

1. Period [day], length of time of the precipitation time series
2. Duration, d\_Ov, [min], overflow duration
3. Frequency, f\_Ov, [occurrence] (aprox.), overflow frequency
4. Total volume, V\_Ov, [m3], total overflow volume
5. Average flow, Q\_Ov, [l/s], average overflow flow
6. Total COD load, B\_COD\_Ov, [kg], total COD load in overflow
7. Average COD concentration, C\_COD\_ov\_av, [mg/l], in overflow
8. 99.9th percentile COD concentration, C\_COD\_Ov\_99.9, [mg/l], in overflow
9. Maximum COD concentration, C\_COD\_Ov\_max, [mg/l], in overflow
10. Total NH4 load, B\_NH4\_Ov, [kg], total NH4 load in overflow
11. Average NH4 concentration, C\_NH4\_Ov\_av, [mg/l], in overflow
12. 99.9th percentile NH4 concentration, C\_NH4\_Ov\_99.9, [mg/l], in overflow
13. Maximum NH4 concentration, C\_NH4\_Ov\_max, [mg/l], in overflow
14. Structure summary results, (a descriptive text line)
15. Volume Tank, VTank [m3], total volumen in the CSO tank

"Lista" contains the identification name(s) of the N structure(s). If export is allowed then three plain text .csv files are created, one for "out1", the second for "out2", the third one a summary for all the structures based in "out2". Also, one .pdf file is printed which illustrates the precipitation and Combined Sewer Overflow (CSO) volume, COD concentration, and NH4 concentration time series. These files are exported to the directory EmiStatR\_output located in the folderOutput path.

## Methods

signature(x = "input") execute EmiStatR function

## Author(s)

J.A. Torres-Matallana; K. Klepiszewski; U. Leopold; G.B.M. Heuvelink

## Examples

```
## running GUI
library("EmiStatR")

appDir <- system.file("shiny", package = "EmiStatR")

## (uncomment for running)
# setwd(appDir)
# runApp("EmiStatR_input")
# runApp("EmiStatR_inputCSO")

## executing EmiStatR
input.default <- input()

## uncomment following lines to execute
```

```
# sim <- EmiStatR(input.default)
# str(sim)

## a dummy example of plot
# par(mfrow=c(2,2), oma = c(0,0,2,0))
# plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[3]], typ="l", col="blue",
#       xlab = "time", ylab = colnames(sim[[1]][[1]])[3], main = "Precipitation")
# plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[10]], typ="l", col="blue",
#       xlab = "time", ylab = colnames(sim[[1]][[1]])[10], main = "CSO, volume")
# plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[13]], typ="l", col="blue",
#       xlab = "time", ylab = colnames(sim[[1]][[1]])[13], main = "CSO, COD concentration")
# plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[14]], typ="l", col="blue",
#       xlab = "time", ylab = colnames(sim[[1]][[1]])[14], main = "CSO, NH4 concentration")
# mtext(paste("Structure", sim[[1]][[3]][[1]]), outer=TRUE, cex = 1.5)
```

---

Esch\_Sure2010

*Dataset: An example time series for the EmiStatR package*

---

## Description

This dataset is a list that contains a data.frame with two columns: time [y-m-d h:m:s] and precipitation depth, P [mm]. The dataset correspond to measurements for 2010 with time steps of 10 minutes at rain gauge station, Esch-sur-Sure, located close to the sub-catchment of the combined sewer overflow chamber at Goesdorf, Grand-Duchy of Luxembourg.

## Usage

```
data("Esch_Sure2010")
```

## Format

A data frame with 52560 observations on the following 2 variables.

time a POSIXct vector

‘P [mm]’ a numeric vector

## Source

<https://www.agrimeteo.lu>

## Examples

```
data(Esch_Sure2010)
```

```
plot(Esch_Sure2010[,2], col="blue", typ="l", xlab = "time", ylab = "Precipitation [mm]")
```

---

inf	<i>Dataset: An example of general input for infiltration characteristics in the study region.</i>
-----	---

---

### Description

This dataset is an example of general input for infiltration characteristics in the sewer system for the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

### Usage

```
data("inf")
```

### Format

A list of 3 elements.

qf: numeric, infiltration water inflow [l/(s\*ha)];

CODf: numeric, chemical oxygen demand (COD) infiltration water pollution per capita (PE) load per day [g/(PE\*day)];

NH4f: numeric, ammonium (NH4) infiltration water pollution per capita (PE) load per day[g/(PE\*day)];

### Examples

```
data("inf")
```

```
str(inf)
```

---

input-class	<i>Class "input"</i>
-------------	----------------------

---

### Description

The class provides a container for inputs required to invoke EmiStatR method.

### Objects from the Class

Objects can be created by calls of the form `input()` or `new("input")`.

**Slots**

- spatial:** Object of class "numeric", 0 (default ) for non-spatial input, 1 for spatial input (not implemented).
- zero:** Object of class "numeric", approximation to zero value. Default 1E-5.
- folder:** Object of class "character", path for the output. Default getwd()
- cores:** Object of class "numeric", number of CPU cores to be used in parallel computing. If cores = 0 no parallel computation is done. Default 1.
- ww:** Object of class "list". This list contains three numeric elements for the wastewater characteristics. First element *qs*, individual water consumption of households [l/(PE d)]. Second element *CODs*, sewage pollution - COD concentration [g/(PE d)]. Third element *NH4s*, sewage pollution - NH4 concentration [g/(PE d)].
- inf:** Object of class "list". This list contains three numeric elements for infiltration water characteristics. First element *qf*, infiltration water inflow [l/(s ha)]. Second element *CODf*, infiltration water pollution - COD concentration [g/(PE d)]. Third element *NH4f*, infiltration water pollution - NH4 concentration [g/(PE d)].
- rw:** Object of class "list". This list contains three elements for rainwater characteristics. First element *CODr* (numeric), rainwater pollution - COD concentration [mg/l]. Second element *NH4r* (numeric), rainwater pollution - NH4 concentration [mg/l]. Third element *stat* (character), name of the rain measurement station.
- P1:** Object of class "data.frame" with two columns named 1. "time" for specifying the date and time in format YYYY-m-d HH:MM:SS, 2. "P [mm]" for specifying the depth values of the rainfall time series in millimeters. Optionally, instead of rainfall depth values can be provided values of direct runoff in cubic meters entering in the system. If runoff values are provided then the second column containing these values should be named as "Runoff\_Volume" or "runoff\_volume", otherwise this column is treated as rainfall depth.
- st:** Object of class "list". This object contains *n* lists, where *n* is the number of structures to simulate. Every list should contain 18 elements: *id*, numeric, identification number [-]; *ns*, character, name of the structure [-]; *nm*, character, name of the municipality [-]; *nc*, character, name of the catchment [-]; *numc*, numeric, number of the catchment [-]; *use*, character, use of the soil [-]; *Atotal*, numeric, total area [ha]; *Aimp*, numeric, impervious area [ha]; *tfs*, numeric, time flow structure [time step]; *pe*, numeric, population equivalent [PE]; *V*, numeric, volume [m<sup>3</sup>]; *lev2vol*, list of 2, *lev* and *vol*, defining the curve *lev* (level [m]) to *vol* (volume [m<sup>3</sup>]); *Qd*, numeric, maximum throttled outflow [l/s]; *Dd*, numeric, orifice diameter [m]; *Cd*, numeric, orifice coefficient [-]; *Cimp*, numeric, coefficient for impervious area [-]; and *Cper*, numeric, coefficient for pervious area [-].
- pe.ts.file:** Object of class "character" with the path and file name of the comma separated value (csv) file that contains the montly (seasonal) factors for population equivalent (*pe*). The first column of this file should be "time" in format "Y-m-d H:M:S" and should span for the entire length of the desired time series. The second column should contain the population equivalent as numeric class for the specified time, i.e. the desired *pe* time series with daily, weekly and monthly factors already applied. Default empty string (""). If not empty string is defined then *pe.daily.file*, *pe.weekly*, *pe.seasonal* are omitted.
- pe.daily.file:** Object of class "data.frame" that contains the daily factors for population equivalent. The first column should be the time in format "H:M:S" and should span for 24 hours. The second column should contain the factors as numeric class for the specified time. These factors must average to 1.

- pe.weekly:** Object of class "list" that contains the factors for population equivalent per day of the week with 7 elements called "mon" for Monday, "tue" for Tuesday, "wed" for Wednesday, "thu" for Thursday, "fri" for Friday, "sat" for Saturday, and "sun" for Sunday. These factors must average to 1.
- pe.seasonal:** Object of class "list" that contains the factors for population equivalent per month of the year with 12 elements called with the three first lower case letters of the month from "jan" for January to "dec" for December. These factors must average to 1.
- qs.ts.file:** Object of class "character" with the path and file name of the comma separated value (csv) file that contains the monthly (seasonal) factors for water consumption (qs). The first column of this file should be "time" in format "Y-m-d H:M:S" and should span for the entire length of the desired time series. The second column should contain the population equivalent as numeric class for the specified time, i.e. the desired qs time series with daily, weekly and monthly factors already applied. Default empty string (""). If not empty string is defined then `pe.daily.file`, `pe.weekly`, `pe.seasonal` are omitted.
- qs.daily.file:** Object of class "character" with the path and file name of the comma separated value (csv) file that contains the daily factors for water consumption. The first column of this file should be the time in format "H:M:S" and should span for 24 hours. The second column should contain the factors as numeric class for the specified time. These factors must average to 1.
- qs.weekly:** Object of class "list" that contains the factors for water consumption per day of the week with 7 elements called "mon" for Monday, "tue" for Tuesday, "wed" for Wednesday, "thu" for Thursday, "fri" for Friday, "sat" for Saturday, and "sun" for Sunday. These factors must average to 1.
- qs.seasonal:** Object of class "list" that contains the factors for water consumption per month of the year with 12 elements called with the three first lower case letters of the month from "jan" for January to "dec" for December. These factors must average to 1.
- export:** Object of class "numeric". If 1 (default) then the results are saved in folderOutput. Set to 0 for not writing in output files.

## Methods

**EmiStatR** signature(x = "input"): execute EmiStatR function

## Author(s)

J.A. Torres-Matallana

## Examples

```
## loading EmiStatR
library("EmiStatR")
showClass("input")

## running EmiStatR with user defined input
data("Esch_Sure2010")
P1 <- Esch_Sure2010[1:1000,] # selecting just the first 1,000 rows
station <- "Esch-sur-Sure"
```

```

# defining estructures E1
E1 <- list(id = 1, ns = "FBH Goesdorf", nm = "Goesdorf", nc = "Obersauer", numc = 1,
          use = "R/I", Atotal = 36, Aimp = 25.2, Cimp = 0.80, Cper = 0.30,
          tfS = 1, pe = 650, Qd = 5,
          Dd = 0.15, Cd = 0.18, V = 190, lev.ini = 0.10,
          lev2vol = list(lev = c(.06, 1.10, 1.30, 3.30),
                        vol = c(0.5, 31, 45, 190))
          )

# defining Input objet
input.user <- input(spatial = 0, zero = 1e-5, folder = getwd(),
                  cores = 1,
                  ww = list(qs = 150, CODs = 104, NH4s = 4.7),
                  inf = list(qf= 0.04, CODf = 0, NH4f =0),
                  rw = list(CODr = 0, NH4r = 0, stat = station),
                  P1 = P1, st = list(E1=E1), export = 0)

str(input.user)

# invoking EmiStatR
sim <- EmiStatR(input.user)

## a visualisation example
dev.new()
par(mfrow=c(2,2), oma = c(0,0,2,0))
plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[3]], typ="l", col="blue",
     xlab = "time", ylab = colnames(sim[[1]][[1]])[3], main = "Precipitation")
plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[10]], typ="l", col="blue",
     xlab = "time", ylab = colnames(sim[[1]][[1]])[10], main = "CSO, volume")
plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[13]], typ="l", col="blue",
     xlab = "time", ylab = colnames(sim[[1]][[1]])[13], main = "CSO, COD concentration")
plot(x=sim[[1]][[1]][[2]], y=sim[[1]][[1]][[14]], typ="l", col="blue",
     xlab = "time", ylab = colnames(sim[[1]][[1]])[14], main = "CSO, NH4 concentration")
mtext(paste("Structure", sim[[1]][[3]][[1]]), outer=TRUE, cex = 1.5)

```

---

IsReg

*Wrapping Function for Function "is.regular" from "zoo" package for "data.frame" objects*


---

## Description

"IsReg" is a wrapping Function for Function "is.regular" from "zoo" package. Given a "data.frame" object it is converted into a "xts" object, while the regularity of the object is checked. The first column of the "data.frame" should contain a character string vector to be converted via as.POSIXct accordingly with the date format (format) and time zone (tz).

## Usage

```
IsReg(data, format, tz)
```



## Arguments

data	an object of class <code>data.frame</code> containing in its first column a character string vector to be converted via <code>as.POSIXct</code> into a date vector accordingly with the date format ( <code>format</code> ) and time zone ( <code>tz</code> ) defined
format	character string giving a date-time format as used by <code>strptime</code> .
tz	a time zone specification to be used for the conversion, if one is required. System-specific, but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

## Details

"IsReg" calls the `as.POSIXct` function from base package to convert an object to one of the two classes used to represent date/times (calendar dates plus time to the nearest second). More details can be found in the "is.regular" function of the "zoo" package.

## Value

Object of class "list". This object contains 2 elements, the first one contains a character string "\_TSregular" if the xts object created is strict regular, or "\_TSirregular" if it is strict irregular. More details can be found in the "is.regular" function of the "zoo" package.

## Author(s)

J.A. Torres-Matallana

## Examples

```
library(EmiStatR)
data("P1")

class(P1)
head(P1)

ts <- IsReg(data = P1, format = "%Y-%m-%d %H:%M:%S", tz = "UTC")
str(ts)

ts[[1]]

head(ts[[2]]); tail(ts[[2]])

plot(ts[[2]], ylab = "Precipitation [mm]")
```

---

Level2Volume	<i>Function for linear interpolation given the relationship of two variables</i>
--------------	--

---

**Description**

Given a relationship between two variables (e.g. level and volume), this function interpolates the corresponding second variable (e.g. volume) for a known value of the first variable (e.g. level). This function is suitable to represent e.g. the dynamics of water storage in a combined sewer overflow chamber.

**Usage**

```
Level2Volume(lev, lev2vol)
```

**Arguments**

lev	A numeric object that represents the known variable e.g. the level in a storage chamber.
lev2vol	A list of two elements. The first element is a vector named "lev" that contains the interpolation steps of the first variable (e.g. level). The second element is a vector that contains the interpolation steps of the second variable (e.g. volume).

**Details**

The function uses the `approx` function from the `stats` package with "yleft" argument equal to the minimum value of the second variable and "yright" argument equal to the maximum value of the second variable.

**Value**

A numeric object with one element representing the interpolated value of the second variable (e.g. volume).

**Author(s)**

J.A. Torres-Matallana; K. Klepiszewski; G. Schutz.

**Examples**

```
library(EmiStatR)

# definition of relationship level to volume
lev2vol <- list(lev = c(.06, 1.10, 1.30, 3.30), vol = c(0, 31, 45, 200))

interpolated_volume <- Level2Volume(lev=c(0, .25, 1.25, 2.25, 4.25), lev2vol = lev2vol)
interpolated_volume
```

---

P1

*Dataset: An example of input time series for the EmiStatR package*

---

### Description

This dataset is a list that contains a data.frame with two columns: time [y-m-d h:m:s] and precipitation depth, P [mm]. The dataset correspond to measurements for January 2016 with time steps of 10 minutes at rain gauge station, Dahl, located close to the sub-catchment of the combined sewer overflow chamber at Goesdorf, Grand-Duchy of Luxembourg.

### Usage

```
data("P1")
```

### Format

A data frame with 4464 observations on the following 2 variables.

time a POSIXct vector  
'P [mm]' a numeric vector

### Source

<https://www.agrimeteo.lu>

### Examples

```
data("P1")  
  
plot(P1[,1], P1[,2], col="blue", typ="l", xlab = "time", ylab = "Precipitation [mm]")
```

---

P1\_20111216

*Dataset: An example of input time series for the EmiStatR package*

---

### Description

This dataset is a data.frame with four columns: time [y-m-d h:m:s] and precipitation depth, P [mm] for Dahl, Eshc-sur-Sure and Eschdorf rain gauge stations. The dataset correspond to measurements for December 16th 2011 for a 10-hour event recorded with time steps of 10 minutes. The three rain gauge stations are located close to the sub-catchment of the combined sewer overflow chamber at Goesdorf, Grand-Duchy of Luxembourg.

### Usage

```
data("P1_20111216")
```

**Format**

A data frame with 61 observations on the following 4 variables.

time a POSIXct vector

Dahl a numeric vector for rainfall depth in millimeters at Dahl rain gauge station

Esch-Sure a numeric vector for rainfall depth in millimeters at Esch-sur-Sure rain gauge station

Eschdorf a numeric vector for rainfall depth in millimeters at Eschdorf rain gauge station

**Source**

<https://www.agrimeteo.lu>

**Examples**

```
data("P1_20111216")
```

```
plot(P1_20111216[,1], P1_20111216[,2], col="blue", typ="l",
     xlab = "time", ylab = "Precipitation at Dahl [mm]")
```

```
plot(P1_20111216[,1], P1_20111216[,3], col="blue", typ="l",
     xlab = "time", ylab = "Precipitation at Esch-sur-Sure [mm]")
```

```
plot(P1_20111216[,1], P1_20111216[,4], col="blue", typ="l",
     xlab = "time", ylab = "Precipitation at Eschdorf [mm]")
```

---

pe\_factor

*Dataset: An example of general input for the daily population equivalent factors in the study region.*

---

**Description**

This dataset is an example of general input for the daily population equivalent (pe) factors in the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

**Usage**

```
data("pe_factor")
```

**Format**

A data.frame with 12 observations of 2 variables:

time: character, time at which the factors are reported [HH:MM:SS];

pe\_factor: numeric, the corresponding factors for population equivalents [-];

**Examples**

```
data("pe_factor")  
  
pe_factor
```

---

qs_factor	<i>Dataset: An example of general input for the daily water consumption factors in the study region.</i>
-----------	--

---

**Description**

This dataset is an example of general input for the daily water consumption (qs) factors in the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

**Usage**

```
data("qs_factor")
```

**Format**

A data.frame with 12 observations of 2 variables:

time: character, time at which the factors are reported [HH:MM:SS];  
qs\_factor: numeric, the corresponding factors for water consumption [-];

**Examples**

```
data("qs_factor")  
  
qs_factor
```

---

qs_factor_ATV	<i>Dataset: An example of general input for the daily water consumption factors (ATV German guideline) in the study region.</i>
---------------	---

---

**Description**

This dataset is an example of general input for the daily water consumption (qs) factors according to the german guideline ATV-A134 applied in the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

**Usage**

```
data("qs_factor")
```

**Format**

A data.frame with 12 observations of 2 variables:

`time`: character, time at which the factors are reported [HH:MM:SS];

`ATV.A134.Qf.m3_h`: numeric, the corresponding factors for water consumption according to the German guideline ATV-A134 [-];

**Examples**

```
data("qs_factor_ATV")
```

```
qs_factor_ATV
```

---

rw

*Dataset: An example of general input for rainwater characteristics in the study region.*

---

**Description**

This dataset is an example of general input for rainwater characteristics in the sewer system for the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

**Usage**

```
data("rw")
```

**Format**

A list of 6 elements:

`depth`: numeric, total rainfall amount of time series P1 [mm];

`pDur`: numeric, total rainfall duration of time series P1 [min];

`CODr`: numeric, rainwater chemical oxygen demand (COD) concentration [mg/l];

`NH4r`: numeric, rainwater ammonium (NH4) concentration [mg/l];

`stat`: character, raingauge station name for time series P1 [-];

`delta.t`: numeric, delta time for time series P1 [min];

**Examples**

```
data("rw")
```

```
str(rw)
```

---

Volume2Level	<i>Function for linear interpolation given the relationship of two variables</i>
--------------	--

---

### Description

Given a relationship between two variables (e.g. volume and level), this function interpolates the corresponding second variable (e.g. level) for a known value of the first variable (e.g. volume). This function is suitable to represent e.g. the dynamics of water storage in a combined sewer overflow chamber.

### Usage

```
Volume2Level(vol, lev2vol)
```

### Arguments

vol	A numeric object that represents the known variable e.g. the volume in a storage chamber.
lev2vol	A list of two elements. The first element is a vector named "lev" that contains the interpolation steps of the first variable (e.g. level). The second element is a vector that contains the interpolation steps of the second variable (e.g. volume).

### Author(s)

J.A. Torres-Matallana, G. Schutz

### Examples

```
library(EmiStatR)

# definition of relationship level to volume
lev2vol <- list(lev = c(.06, 1.10, 1.30, 3.30), vol = c(0, 31, 45, 200))

interpolated_level <- Volume2Level(vol=c(0, 25, 41, 190, 220), lev2vol = lev2vol)
interpolated_level
```

---

ww	<i>Dataset: An example of general input for wastewater characteristics in the study region.</i>
----	---

---

### Description

This dataset is an example of general input for wastewater characteristics in the sewer system for the study region (Haute-Sure catchment, Grand-Duchy of Luxembourg).

**Usage**

```
data("ww")
```

**Format**

A list of 3 elements.

qs: numeric, infiltration inflow [ $l/(s*ha)$ ];

CODs: numeric, chemical oxygen demand (COD) sewage pollution per capita (PE) load per day  
[ $g/(PE*day)$ ];

NH4s: numeric, ammonium (NH<sub>4</sub>) sewage pollution per capita (PE) load per day [ $g/(PE*day)$ ];

**Examples**

```
data("ww")
```

```
str(ww)
```



# Index

- \* **Aggregation**
  - Agg, [3](#)
- \* **Agg**
  - Agg, [3](#)
- \* **CInp2TS**
  - CInp2TS, [4](#)
- \* **Delay**
  - Delay, [6](#)
- \* **EmiStatR**
  - EmiStatR-methods, [10](#)
- \* **Is Regular**
  - IsReg, [16](#)
- \* **IsReg**
  - IsReg, [16](#)
- \* **Level to Volume**
  - Level2Volume, [18](#)
- \* **Level2Volume**
  - Level2Volume, [18](#)
- \* **Temporal aggregation**
  - Agg, [3](#)
- \* **Volume2Level**
  - Volume2Level, [23](#)
- \* **classes**
  - input-class, [13](#)
- \* **datasets**
  - E1, [7](#)
  - E2, [8](#)
  - E3, [9](#)
  - Esch\_Sure2010, [12](#)
  - inf, [13](#)
  - P1, [19](#)
  - P1\_20111216, [19](#)
  - pe\_factor, [20](#)
  - qs\_factor, [21](#)
  - qs\_factor\_ATV, [21](#)
  - rw, [22](#)
  - ww, [23](#)
- \* **input**
  - EmiStatR-methods, [10](#)
- \* **methods**
  - EmiStatR-methods, [10](#)
- \* **package**
  - EmiStatR-package, [2](#)
- Agg, [3](#)
- CInp2TS, [4](#)
- Delay, [6](#)
- E1, [7](#)
- E2, [8](#)
- E3, [9](#)
- EmiStatR, [3](#)
- EmiStatR (EmiStatR-methods), [10](#)
- EmiStatR, input-method (input-class), [13](#)
- EmiStatR-methods, [10](#)
- EmiStatR-package, [2](#)
- Esch\_Sure2010, [12](#)
- inf, [13](#)
- input (input-class), [13](#)
- input-class, [13](#)
- IsReg, [16](#)
- Level2Volume, [18](#)
- P1, [19](#)
- P1\_20111216, [19](#)
- pe\_factor, [20](#)
- qs\_factor, [21](#)
- qs\_factor\_ATV, [21](#)
- rw, [22](#)
- Volume2Level, [23](#)
- ww, [23](#)