Package 'elementR'

May 3, 2018

Type Package

Title An Framework for Reducing Elemental LAICPMS Data from Solid Structures

Version 1.3.6

Date 2018-05-03

Author Charlotte Sirot, Francois Guilhaumon

Maintainer Charlotte Sirot <charlott.sirot@gmail.com>

Description Aims to facilitate the reduction of elemental microchemistry data from solid-phase LAICPMS analysis (laser ablation inductive coupled plasma mass spectrometry). The 'elementR' package provides a reactive and user friendly interface (based on a 'shiny' application) and a set of 'R6' classes for conducting all steps needed for an optimal data reduction while leaving maximum control for user.

Repository CRAN

Depends R (>= 3.2.3)

Imports gdata, shiny,devtools, shinyjs, gnumeric, R6, shinydashboard, abind, stringr, lmtest, tcltk,tcltk2, reader, readODS, readxl, EnvStats, outliers, zoo, colourpicker, stats, graphics, utils, httpuv

License GPL (>= 2)

Suggests testthat

URL https://github.com/charlottesirot/elementR

BugReports https://github.com/charlottesirot/elementR/issues

NeedsCompilation no

Date/Publication 2018-05-03 14:08:24 UTC

R topics documented:

`convertingReplicate` *convertingReplicate*

Description

Convert and export the elemental concentrations (relative to an internal standard) from files of a repository from ppt/ppt to Mol/Mol and vice versa while keeping the repository structure.

Usage

```
convertingReplicate()
```

Details

By running `convertingReplicate()`, the user has access to an interface through its web browser opened automatlically as soon as the function is launched. This interface allows to upload the repository of the files to be converted, to choose the unit of conversion and to export the final converted data. Note that `convertingReplicate()` will convert the whole data frame uploaded except the first column that is usually dedicted to the time (transect mode) or to the names of the replicate (spot mode) according to `runElementR()` method.

See Also

`convertMol_to_PPM`. `convertPPM_to_Mol`.

convertMol_to_PPM *convertMol_to_PPM*

Description

Convert the elemental concentrations (relative to an internal standard) of a data frame from Mol/Mol to ppm/ppm

Usage

```
convertMol_to_PPM(dat, AtomicMass, InternStand)
```

Arguments

dat	a data frame of elemental concentrations (relative to an internal standard) in Mol/Mol
AtomicMass	a matrix describing the atomic weight of the elements included in the session
InternStand	the name of the internal standard used in the calculation of the concentrations

Details

A matrix describing the atomic weight of the element is included in the present package (AtomicMass.csv). Note that `convertMol_to_PPM()` will convert the whole data frame uploaded except the first column that is usually dedicted to the time (transect mode) or to the names of the replicate (spot mode) according to `runElementR()` method.

See Also

convertingReplicate. convertPPM_to_Mol.

Examples

```
## Convert the file Example1_replicate1.csv included in the package from Mol/Mol into ppm/ppm

# indicate the path and read the file to be converted
filePath <- system.file("Example_conversion/Ex1/Example1_Replicate1.csv", package="elementR")

dat <- readData(filePath, sep = ",", dec = ".")

# indicate the path and read the file containing the atomic weight of the elements
AtomWeightPath <- system.file("AtomicMass.csv", package="elementR")

AtomicMass <- readData(AtomWeightPath, sep = ",", dec = ".")

# set the internal standard
InternStand <- "Ca43"

standard <- convertMol_to_PPM(dat, AtomicMass, InternStand)
```

```
## Display the converted data
standard
```

---

convertPPM_to_Mol *convertPPM_to_Mol*

Description

Convert the elemental concentrations (relative to an internal standard) of a data frame from ppm/ppm to Mol/Mol

Usage

```
convertPPM_to_Mol(dat, AtomicMass, InternStand)
```

Arguments

dat	a data frame of elemental concentrations (relative to an internal standard) in Mol/Mol
AtomicMass	a matrix describing the atomic weight of the elements included in the session
InternStand	the name of the internal standard used in the calculation of the concentrations

Details

A matrix describing the atomic weight of the element is included in the present package (AtomicMass.csv). Note that `convertPPM_to_Mol()` will convert the whole data frame uploaded except the first column that is usually dedicted to the time (transect mode) or to the names of the replicate (spot mode) according to `runElementR()` method.

See Also

convertingReplicate. convertMol_to_PPM.

Examples

```
## Convert the file Example1_replicate1.csv included in the package from ppm/ppm into Mol/Mol

# indicate the path and read the file to be converted
filePath <- system.file("Example_conversion/Example2_Replicate1.csv", package="elementR")

dat <- readData(filePath, sep = ",", dec = ".")

# indicate the path and read the file containing the atomic weight of the elements
AtomWeightPath <- system.file("AtomicMass.csv", package="elementR")

AtomicMass <- readData(AtomWeightPath, sep = ",", dec = ".")
```

```
# set the internal standard
InternStand <- "Ca43"

standard <- convertPPM_to_Mol(dat, AtomicMass, InternStand)

## Display the converted data
standard
```

---

elementR *A Shiny Application for Reducing Elemental LA-ICPMS Data from Solid Structures*

Description

Aims to facilitate the reduction of elemental microchemistry data from solid-phase LA-ICPMS analysis (laser ablation inductive coupled plasma mass spectrometry). The elementR package provides a reactive and user friendly interface for conducting all steps needed for an optimal data reduction while leaving maximum control for user.

Author

Charlotte Sirot, Francois Guilhaumon, Franck Ferraton, Audrey Darnaude, Jacques Panfili, Amber Child

Maintainer

Charlotte Sirot <charlott.sirot@gmail.com>

References

Calculations of elementR procedures are based on a consensus of the literature:

- Elsdon & Gillanders. Interactive effects of temperature and salinity on otolith chemistry: challenges for determining environmental histories of fish. Can. J. Fish. Aquat. Sci. Vol. 59, 2002.
- Fowler et al. Experimental assessment of the effect of temperature and salinity on elemental composition of otolith using laser ablation ICPMS. J. Fish. Aquat. Sci. Vol. 52, 1995.
- Milton & Chenery. The effect of otolith storage methods on the concentrations of elements detected by laser-ablation ICPMS. J. of Fish Biology, Vol. 53, 1998.
- Thorrold et al. 1998. Accurate classification of juvenile weakfish Cynoscion regalis to estuarine nursery areas based on chemical signatures in otoliths. Marine Ecology Press Series, Vol. 173, 1998.

Examples

```
#runElementR()
```

elementR_data *Object elementR_data*

Description

The R6Class object elementR_data contains the main information needed for the filtration of a single replicate (sample or standard).

Usage

```
elementR_data
```

Format

An R6Class generator object

Details

When runElementR is running and as soon as a project is loaded, an elementR_data is automatically created for each replicate included in the session (standard and sample). Each of these objects contains the basic information regarding the considered replicate (name, path and raw data) and is filled by the intermediate and final data as user proceeds to the filtration procedure.

Fields

name A character string corresponding to the name of the considered replicate

data A matrix corresponding to the raw data of the considered replicate

fPath A character string corresponding the path of the raw data

bins A numerical value corresponding to the time at which end the blank values

plat A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

dataBlank A matrix corresponding to the blank data

dataPlateau A matrix corresponding to the plateau data

dataSuppBlank A matrix corresponding to the data obtained by substracting the averaged blank value (here, BlankAverarge) from the dataPlateau

dataSupLOD A matrix of data corresponding to the values of dataSuppBlank up to the limit of detection (here LOD)

dataNorm A matrix of data corresponding to the values of dataSupLOD normalized by the chemical element chosen as internal standard (here, elemstand)

`elemstand` A character string corresponding to the name of the chemical element chosen as internal standard

`LOD` A vector of numerical values corresponding to the limit of detection for each chemical element of the considered replicate

`BlankAverarge` A vector of numerical values corresponding to the averaged blank value for each chemical element of the considered replicate

`remplaceValue` A character string corresponding to the value replacing the `dataSuppBlank` below the limit of detection

Methods

`initialize(filePath, sep , dec)` Aim: Create and set basic information of the considered replicate; Argument: filePath = the path of the considered replicate data, dec = the decimal system of the data, sep = the separator character of the data; Output: an R6Class `elementR_data` object

`setBins(bins)` Aim: set `bins`; Argument: bins = A numerical value corresponding to the time at which end the blank values

`setPlat(plat)` Aim: set `plat`; Argument: plat = A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

`setDataBlanc(bins)` Aim: set `dataBlank`; Argument: bins = A numerical value corresponding to the time at which end the blank values

`setDataPlateau(plat)` Aim: set `dataPlateau`; Argument: plat = A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

`setDataSuppBlank(bins,plat)` Aim: set `dataSuppBlank`; Arguments: bins = A numerical value corresponding to the time at which end the blank values, plat = A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

`setDataSupLOD(bins,plat)` Aim: set `dataSupLOD`; Arguments: bins = A numerical value corresponding to the time at which end the blank values, plat = A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

`setDataNorm(bins,plat)` Aim: set `dataNorm`; Arguments: bins = A numerical value corresponding to the time at which end the blank values, plat = A vector containing two numerical values corresponding respectively to the time at which begin and end the plateau values

`reset()` Aim: replace `dataConcCorr` by NA

`OutlierDetectTietjen(x, nbOutliers)` Aim: return the place of the outlier of a vector according to Tietjen and outlier methods; Arguments: x = a vector, nbOutliers = the number of suspected outliers; Outputs: a vector of the position of the outlier in the vector

`outlierDetection(dat, method, nbOutliers)` Aim: return the place of the outlier of a vector; Arguments: dat = a vector, method = the method used for the detection ("Tietjen.Moore Test", "SD criterion", "Rosner's test"), nbOutliers = the number of suspected outliers; Outputs: a vector of the position of the outlier in the vector

`detectOutlierMatrix(dat, method, nbOutliers)` Aim: return the place of the outlier for each column of a matrix; Arguments: dat = a matrix, method = the method used for the detection ("Tietjen.Moore Test", "SD criterion", "Rosner's test"), nbOutliers = the number of suspected outliers; Outputs: a list of vector corresponding to the position of the outlier in each column of the matrix

`outlierReplace(dat, outlierList, rempl)` Aim: replace the outliers value of a matrix by rempl; Arguments: dat = a matrix, a list showing the place of the outlier for each column, rempl: the value to replace if outliers

`is.possibleOutlier(dat)` Aim: check that the vector fits with the needs for outlier detection (length of data > 30 and not all the same); Arguments: dat = a vector of data; OUtputs: TRUE: the investigated vector meets the conditions, FALSE: the investigated vector does not meet the conditions

See Also

`elementR_sample`. `elementR_standard`.

Examples

```
## create a new elementR_data object based on the "filePath"
## from a file containing data (accepted format of data: .csv, .ods, .xls, .xlsx)

filePath <- system.file("Example_Session/standards/Stand3.xls", package="elementR")

standard <- elementR_data$new(filePath)

## Display the raw data

standard$data
```

---

elementR_project *Object elementR_project*

Description

The `R6Class` object `elementR_project` contains all the information needed for running an elementR session

Usage

```
elementR_project
```

Format

An R6Class generator object

Details

The elementR_project structure allows to organized data in a session framework, faciliting therefore numerous major functionalities: handling as many standard replicates as wanted, machine drift verification and correction, sample replicate realignment and averaging. Moreover, this object can be easily exported, allowing user to re-open it later for finishing or editing final results.

Fields

name A character string corresponding to the name of the project

folderPath A character string corresponding to the path of the project

standardsPath A character string corresponding to the path of the standard folder

standardsFiles A vector containing the names of each standard file

standards A list containing the elementR_repStandard of each type of standard

samplesPath A character string corresponding to the path of the sample folder

samplesFiles A vector containing the names of each sample file

samples A list containing the elementR_repSample of each sample

EtalonPath A character string corresponding to the path of the calibration file

EtalonData A matrix corresponding to the calibration data

listeElem A vector containing the names of the chemical elements included in the project

flag_stand A vector indicating which standards have been filtered

flag_Sample A vector indicating which samples have been filtered

flagRealign A list vectors indicating which samples have been realigned or averaged

standardRank A vector corresponding to the standard rank in ICPMS analysis

sampleRank A vector corresponding to the sample rank in ICPMS analysis

elementChecking A list indicating the number and the location of the error(s) of structure within data included in the project

errorSession A numerical value indicating the non numeric error(s) within data included in the project

regressionModel A matrix summarizing, for each chemical element, the parameters of the linear regression corresponding to the machine drift

machineCorrection A vector summarizing the chemical element(s) to correct from machine drift

flagMachineCorrection A numerical value indicating the validation of the machine correction step

nbCalib A vector corresponding to the number of standard values available for each chemical element to proceed the linear regression

elemStand A character string indicating the chemical element considered as internal standard (by default = Ca)

summarySettings A matrix summarizing all the parameters set by user for each replicate (sample and standard)

ChoiceUserCorr A logical value corresponding to the choice of the user to correct or no the session based on the first step of configuration

R2Threshold the threshold to switch the machine drift correction from a linear to a neighbor correction

Methods

set_summarySettings(name, rank, bins, plat1, plat2, average, LOD) Aim: set summarySettings; Arguments: name = a character string corresponding to the name of the replicate to set, rank= its rank in ICPMS analysis, bins = a numerical value corresponding to the time at which end the blank values, plat1 = a numerical value corresponding to the time at which begin the plateau values, plat2 = a numerical value corresponding to the time at which end the plateau values, average = a vector corresponding to the blank averaged value (here, BlankAverarge) for each chemical element of the considered replicate, LOD = a vector corresponding to the limit of detection (here, LOD) for each chemical element of the considered replicate

is.integer0(x) Aim: test the integer(0); Arguments: x = a vector to test; Outputs: TRUE or FALSE

closest(x,y) Aim: find the nearest value among a vector of numerical data; Arguments: x = a vector of numerical values, y = the investigated value; Output: val = a list of two values: the nearest value and its place within the vector

PlotIC(name, Mean,SD, coord, lengthSeg, xlim, ylim, type = "p", xlab, ylab) Aim: plot mean +/- SD; Arguments: name = a vector of the names to display on xaxis, Mean = a vector of mean, SD = a vector of SD, coord = a vector of coordonnates to place xticks, lengthSeg = a numeric value cooresponding to the length of the top segment of the SD bar, xlim & ylim = the limits of plots, xlab & ylab = the labels of axis

setEtalon(x, sep, dec) Aim: define EtalonPath and EtalonData and check the validity of their data structure; Arguments: x = a character string corresponding to the path of the calibration file, dec = the decimal system of the data, sep = the separator character of the data

setflagMachineCorrection(x) Aim: set flagMachineCorrection; Arguments: x = the numerical value to set

`NonNumericCheck(data, col)` Aim: check non numeric characters of data; Arguments: data = a dataframe or a matrix, col = a vector of numerical values corresponding to the column(s) to investigate; Output: errB = a numerical value corresponding to the number of cells containing non numeric characters

`setflagStand(place, value)` Aim: set `flag_stand`; Arguments: place = a numerical value corresponding to the considered replicate, value = the numerical value to set

`setflagSample(sample, replicate, value)` Aim: set `flag_Sample`; Arguments: sample = a numerical value corresponding to the considered sample, replicate = a numerical value corresponding to the considered replicate, value = the numerical value to set

`setCorrection(x)` Aim: set `machineCorrection`; Arguments: x = a vector indicating the chemical elements to correct from machine drift

`correction()` Aim: proceed to the linear regression on standards replicates and set `nbCalib` & `regressionModel`

`setRank(type, value)` Aim: set the order in which ICPMS runs each standard (`standardRank`) and sample (`sampleRank`) replicates; Arguments: type = a character string indicating the type of replicate standard ("standard") or sample ("sample"), value = a numerical value corresponding to the rank of the considered replicate

`set_flagRealign(replicate, type, value)` Aim: set `flagRealign`; Arguments: replicate = a numerical value corresponding to the number of the considered replicate, type = a character string indicating the raster or spot mode, value = the numerical value to set

`setElemStand(elem)` Aim: define `elemStand` and transmit this value to all `elementR_rep` and `elementR_data` objects inlcuded in the project; Arguments: elem = a character string corresponding to the element considered as intern standard

`initialize(folderPath, sep, dec)` Aim: create the project; Arguments: filePath = the path of the considered project, dec = the decimal system of the data, sep = the separator character of the data; Outputs: R6Class `elementR_project`

`set_ChoiceUserCorr(x)` Aim: information about the will of user to check or not the machine drift; Arguments: x = T (for checking machine drift), F (for not checking machine drift)

`setR2Threshold(x)` Aim: set `R2Threshold`; Arguments: x = a value between 0 and 1

`insert.at(a, pos, toInsert)` Aim: insert values in vectors; Arguments: a = a vector, pos = the position to insert, toInsert = a vector to insert

`detectPlateau(dat, col)` Aim: detection of the plateau limits of a matrix based on clustering methods and on the internal standard element; Arguments: dat = the data to proceed, col = the column used for the detection

`detectBlank(dat, col)` Aim: detection of the blank limits of a matrix based on the derivative value and on the internal standard element; Arguments: dat = the data to proceed, col = the column used for the detection

See Also

elementR_rep. elementR_data.

Examples

```
## create a new elementR_repStandard object based on the "filePath"
## from a folder containing sample replicate

filePath <- system.file("Example_Session", package="elementR")

exampleProject <- elementR_project$new(filePath)

## Display the raw data

exampleProject$samplesFiles
```

---

elementR_rep *Object elementR_rep*

Description

The R6Class object elementR_rep contains the main information needed for the filtration of a batch of replicates (standard or sample replicates).

Usage

```
elementR_rep
```

Format

An R6Class generator object

Details

When runElementR is running and as soon as a project is loaded, an elementR_rep is automatically created for each batch of replicates (i.e. each folder of standards or samples) included in the session. Each of these objects contains the basic information regarding the considered batch (name and path of the folder, the whole data of each replicates) and is filled by the intermediate and final data as user proceeds to the filtration procedure.

Fields

rep_name A character string corresponding to the name of the considered folder

rep_folderPath A character string corresponding to the path of the considered folder

rep_Files A vector containing the name of the files within the considered folder

`rep_data` A list containing the `elementR_data` corresponding to the replicates included in the considered folder

`rep_pas` A numerical value corresponding to the time between two consecutive analysis within data of the considered folder

`dec` The decimal system used in the data (either , or .)

`sep` The separator character used in the data

Methods

`setRep_pas()` Aim: set `rep_pas`

`initialize(filePath, sep, dec)` Aim: Create and set the basic information of the considered folder; Argument: filePath = the path of the considered folder, dec = the decimal system of the data, sep = the separator character of the data; Output: an R6Class `elementR_rep` object

See Also

`elementR_repStandard`. `elementR_repSample`.

Examples

```
## see elementR_repStandard or elementR_repSample as the creation of elementR_rep depends
## on the type of data created
```

---

elementR_repSample *object elementR_repSample*

Description

The `R6Class` object `elementR_repSample` contains the main information needed for the filtration of a batch of replicates from the same sample.

Usage

```
elementR_repSample
```

Format

An `R6Class` generator object

Details

As a subclass object, the `elementR_sample` object already contains all fields and methods from the `elementR_rep`. Moreover, it also contains items specifically designed for sample filtration.

Inheritance

The `elementR_repSample` object inherits from the `elementR_rep`.

Fields

`rep_type` A character string indicating the type of the considered batch (here, "sample")

`rep_type2` A character string corresponding to the processing mode of averaging ("raster" or "spot")

`rep_dataFiltre` A list containing the data to average of each replicate of the considered sample (`dataOutlierFree` for spot mode and `dataNorm` for raster mode)

`rep_dataFinalSpot` A matrix containing the average and the standard deviation per chemical element of the `dataOutlierFree` of the final replicates (i.e. chosen to be part of the final calculation)

`rep_dataIntermRaster` A list containing the realigned `dataNorm` of the final replicates (i.e. chosen to be part of the final calculation)

`rep_dataFinalRaster` A matrix corresponding to the average values of the data contained in `rep_dataIntermRaster`

`rep_autoCorrel` a vector which contains (1) laser diameter, (2) laser speed, (3) which point to keep

`rep_dataFinalRasterNonCorr` a matrix of the final data without correlated points

Methods

`setrep_type2(x)` Aim: set `rep_type2`; Arguments: x = a character string indicating spot or raster mode

`Realign2(data, pas)` Aim: Realign data; Arguments: data = a list of matrix corresponding to the data to realign, pas = the step of time between two consecutive analysis within data of the considered sample; Output: data = a list of matrix containing the realigned data

`setRep_dataFiltre(x)` Aim: set `rep_dataFiltre`; Arguments: x = a logical value corresponding to the choice of user to correct or not the machine drift

`setRep_dataFinalSpot(x)` Aim: set `rep_dataFinalSpot`; Arguments: x = the matrix to set

`intermStepSpot()` Aim: create and return an intermediate matrix containing the average and the standard deviation per chemical element for all sample replicates; Output: outputTab = a matrix with two lines corresponding to the average and the standard deviation per chemical element for all sample replicates

`intermStepRaster(decalage, input, outliers, replace)` Aim: create and return an intermediate matrix containing realigned data for all sample replicates; Inputs: decalage = a vector indicating the translation to operate, input = the data to realign, outliers = a list of outliers, replace = the value to replace in case of outlier, Output: outputList = a list of matrix containing realigned data for all sample replicates

`setRep_dataIntermRaster(x)` Aim: set `setRep_dataIntermRaster`; Arguments: x = the list of matrix to set

`setRep_dataFinalRaster()` Aim: set `rep_dataFinalRaster`

`create()` Aim: create and set the field `rep_data` by filling it with the `elementR_sample` objects corresponding to sample replicates included in this batch

`set_rep_autoCorrel(x)` Aim: set `rep_autoCorrel`, Input: x = the value to set

`set_rep_dataFinalRasterNonCorr()` Aim: set `rep_dataFinalRasterNonCorr`

`RealignCol(dat1, dat2, col, step)` Aim: realign two tables according to a chosen column (based on a convolution); Inputs: dat1 & dat2 = matrix to realign, col = the column to realign, step = the step between two consecutive analysis; Outputs: the realign data

`RealignColList(listRealig, col, step)` Aim: realign a list of matrix according to a chosen column (based on a convolution); Inputs: listRealig = a list of matrix to realign, col = the column to realign, step = the step between two consecutive analysis; Outputs: the realign data

`RealignAll(dat1, dat2, step)` Aim: realign a list of matrix according to all columns (based on a convolution); Inputs: dat1 & dat2 = matrix to realign, step = the step between two consecutive analysis; Outputs: the realign data

`RealignListAll(listRealig, step)` Aim: realign a list of matrix according to all columns (based on a convolution); Inputs: listRealig = a list of matrix to realign, step = the step between two consecutive analysis; Outputs: the realign data

See Also

`elementR_rep`. `elementR_repStandard`.

Examples

```
## create a new elementR_sample object based on the "filePath"
## from a folder containing sample replicate

filePath <- system.file("Example_Session/samples/Sample_1", package="elementR")

sampleBatch <- elementR_repSample$new(filePath)

## Display the data contained in this batch

sampleBatch$rep_data
```

elementR_repStandard *Object elementR_repStandard*

Description

The R6Class object elementR_repStandard contains the main information needed for the filtration of a batch of standard replicates.

Usage

delementR_repStandard

Format

An R6Class generator object

Details

As a subclass object, the elementR_repStandard object already contains all fields and methods from the elementR_rep. Moreover, it also contains items specifically designed for standard filtration.

Inheritance

The delementR_repStandard object inherits from the elementR_rep.

Fields

rep_type A character string indicating the type of the batch considered (here, "standard")

rep_dataFinale A matrix containing data_standFinalMean and data_standFinalSD for all standard replicates included in the considered batch

rep_dataFinaleMean A vector containing the average per chemical element of the rep_dataFinale

rep_dataFinaleSD A vector containing the standard deviation per chemical element of the rep_dataFinale

Methods

setrep_FinalMeanSD() Aim: define and set rep_dataFinaleMean and rep_dataFinaleSD

setRep_table(nelem) Aim: define and set rep_dataFinale; Arguments: nelem = a vector containing the names of the chemical elements to include in the rep_dataFinale

create() Aim: create and set rep_data by filling it with the elementR_standard objects corresponding to standard replicates included in this batch

See Also

elementR_rep. elementR_repSample.

Examples

```
## create a new elementR_repStandard object based on the "filePath"
## from a folder containing sample replicate

filePath <- system.file("Example_Session/standards", package="elementR")

standBatch <- elementR_repStandard$new(rep_folderPath = filePath)

## Display the files contained in this batch

standBatch$rep_Files
```

---

elementR_sample *Object elementR_sample*

Description

The R6Class object elementR_sample contains the main information needed for the filtration of a single sample replicate.

Usage

```
elementR_sample
```

Format

An R6Class generator object

Details

As a subclass object, the elementR_sample object already contains the whole fields and methods from the elementR_data. Moreover, it also contains items specifically designed for sample filtration.

Inheritance

The elementR_sample object inherits from the elementR_data

Fields

type A character string corresponding to the type of replicate (here, "sample")

dataConc A matrix corresponding to the dataNorm converted in concentration

dataConcCorr A matrix corresponding to the dataConc corrected (or not) from the machine drift

Methods

`setDataConc(bins, plat, calibFile, meanStand, rempl)` Aim: set `dataConc`; Arguments: bins = a numerical value corresponding to the time at which end the blank values, plat = a vector of two numerical values corresponding respectively to the time at which begin and end the plateau, calibFile = a matrix corresponding to the data of the calibration file, meanStand = a vector containing the averaged signal intensity per chemical element for all standard replicates of the running session, rempl = the value replacing data if below the limit of detection

`setDataConcCorr(bins, plat, name, calibFile, meanStand, rankSample, rankStandard, model, correction` Aim: set `dataConcCorr`; Arguments: bins = a numerical value corresponding to the time at which end the blank values, plat = a vector of two numerical values corresponding respectively to the time at which begin and end the plateau, name = a character string corresponding to the name of the sample replicates, calibFile = a matrix corresponding to the the calibration file, meanStand = a vector containing the averaged signal intensity per chemical element for all standard replicates of the running session, rankSample = a vector containing the rank of each sample in ICPMS analysis, rankStandard = a vector containing the rank of each standard in ICPMS analysis, correction = a vector indicating the chemical elements to correct from machine drift, model = a matrix containing the parameters of the linear regression corresponding to the machine drift for all chemical elements, threshold = the R2 threshold to consider that the model does not fit to a linear model

`renderData(curve)` Aim: render data without proceding to their calculation; Argument: curve = a character string corresponding to the type of data to render ("Blank" for calculate and/or render the `dataBlank`, "Raw" for `data`, "Plateau" for `dataPlateau`, "Blank removed" for `dataSuppBlank`, ">LOD" for `dataSupLOD`, "Normalized" for `dataNorm`, "Concentration" for `dataConc` and "Conc. corrected" for `dataConcCorr`); Output: a matrix of the required data

`getData(curve, bins, plat, name, calibFile, meanStand, rankSample, rankStandard, model, correction)` Aim: calculate and render the required data ; Arguments: curve = a character string corresponding to the type of data to calculate (for more details, see renderData arguments), bins = a numerical value corresponding to the time at which end the blank values, plat = a vector of two numerical values corresponding respectively to the time at which begin and end the plateau, name = a character string corresponding to the name of the sample replicates, calibFile = a matrix corresponding to the the calibration file, meanStand = a vector containing the averaged signal intensity per chemical element for all standard replicates of the running session, rankSample = a vector containing the rank of each sample in ICPMS analysis, rankStandard = a vector containing the rank of each standard in ICPMS analysis, correction = a vector indicating the chemical elements to correct from machine drift, model = a matrix containing the parameters of the linear regression corresponding to the machine drift for all chemical elements, threshold = the R2 threshold to consider that the model does not fit to a linear model

See Also

`elementR_data`. `elementR_standard`.

Examples

```
## create a new elementR_sample object based on the "filePath" from a file containing data
```

```
## replicate (accepted format of data: .csv, .ods, .xls, .xlsx)

filePath <- system.file("Example_Session/samples/Sample_1/Sample1_Rep1.csv", package="elementR")

sampleExample <- elementR_sample$new(filePath)

## Display the name of the object

sampleExample$name
```

`elementR_standard` *Object elementR_standard*

Description

The `R6Class` object `elementR_standard` contains the main information needed for the filtration of a single standard replicate.

Usage

```
elementR_standard
```

Format

An `R6Class` generator object

Details

As a subclass object, the `elementR_standard` object already contains all fields and methods from the `elementR_data`. Moreover, it also contains items specifically designed for standard filtration.

Inheritance

The `elementR_standard` object inherits from the `elementR_data`.

Fields

`type` A character string indicating the type of replicate (here, "standard")

`dataOutlierFree` A matrix corresponding to the `dataNorm` without abnomalities

`data_standFinalMean` A vector corresponding to the average of `dataOutlierFree` per chemical element

`data_standFinalSD` A vector corresponding to the standard deviation of `dataOutlierFree` per chemical element

Methods

`setDataOutlierFree(bins, plat, rempl, method, nbOutliers)` Aim: set `dataOutlierFree`; Arguments: bins = a numerical value corresponding to the time at which end the blank values, plat = a vector of two numerical values corresponding respectively to the time at which begin and end the plateau, rempl = value to replace outliers, method = the method used to detect outliers ("Tietjen.Moore Test", "SD criterion", "Rosner's test"), nbOutliers = nb of suspected outliers

`setdata_standFinal()` Aim: set `data_standFinalMean` and `data_standFinalSD`

`renderData(curve)` Aim: render data without proceeding to their calculation; Argument: curve = a character string corresponding to the type of data to render ("Blank" for `dataBlank`, "Raw" for `data`, "Plateau" for `dataPlateau`, "Blank removed" for `dataSuppBlank`, ">LOD" for `dataSupLOD`, "Normalized" for `dataNorm`, "Outliers free" for `dataOutlierFree`); Output: a matrix of the required data

`getData(curve, bins, plat, rempl, method, nbOutliers)` Aim: calculate and render the required data ; Arguments: curve = a character string corresponding to the type of data to render (for more details, see the curve argument of the renderData function), bins = a numerical value corresponding to the time at which end the blank values, plat = a vector of two numerical values corresponding respectively to the time at which begin and end the plateau, rempl = value to replace outliers, method = the method used to detect outliers ("Tietjen.Moore Test", "SD criterion", "Rosner's test"), nbOutliers = nb of suspected outliers; Output: a matrix of the required data

See Also

`elementR_data`. `elementR_sample`.

Examples

```
## create a new elementR_standard object based on the "filePath" from a file containing data

filePath <- system.file("Example_Session/standards/Stand1.csv", package="elementR")

standardExample <- elementR_standard$new(filePath)

## Display the raw data

standardExample$data
```

`readData` *readData*

Description

Read the content of an Excel (.xls and .xlsx), OpenOffice (.ods) and text (.csv) worksheet

Usage

```
readData(x, sep, dec)
```

Arguments

x	a character string corresponding to the path or name of the file to read
sep	a character string corresponding to the separator
dec	a character string corresponding to the decimal

Details

For the Excel and text format, readData reads by default the first worksheet of the file and the one called "data" for the OpenOffice format.

Examples

```
## Read data based on its path "filePath"

filePath <- system.file("Example_Session/standards/Stand1.csv", package="elementR")

readData(filePath, sep = ";", dec = ".")
```

---

runElementR *runElementR*

Description

Launch the shiny application contained in the elementR package.

Usage

```
runElementR()
```

Details

By running runElementR(), user has access through its web browser opened as soon as the application is launched to the whole functionalities for reducing data from solid-phase ICPMS analysis (project creation or edition, data or project export, standard and sample filtration, verification of the machine drift).

splitReplicate *splitReplicate*

Description

Split a single file corresponding to a whole session (Excel (.xls and .xlsx), OpenOffice (.ods) and text (.csv) worksheet) in separate files containing each only one replicate in order to fit to runElementR requirment

Usage

```
splitReplicate()
```

Details

By running `splitReplicate()`, the user has access to an interface through its web browser opened automatlically as soon as the function is launched. This interface allows to upload the file to split (sep = ";", dec = "."). The function helps the user to divide the file thanks to a clustering method (kmean). At the end, the user has the possibility to export the split data in the chosen directory (export in .csv).

Index