Package: ribiosIO (via r-universe)

May 16, 2026

Type Package

Title Input/Output Utilities of the 'ribios' Suite

Version 1.1.0

Date 2026-01-24

Description Provides data structures and functions for file input/output in the 'ribios' software suite, supporting common bioinformatics and computational biology file formats, designed for fast loading and high performance with minimal dependencies.

Depends R (>= 3.4.0)

Imports ribiosUtils, methods, utils

Suggests testthat

License GPL-3

URL https://github.com/bedapub/ribiosIO

BugReports https://github.com/bedapub/ribiosIO/issues

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'GctMatrix.R' 'ampliseq.R' 'checkfile.R' 'iofile.R' 'isGctFile.R' 'loadFile.R' 'read_bed.R' 'read_biokit_exprs.R' 'read_chip.R' 'read_cls.R' 'read_david.R' 'read_exprs_matrix.R' 'read_fasta.R' 'read_gct.R' 'read_gmt.R' 'read_illumina_sampleSheet.R' 'read_pheno.R' 'read_trimmed_lines.R' 'ribiosIO.R' 'write.tableList.R' 'writeMatrix.R' 'writeMatrix.tableList.R' 'writeStrList.R' 'write_gct.R' 'write_gmt.R'

Repository https://bedapub.r-universe.dev

Date/Publication 2026-02-18 09:12:47 UTC

RemoteUrl https://github.com/bedapub/ribiosIO

RemoteRef HEAD

RemoteSha 9cfe75462c48def5245386746a0695eb2792b67d

Contents

`[.GctMatrix` *Subsetting for GctMatrix*

Description

Subsetting for GctMatrix

Usage

```
## S3 method for class 'GctMatrix'
x[i, j, ...]
```

Arguments

`x`	A GctMatrix object
`i`	Index to subset rows, either integers, logical values, or characters. Other types will be converted to characters.
`j`	Index to subset columns.
`...`	Other parameters passed to matrix subsetting

Value

A `GctMatrix` object, subsetted according to the given indices.

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
gm1[1:2,]
gm1[c(TRUE, FALSE, TRUE),]
gm1[c("G3", "G1"),]
gm1[1:3,2:1]
gm1[1,]
gm1[,-1]
```

as.matrix.GctMatrix *Coerce a GctMatrix object into a matrix*

Description

Coerce a GctMatrix object into a matrix

Usage

```
## S3 method for class 'GctMatrix'
as.matrix(x, ...)
```

Arguments

x	A GctMatrix object
...	Not used

Value

A matrix with a desc attribute

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
print(gm1)
print(as.matrix(gm1))
```

as_numeric_matrix *Transform a data.frame to a numeric matrix without characters coereced as factors*

Description

Transform a data.frame to a numeric matrix without characters coereced as factors

Usage

```
as_numeric_matrix(df, warning = FALSE)
```

Arguments

df	A data.frame
warning	Logical, whether the function should warn when non-numeric characters are transformed

Value

A numeric matrix

`cbindGct` *Column bind (cbind) two GctMatrix objects*

Description

Column bind (cbind) two GctMatrix objects

Usage

```
cbindGct(
  gctMatrix1,
  gctMatrix2,
  feature = c("union", "intersection"),
  missingValue = 0
)
```

Arguments

`gctMatrix1`	The first object
`gctMatrix2`	The second object
`feature`	What happens if the set of the features in both objects differ? Either union or intersection is possible.
`missingValue`	Missing values, `NA` or numeric values (such as 0) are accepted

Value

A larger matrix, with gctMatrix1 on the left and gctMatrix2 on the right, with merged features and descriptions.

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
m2 <- matrix(c(9:7, 12:10), nrow=3, dimnames=list(sprintf("G%d", 3:1), sprintf("S%d", 3:4)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
gm2 <- GctMatrix(m2, desc=sprintf("Gene%d", 3:1))
gm1
gm2
gm12 <- cbindGct(gm1, gm2)
gm12
m3 <- matrix(13:18, nrow=3, dimnames=list(sprintf("G%d", 2:4), sprintf("S%d", 5:6)))
gm3 <- GctMatrix(m3, desc=sprintf("Gene%d", 2:4))
gm3
gm123Intersect <- cbindGct(gm12, gm3, feature="intersect")
print(gm123Intersect, showAll=TRUE)
```

```
gm123Union <- cbindGct(gm12, gm3, feature="union")
print(gm123Union, showAll=TRUE)
gm123UnionNA <- cbindGct(gm12, gm3, feature="union", missingValue = NA)
print(gm123UnionNA)
```

`find_ampliseq` *Find and read-in AmpliSeq files*

Description

Find and read-in AmpliSeq files into an expression matrix

Usage

```
find_ampliseq(dir)

read_ampliseq(files)

find_and_read_ampliseq(dir)
```

Arguments

`dir`	The top-level directory where a AmpliSeq run is saved. An example: '/data64/sequencing/iontorrent_data/ 139-AmpliSeqRNA_pathway_FD14_277_360/'
`files`	AmpliSeq files, potentially found by `find_ampliseq`

Details

Directory is recursively checked for files that match the name pattern '*.cov.xls' (cov means coverage). Invalid links (judged by file size) are excluded.

Only data of total read counts are read-in.

Value

`find_ampliseq` returns a character vector of full names of valid files.

`read_ampliseq` returns a numeric matrix of gene expression in counts. Row names are unique gene names.

`find_and_read_ampliseq` combines the two functions and returns the expression matrix as `read_ampliseq` does.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```
ampdir <- system.file("extdata/ampliseq-data", package="ribiosIO")
ampfiles <- find_ampliseq(ampdir)
ampmat <- read_ampliseq(ampfiles)

ampmat.onestep <- find_and_read_ampliseq(ampdir)
```

---

gctDesc *Retrieve feature (row) descriptions from a GctMatrix S3-object*

Description

Retrieve feature (row) descriptions from a GctMatrix S3-object

Usage

```
gctDesc(gctMatrix, index)
```

Arguments

gctMatrix	A GctMatrix object
index	Logical or integer index

Value

Character vector, feature descriptions

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
gctDesc(gm1)
gctDesc(gm1, 1:2)
```

GctMatrix *Create a GctMatrix object*

Description

Create a GctMatrix object

Usage

```
GctMatrix(matrix, desc)
```

Arguments

matrix	A numeric matrix
desc	Character vector of feature description, length must equal nrow of the matrix

Value

A GctMatrix object

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
m2 <- matrix(c(9:7, 12:10), nrow=3, dimnames=list(sprintf("G%d", 3:1), sprintf("S%d", 3:4)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
gm2 <- GctMatrix(m2, desc=sprintf("Gene%d", 3:1))
print(gm1)
print(gm2)
```

---

gctMatrix2longdf *Convert a GctMatrix into a long data frame*

Description

Convert a GctMatrix into a long data frame

Usage

```
gctMatrix2longdf(gctMatrix)
```

Arguments

gctMatrix	A GctMatrix object

Value

A data.frame with four columns: feature, desc, sample, and value

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gct.file <- file.path(idir, "test.gct")
test.mat <- read_gct_matrix(sample.gct.file, keep.desc=TRUE)
test.long <- gctMatrix2longdf(test.mat)
```

getDataDir *Get the data directory*

Description

Get the data directory

Usage

```
getDataDir()
```

Value

A directory The value stored in options is returned

iofile *Get file names for data import/export*

Description

Get file names for data import/export

Usage

```
iofile(x = NULL)
```

Arguments

x	File or directory name Quite often we need to import and export data (especially bulky files) into a directory other than the local file. This function is a shortcut to get full names of import/export files. The function first determines whether the option `dataDir` in the options of `ribiosIO` exists. If yes, its value will be used as the directory from/to which input/export files should be read/written. If the value does not exist yet, the function tries to use a folder named `data` in the current working directory as `dataDir`. If this local folder exists, its name will be assigned to the `dataDir` option. If the folder does not exist, the function will report an error and quit. The steps above guarantees that there is an option named `dataDir`, pointing to a directory where files are read from or written to. The parameter `x` can be file or directory names in the `dataDir` directory. In this case, `iofile(x)` returns their full names. When `x` is missing or `NULL`, `iofile()` returns the value of `dataDir`. A common usage for the later case is `dir(iofile())`.

Value

Character string, the full path to the data directory (when x is NULL) or the full file path(s) within the data directory.

Examples

```
setDataDir(system.file("extdata", package="ribiosIO"))
dir(iofile())
readLines(iofile("test.gct"), n=2)
```

---

is_factor_file *Check if a file encodes a factor*

Description

Check if a file encodes a factor

Usage

```
is_factor_file(con = stdin())

is_cls_file(con = stdin())
```

Arguments

con	Connection from which to read the file

Value

Logical, TRUE if the file is a valid CLS factor file, FALSE otherwise.

Examples

```
set.seed(1887)
tempfac <- factor(sample(LETTERS, 30, replace=TRUE), levels=sample(LETTERS))
tempfile <- tempfile()
write_factor(tempfac, tempfile)
is_factor_file(tempfile)
write_factor(tempfac, tempfile, sep=" ")
is_factor_file(tempfile)
```

isGctFile *Test a file is a GCT file or not*

Description

Test a file is a GCT file or not

Usage

```
isGctFile(file, strict.column.names = FALSE)
```

Arguments

file	Character string, a file name
strict.column.names	Logical, whether the names of the first two columns must be 'NAME(tab)Description'

Details

A file is a valid GCT file if it meets following three rules:

1. The first line of the file is #1.2
2. The second line contains number of rows and number of columns, separated by a tab.
3. The rest of file contain a rectangular matrix, with the first two columns named NAME and Description respectively.

Value

A logical value: TRUE means file is of the GCT format.

References

https://software.broadinstitute.org/cancer/software/genepattern/file-formats-guide

See Also

read_gct_matrix to read in GCT files

Examples

```
myInFile <- system.file("extdata/test.gct", package="ribiosIO")
isGctFile(myInFile)
myInfileLS <- system.file("extdata/test_lessStrict.gct", package="ribiosIO")
isGctFile(myInfileLS)
```

`loadFile` *Attempt to load a binary RData file*

Description

The function attempts to load a binary file, returning `TRUE` if succeeded. Otherwise it returns `FALSE`.

Usage

```
loadFile(rDataFile, env = parent.frame())
```

Arguments

`rDataFile`	Character, RData file name
`env`	Environment, where should be the RData loaded into. By default it is loaded into the caller's environment.

Value

Logical, `TRUE` if the file was loaded successfully, `FALSE` otherwise.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`iofile` can be used to find file from input data directory.

Examples

```
rf <- tempfile()
myData <- c(3,4,5)
save(myData, file=rf)
env <- new.env()
stopifnot(loadFile(rf, env=env))
```

loadObject *Load an object by its name from a RData file*

Description

Load an object by its name from a RData file

Usage

```
loadObject(file, obj = NULL, verbose = FALSE)
```

Arguments

file	A RData file
obj	Object name. If set as NULL, all objects are returned
verbose	Whether the loading process should be verbose, see load

Value

The object loaded from the RData file. If obj is NULL, returns the first object found.

loadObjectInEnv *Load objects from a RData file and return them in an environment*

Description

Load objects from a RData file and return them in an environment

Usage

```
loadObjectInEnv(file, obj = NULL, verbose = FALSE)
```

Arguments

file	A RData file
obj	Character string(s), optional object names. If set as NULL, all objects are returned
verbose	Whether the loading process should be verbose, see load

Value

An environment containing the loaded objects.

loadRDS *Load an object from a RDS file and returns a logical flag*

Description

Load an object from a RDS file and returns a logical flag

Usage

```
loadRDS(rdsFile, variableName, refhook = NULL)
```

Arguments

rdsFile	Character string, name of the rds file to be loaded
variableName	Character string or variable name, variable name to which the loaded value is assigned to
refhook	Logical, passed to readRDS

Value

Logical, TRUE if the file loading was successful, otherwise FALSE

longdf2gctMatrix *Convert a long data.frame into a GctMatrix*

Description

Convert a long data.frame into a GctMatrix

Usage

```
longdf2gctMatrix(
  longdf,
  row.col = 1L,
  desc.col = 2,
  column.col = 3,
  value.col = 4,
  missingValue = NULL
)
```

Arguments

longdf	A data.frame object
row.col	Integer or character string, index or name of the column in which row names are stored
desc.col	Integer or character string,, index or name of the column in which feature descriptions are stored
column.col	Integer or character string, index or name of the column in which sample names are stored
value.col	Integer or character string, index or name of the column in which values are stored
missingValue	Value used for missing values. If NULL, missing values are reported as NA and a warning will be raised if any value is missing. If NA, missing values are reported as NA and no warning is raised.

Value

A GctMatrix object

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gct.file <- file.path(idir, "test.gct")
test.mat <- read_gct_matrix(sample.gct.file, keep.desc=TRUE)
test.long <- gctMatrix2longdf(test.mat)
test.rmat <- longdf2gctMatrix(test.long)
```

---

optional_suppress_warning

Supress warning optionally

Description

Supress warning optionally

Usage

```
optional_suppress_warning(expr, suppress = TRUE)
```

Arguments

expr	R expression
suppress	Logical, whether or not to suppress warnings

Value

side effect is used

print.GctMatrix *Print method for GctMatrix object*

Description

Print method for GctMatrix object

Usage

```
## S3 method for class 'GctMatrix'
print(x, showAll = FALSE, ...)
```

Arguments

x	A GctMatrix object
showAll	Logical, whether all values should be printed
...	Paramters passed to the default method of print

Value

No return value, called for side effects (prints to console).

Examples

```
m1 <- matrix(1:6, nrow=3, dimnames=list(sprintf("G%d", 1:3), sprintf("S%d", 1:2)))
gm1 <- GctMatrix(m1, desc=sprintf("Gene%d", 1:3))
gm1
mBig <- matrix(round(rnorm(1000),3),
   nrow=100, dimnames=list(sprintf("G%d", 1:100), sprintf("S%d", 1:10)))
gmBig <- GctMatrix(mBig, desc=sprintf("Gene%d", 1:100))
gmBig
print(gmBig, showAll=TRUE)
```

read_ampliseq_bedcovgct *Read bedcov output of AmpliSeq amplicons and convert them to read counts*

Description

Read bedcov output of AmpliSeq amplicons and convert them to read counts

Usage

```
read_ampliseq_bedcovgct(file, bedFile)
```

Arguments

file	Character string, a GCT file containing bedcov output of amplicons
bedFile	Character string, an annotated BED file encoding amplicons

Value

A GctMatrix object containing read counts

The function is used to convert read base counts returned by samtools bedcov to read counts using Amplicon information encoded in the bed file

See Also

read_annotated_ampliseq_amplicons

Examples

```
bedlines <- paste0("#track type=bedDetail ionVersion=4.0 name=\"IAD50039-4_IAD87652-4_Design\"",
 "solution_type=4 description=\"TargetRegions_AmpliSeqID_IAD50039 AmpliSeq_Version=3.0.1",
 " Workflow=RNA merged with TargetRegions_AmpliSeqID_IAD87652 AmpliSeq_Version=4.48 Workflow=RNA\"",
 " color=77,175,74 priority=2", "\n",
 "NM_000014\t3316\t3421\tAMPL1384\t.\tGENE_ID=A2M;EntrezGeneID=2", "\n",
 "NM_005502\t2488\t2589\tAMPL28385508\t.\tGENE_ID=ABCA1;EntrezGeneID=19","\n",
 "NM_000927\t2520\t2624\tAMPL5599607\t.\tGENE_ID=ABCB1;EntrezGeneID=5243","\n",
 "NM_000443\t1367\t1470\tAMPL5513474\t.\tGENE_ID=ABCB4;EntrezGeneID=5244")
 gctLines <- paste0("#1.2", "\n",
 "3\t3","\n",
 "NAME\tDescription\tS1\tS2\tS3","\n",
 "A2M\tNM_000014\t105\t210\t315", "\n",
 "ABCA1\tNM_005502\t202\t303\t404", "\n",
 "ABCB1\tNM_000927\t312\t416\t520")
 bedcovGct <- read_ampliseq_bedcovgct(textConnection(gctLines),
   textConnection(bedlines))
 bedcovGct
```

---

read_annotated_ampliseq_amplicons

Read AmpliSeq amplicon informaiton from an annotated BED file

---

Description

Read AmpliSeq amplicon informaiton from an annotated BED file

Usage

```
read_annotated_ampliseq_amplicons(bedFile)
```

Arguments

bedFile	Character string, an annotated BED file with Gene_ID (gene symbols) and EntrezGeneID (Entrez Gene IDs) in the eighth column.

Value

A data.frame, besides reporting the columns in the BED file, contains following additional annotation information:

1. Amplicon
2. GeneID
3. GeneSymbol
4. RefSeq
5. Length

Note

There are several versions of BED file used. This function works only with the latest version.

See Also

read_bed

Examples

```
lines <- paste0("#track type=bedDetail ionVersion=4.0 name=\"IAD50039-4_IAD87652-4_Design\"",
 "solution_type=4 description=\"TargetRegions_AmpliSeqID_IAD50039 AmpliSeq_Version=3.0.1",
 " Workflow=RNA merged with TargetRegions_AmpliSeqID_IAD87652 AmpliSeq_Version=4.48 Workflow=RNA\"",
 " color=77,175,74 priority=2", "\n",
 "NM_000014\t3316\t3421\tAMPL1384\t.\tGENE_ID=A2M;EntrezGeneID=2", "\n",
 "NM_005502\t2488\t2589\tAMPL28385508\t.\tGENE_ID=ABCA1;EntrezGeneID=19","\n",
 "NM_000927\t2520\t2624\tAMPL5599607\t.\tGENE_ID=ABCB1;EntrezGeneID=5243","\n",
 "NM_000443\t1367\t1470\tAMPL5513474\t.\tGENE_ID=ABCB4;EntrezGeneID=5244")
 read_annotated_ampliseq_amplicons(textConnection(lines))
```

---

read_bed *Read a BED file*

Description

Read a BED file

Usage

```
read_bed(file, ...)
```

Arguments

file	Character string, name of a BED file.
...	Other parameters passed to read.table. If not specified, default settings are used.

Value

A data.frame containing all information in the BED file.

References

Definition of BED files can be found at https://www.ensembl.org/info/website/upload/bed.html.

See Also

read.table

Examples

```
lines <- paste0("#track type=bedDetail ionVersion=4.0 name=\"IAD50039-4_IAD87652-4_Design\"",
 "solution_type=4 description=\"TargetRegions_AmpliSeqID_IAD50039 AmpliSeq_Version=3.0.1",
 " Workflow=RNA merged with TargetRegions_AmpliSeqID_IAD87652 AmpliSeq_Version=4.48 Workflow=RNA\"",
 " color=77,175,74 priority=2", "\n",
 "NM_000014\t3316\t3421\tAMPL1384\t0\t+\t.\tGENE_ID=A2M;EntrezGeneID=2", "\n",
 "NM_005502\t2488\t2589\tAMPL28385508\t0\t+\t.\tGENE_ID=ABCA1;EntrezGeneID=19","\n",
 "NM_000927\t2520\t2624\tAMPL5599607\t0\t+\t.\tGENE_ID=ABCB1;EntrezGeneID=5243","\n",
 "NM_000443\t1367\t1470\tAMPL5513474\t0\t+\t.\tGENE_ID=ABCB4;EntrezGeneID=5244")
 read_bed(textConnection(lines))
```

---

read_biokit_exprs *qRead BioKit expression file into a data.frame*

Description

qRead BioKit expression file into a data.frame

Usage

```
read_biokit_exprs(filename)
```

Arguments

filename	A BioKit expression file The function uses an efficient C routine to read BioKit expression files. An Roche NGS expression file is eseentially a tab-delimited file. THe first six columns are mandatory (feature/tag name, multiple mapping RPKM, multiple mapping read count, unique mapping RPKM, unique mapping read count, and multiple mapping proportion). Right to these columns there can be arbitrary numbers of columns appended to annotate the features. In the current output, rows may have different numbers of columns: particularly for features without corresponding items in the annotation file used in the pipeline, their rows will contain the mandatory columns plus one extra column with the value "unknown". This is handled automatically by the function.

Value

A `data.frame` contains both mandatory and additional columns. The first column of the expression file will be used as the row names of the `data.frame` object.

See Also

`read_gct` for reading gct files, a commonly used file format for expression data.

Examples

```
biokitExampleFile <- system.file("extdata/biokit_expression_files/biokit-output-1.expression",
package="ribiosIO")
biokitExprs <- read_biokit_exprs(biokitExampleFile)
```

---

read_chip *Read CHIP file*

Description

The CHIP file format is commonly used to annotate probesets or other identifiers to gene symbols and gene names. This function imports CHIP files, using a C procedure to accelerate the speed.

Usage

```
read_chip(x)
```

Arguments

x	File name

Details

The current implementation only parses the first three columns and ignores the rest of columns. This behavior may change in future versions to provide larger flexibility of parsing CHIP-like files.

Value

A data.frame is returned with three columns: `ProbeSetID`, `GeneSymbol` and `GeneTitle`. The column names are concordant with the GSEA convention, except that the empty spaces are omitted.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

References

BROAD institute GSEA manual, available at https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats.

Examples

```
testFile <- system.file("extdata/test.chip", package="ribiosIO")
testChip <- read_chip(testFile)
head(testChip)
stopifnot(identical(colnames(testChip), c("ProbeSetID", "GeneSymbol", "GeneTitle")))
```

`read_david` *Read tab-delimited result files from DAVID Bioinformatics Resources*

Description

Read tab-delimited result files from DAVID Bioinformatics Resources

Usage

```
read_david(file)
```

Arguments

`file`	A file from DAVID Bioinformatics Resources

Value

A `data.frame` containing all information encoded in the file

Examples

```
davidFile <- system.file("extdata/example-DAVID-output-subset.txt", package="ribiosIO")
davidResult <- read_david(davidFile)
```

`read_exprs_matrix` *Read an expression matrix from file*

Description

Read an expression matrix from file. The file is either a GCT format file, a tab-delimited file or a space-delimited file.

Usage

```
read_exprs_matrix(x)
```

Arguments

`x`	File name

Details

An expression matrix of size m x n contains exprssion levels of m features in n samples. This function supports three commonly used file formats for expression levels: `GCT` format, `tab-delimited file` and `space-delimited file`.

Value

A `matrix`

Note

The function uses a very simple logic to guess whether the file is tab-delimited or space-delimited: it reads in the first n lines (currently `n=3`), and checks whether there is any tab character (`\t`): if yes, the file is parsed as tab-delmited, otherwise as space-delimited. Therefore, a space-delimited file should not contain tabs in case it needs to be parsed.

From ribiosIO version 1.0.2, this function supports duplicated row names.

From ribiosIO version 1.0-21, this function supports matrix file in which the second column is not numeric. This can happen, for instance, if the user decides to include descriptions. If such descriptions are detected, they are stored in the attribute "desc" so as to be later written into gct files.

From ribiosIO version 1.0-39, the function tolerates non-numeric values (such as '5?') in tab-delimited files better. However note that such values in the second column will cause problem because they will make the program interpret the second column as description but not numeric values.

Author(s)

Jitao David Zhang <jitao_david.zhang at roche.com>

See Also

The function calls internally the `read_gct_matrix` function to parse GCT files.

Examples

```
testfile.path <- system.file("extdata", package="ribiosIO")

## import gct
read_exprs_matrix(file.path(testfile.path,"test_read_exprs_matrix.gct"))

## import tab-separated file
read_exprs_matrix(file.path(testfile.path, "test_read_exprs_matrix.tsv"))

## import space-separated file
read_exprs_matrix(file.path(testfile.path, "test_read_exprs_matrix.txt"))

## import tab-separated file with descriptions
read_exprs_matrix(file.path(testfile.path, "test_read_exprs_matrix_desc.tsv"))

## import tab-separated file with non-numeric values
read_exprs_matrix(file.path(testfile.path, "test_nonnumbers.txt"))
```

---

read_factor *Read in a factor writtin in the CLS format*

Description

Read in a factor writtin in the CLS format

Usage

```
read_factor(con = stdin(), offset = 0)

read_cls(con = stdin(), offset = 0)
```

Arguments

`con`	File or connection to read file from
`offset`	The integer representing the first level, default is set to 0, for some software it can be set to 1

Value

A factor with levels as defined in the CLS file.

Note

The original CLS format specifies that both tab or space can be used as separators. This makes it unable to represent factors with sapces in levels. In order to accomodate CLS format for these factors, we propose using tab as separators in CLS files when encoding factors in R. The default setting of `read_factor` and `write_factor` uses tab. Though `read_factor` can handle both separators, as long as in the file a separator is consistenly used.

See Also

`write_factor`

Examples

```
set.seed(1887)
tempfac <- factor(sample(LETTERS, 30, replace=TRUE), levels=sample(LETTERS))
tempfile <- tempfile()
write_factor(tempfac, tempfile)
stopifnot(identical(tempfac, read_factor(tempfile)))

write_factor(tempfac, tempfile, sep=" ")
stopifnot(identical(tempfac, read_factor(tempfile)))

idir <- system.file("extdata", package="ribiosIO")
sample.cls <- read_factor(file.path(idir, "test.cls"))
expFac <- factor(c("Case", "Control")[c(1,0,1,0,0,1,0,0,0,1,0,1,0,0,0,1,0,1,0,0,1,1,1,1,0,0)+1],
levels=c("Case", "Control"))
stopifnot(identical(sample.cls, expFac))
```

`read_fasta` *Read (write) FASTA sequences into (from) named character vectors*

Description

`read_fasta` reads sequences in FASTA format in named character vectors. `write_fasta` writes sequences stored as named character vectors into FASTA file.

Usage

```
read_fasta(file)

write_fasta(x, file)
```

Arguments

`file`	FASTA format file
`x`	Named characters

Details

Names of sequences to be written do not have to begin with the greater-than sign, as they are appended by the function when writing. Similarly, the `read_fasta` removes the leading greater-than sign of sequence names.

Value

For `read_fasta`, a named character vector of FASTA sequences.

For `write_fasta`, the side effect is used and no value is returned.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```
tmpfile <- tempfile()
test.seq <- c("mySeq1"="ATGCG", "mySeq2 correct"="TTGTTCGACGT")
write_fasta(test.seq, tmpfile)
read_fasta(tmpfile)
```

---

read_gct_matrix *Calling C routine to read GCT file into a matrix*

Description

The function `read_gct_matrix` calls the C routine `read_gct` to read GCT file into a matrix.

Usage

```
read_gct_matrix(gct.file, keep.desc = TRUE)

read_gctstr_matrix(string, keep.desc = TRUE)
```

Arguments

`gct.file`	Character, name of a gct-format file
`keep.desc`	Logical, whether the description of features should be returned as an attribute of the matrix
`string`	Character string, a character string in the GCT-file format

Details

The function `read_gctstr_matrix` calls the C rountine as well, to parse a character string in the GCT file format into a matrix.

This function reads GCT files into a matrix, which is a basic data structure of R. For integration with Bioconductor's ExpressionSet objects, consider using the ribiosExpression package (available on GitHub).

Value

An matrix, optionally with feature descriptions as an attribute (`desc`) when `keep.desc` is set to `TRUE`.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`isGctFile` to test if a file is in GCT format.

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gct.file <- file.path(idir, "test.gct")

test.mat <- read_gct_matrix(sample.gct.file, keep.desc=TRUE)
test.simmat <- read_gct_matrix(sample.gct.file, keep.desc=FALSE)

sample.gct.string <- paste(readLines(sample.gct.file),collapse="\n")
teststr.mat <- read_gctstr_matrix(sample.gct.string, keep.desc=TRUE)
```

---

read_gmt_dataframe *Read gene-sets in a GMT file into a data.frame*

Description

Read gene-sets in a GMT file into a data.frame

Usage

```
read_gmt_dataframe(gmt.file, description = FALSE)
```

Arguments

`gmt.file`	Character, name of one gmt-format file
`description`	Logical, whether the result should contain descriptions of gene-sets as a column.

Value

A `data.frame`. If `description` is set to `FALSE`, the `data.frame` contains two columns: `geneset` and `gene`; otherwise, it contains three columns: `geneset`, `description`, and `gene`.

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gmt.file <- file.path(idir, "test.gmt")

testGmtDataframe <- read_gmt_dataframe(sample.gmt.file)
```

---

read_gmt_list *Calling C routine to read GMT file into a list*

Description

The function `read_gmt_list` calls the C routine `read_gmt` to read GMT file into a list.

Usage

```
read_gmt_list(gmt.file)
```

Arguments

`gmt.file`	Character, name of one gmt-format file

Details

Empty lines or lines without genes are omitted.Empty fields in "genes" are omitted as well.

Value

A list, the length of which equals the number of genesets. Each list contains three items:

`name`	Character, gene set name
`description`	Character, gene set description
`genes`	Character vector, genes in the set

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gmt.file <- file.path(idir, "test.gmt")

test.gmt <- read_gmt_list(sample.gmt.file)
```

read_illumina_sampleSheet *Read the Data block of Illumina sample sheet as data.frame*

Description

Read the Data block of Illumina sample sheet as data.frame

Usage

```
read_illumina_sampleSheet(file, sep = ",")
```

Arguments

file	An Illumina SampleSheet, with one Data block
sep	Character, separator between columns, comma by default

Value

A `data.frame` of the data block

Examples

```
myText <- paste("[Header]",
  "IEMFileVersion,5",
  "",
  "[Reads]",
  "51",
  "1",
  "[Data]",
  "Lane,Sample_ID,Description",
  "1,1,Sample1",
  "1,2,Sample2",
  "2,3,Sample3",
  "2,4,Sample4", sep="\n")
read_illumina_sampleSheet(textConnection(myText))
```

read_pheno *Read pheno data from CLS or tab-delimited file*

Description

Read pheno (sample annotation) data from CLS file or tab-delimited file (sample information file).

Usage

```
read_pheno(file)

read_pheno_factor(file)
```

Arguments

file	A CLS file or tab-delimited file

Details

read_pheno returns a data.frame.

read_pheno_factor returns a factor, indicating sample groups. If the input file is a tab-delimited file, it filters out columns which are identical for all samples and columns which are unique for each sample. Consequently the remaining covariates are concatenated by the underscore character to form a factor. See examples below

Value

read_pheno returns a data.frame containing sample annotations. In case of CLS input file, the data.frame contains two columns: Array (indices of arrays) and Class (classes indexed in the GCT file). In case of tab-delimited file, the file will be parsed into the data.frame, assuming the file having column names but no row names.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

References

For CLS and sample information file formats, see the GenePattern file formats documentation at https://software.broadinstitute.org/cancer/software/genepattern/file-formats-guide.

See Also

read_cls and read.csv.

Examples

```
testClsFile <- system.file("extdata/test.cls", package="ribiosIO")
testPhenoFile <- system.file("extdata/testSampleInfo.txt",
package="ribiosIO")

(clsPheno <- read_pheno(testClsFile))
(txtPheno <- read_pheno(testPhenoFile))

## read_pheno_factor
(clsPhenoClass <- read_pheno_factor(testClsFile))
(txtPhenoClass <- read_pheno_factor(testPhenoFile))
```

```
testPhenoFileCov <- system.file("extdata/testSampleInfo-cov.txt",package="ribiosIO")
read_pheno_factor(testPhenoFileCov)
```

`read_raw_ampliseq_amplicons` *Read AmpliSeq amplicon informaiton from an raw BED file*

Description

Read AmpliSeq amplicon informaiton from an raw BED file

Usage

```
read_raw_ampliseq_amplicons(bedFile)
```

Arguments

`bedFile`	Character string, a raw BED file coming from the AmpliSeq design pipeline (version 7.41+)

Value

A `data.frame`, besides reporting the columns in the BED file, contains following additional annotation information:

1. Amplicon
2. GeneSymbol (which may not be up-to-date)
3. RefSeq
4. Length

Examples

```
lines <- paste0("#track type=bedDetail ionVersion=4.0 name=\"IAD50039-4_IAD87652-4_Design\"",
 "solution_type=4 description=\"TargetRegions_AmpliSeqID_IAD50039 AmpliSeq_Version=3.0.1",
 " Workflow=RNA merged with TargetRegions_AmpliSeqID_IAD87652 AmpliSeq_Version=4.48 Workflow=RNA\"",
 " color=77,175,74 priority=2", "\n",
 "NM_000014\t3316\t3421\tAMPL1384\t.\tA2M", "\n",
 "NM_005502\t2488\t2589\tAMPL28385508\t.\tABCA1","\n",
 "NM_000927\t2520\t2624\tAMPL5599607\t.\tABCB1","\n",
 "NM_000443\t1367\t1470\tAMPL5513474\t.\tABCB4")
 read_raw_ampliseq_amplicons(textConnection(lines))
```

`read_trimmed_lines` *Read lines, thereby trimming empty spaces around the strings and removing empty lines*

Description

Read lines, thereby trimming empty spaces around the strings and removing empty lines

Usage

```
read_trimmed_lines(file, skipNul = TRUE, ...)
```

Arguments

`file`	A text file
`skipNul`	Skip NULL line (passed to `readLines`)
`...`	Other paratmers than `skipNul` passed to readLines

Value

Character vector of trimmed, non-empty lines.

Examples

```
lines <- "  ABC \n\tHBV\n\nFCB  \n\n"
trimmedLines <- read_trimmed_lines(textConnection(lines))
stopifnot(identical(trimmedLines, c("ABC", "HBV", "FCB")))
```

`readMatrix` *Read in numeric matrix from tab-delimited format written by* `writeMatrix`

Description

`readMatrix` reads a matrix written by `writeMatrix` into a R session

Usage

```
readMatrix(file, row.names = TRUE, as.matrix = TRUE, ...)
```

Arguments

file	file to be read in
row.names	Logical, whether the first column contains row names (should be consistent with the setting in writeMatrix)
as.matrix	Logical, whether the data.frame object should be cast into a matrix
...	Other parameters passed to read.table, for instance stringsAsFactors. Following parameters are *not* allowed to change: header, sep, quote, dec, check.names, strip.wrhite, comment.char.

Details

Default behaviour of read.table is adapted to the convention used in writeMatrix

Value

Matrix when as.matrix is set to TRUE and otherwise data.frame

Examples

```
test.mat <- matrix(rnorm(1000), nrow=10, dimnames=list(LETTERS[1:10], 1:100))
tmpfile <- tempfile()
writeMatrix(test.mat, tmpfile)
readin.mat <- readMatrix(tmpfile)
if(require(ribiosUtils)) identicalMatrix(test.mat, readin.mat)
```

---

readTable *Read in data.frame from tab-delimited format written by* writeMatrix

---

Description

readTable reads a data.frame written by writeMatrix into a R session

Usage

```
readTable(file, row.names = TRUE, ...)
```

Arguments

file	file to be read in
row.names	Logical, whether the first column contains row names (should be consistent with the settign in writeMatrix)
...	Other parameters passed to read.table, for instance stringsAsFactors. Following parameters are *not* allowed to change: header, sep, quote, dec, check.names, strip.wrhite, comment.char.

Details

Default behaviour of `read.table` is adapted to the convention used in `writeMatrix`

Value

A `data.frame` object

Examples

```
test.df <- data.frame(Team=c("HSV", "BVB", "VFB"), Score=c(21, 19, 17))
tmpfile <- tempfile()
writeMatrix(test.df, tmpfile)
readin.df <- readTable(tmpfile)
stopifnot(identical(as.character(readin.df$Team), c("HSV", "BVB", "VFB")))
stopifnot(identical(readin.df$Score, c(21L, 19L, 17L)))
```

---

setDataDir *Set the data directory*

Description

Set the data directory

Usage

```
setDataDir(path)
```

Arguments

`path`	Path to the data directory

Value

`NULL` The value is set in the options

strList2DataFrame *Format a string list into a data.frame*

Description

Format a string list into a data.frame

Usage

```
strList2DataFrame(strList, colnames = names(strList), index = FALSE)
```

Arguments

strList	A list of character strings. Other data types (e.g. factors) are converted to strings.
colnames	Column names of the resulting data.frame, by default the names of the list
index	Logical value, whether the row.names attribute of the data.frame should be integer indexes

Value

A character matrix with list elements as columns, padded with empty strings to equal length.

Examples

```
myList <- list("A"=LETTERS[3:5], "B"=LETTERS[4])
strList2DataFrame(myList)
strList2DataFrame(myList, colnames=c("FirstColumn", "SecondColumn"))
strList2DataFrame(myList, colnames=c("FirstColumn", "SecondColumn"), index=TRUE)

myFacList <- list("A"=gl(2,3, labels=LETTERS[1:2]),
    "B"=gl(3,4, labels=LETTERS[1:3]))
strList2DataFrame(myFacList)
```

write.tableList *Write a list of data.frames (tables) into files*

Description

Write a list of data.frames (tables) into files

Usage

```
write.tableList(list, file.names, ...)
```

Arguments

list	A list of data frames
file.names	File names. If missing, the names of the list will be used. Must be of the same length as the list
...	Other parameters that are passed to write.table

Value

No return value, called for side effects (writes files).

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

write.table

Examples

```
df1 <- data.frame(name=c("A", "B", "C"), value=1:3)
df2 <- data.frame(name=c("C", "D", "E"), value=seq(9,3,-3))
dflist <- list(file1=df1, file2=df2)

tmpdir <- tempdir()
write.tableList(dflist,
  file.names=file.path(tmpdir, c("file1.txt", "file2.txt")))
```

---

write_annotated_ampliseq_amplicons

Write AmpliSeq amplicon informaiton into an annotated BED file

---

Description

Write AmpliSeq amplicon informaiton into an annotated BED file

Usage

```
write_annotated_ampliseq_amplicons(
  df,
  bedFile,
  version = format(Sys.time(), "%Y%m%d")
)
```

Arguments

df	A data.frame containing following columns (names do not matter): 1. chrom (RefSeq IDs) 2. chromStart (integer) 3. chromEnd (integer) 4. name (Amplicon IDs) 5. score (A single value, .) 6. ID (in the format of GENE_ID=$GENESYMBOL;EntrezGeneID=$EG_ID)
bedFile	Character string, the output file
version	Character string, a version number. By default, the current date is used.

Value

No return value, called for side effects (writes an annotated BED file).

See Also

read_annotated_ampliseq_amplicons

Examples

```
mydf <- data.frame(chrom=c("NM_000014", "NM_000015", "NM_000021"),
  chromStart=c(3316, 50, 1212),
  chromEnd=c(3421, 146, 1320),
  name=c("AMPL1384", "AMPL7195", "AMPL14470"),
  score=".",
  ID=c("GENE_ID=A2M;EntrezGeneID=2",
       "GENE_ID=NAT2;EntrezGeneID=10",
       "GENE_ID=PSEN1;EntrezGeneID=5663"))
myBed <- tempfile()
write_annotated_ampliseq_amplicons(mydf, myBed)
mydfOut <- read_annotated_ampliseq_amplicons(myBed)
```

---

write_factor *Write a factor in the CLS format*

Description

Write a factor in the CLS format

Usage

```
write_factor(fac, con = stdout(), offset = 0, sep = c("\t", " "))

write_cls(fac, con = stdout(), offset = 0, sep = c("\t", " "))
```

Arguments

fac	A factor
con	Connection to write to
offset	he integer representing the first level, default is set to 0, for some software it can be set to 1
sep	Separator used in the CLS format, can be '\t' (recommended) or ' ' (not to be used when space exists in levels)

Value

No return value, called for side effects (writes to connection).

Note

The original CLS format specifies that both tab or space can be used as separators. This makes it unable to represent factors with sapces in levels. In order to accomodate CLS format for these factors, we propose using tab as separators in CLS files when encoding factors in R. The default setting of `read_factor` and `write_factor` uses tab.

See Also

`read_factor`

Examples

```
set.seed(1887)
tempfac <- factor(sample(LETTERS, 30, replace=TRUE), levels=sample(LETTERS))
tempfile <- tempfile()
write_factor(tempfac, tempfile)
readLines(tempfile)
stopifnot(identical(tempfac, read_factor(tempfile)))
```

---

write_gct *Write matrix in GCT file format*

Description

Write matrix in GCT file format

Usage

```
write_gct(matrix, file = stdout(), feat.name, feat.desc, na = "")
```

Arguments

matrix	A numeric matrix
file	Output file name. By default the file is written to standard output
feat.name	Character vector, optional. Feature names; if missing the row names are used as feature names. If given, feat.name must be of the same length as the row number of the input matrix.
feat.desc	Character vector, optional. Feature descriptions; if missing, empty strings will be used as descriptions.
na	Character string, how 'NA' values will be printed?

Details

Input matrix will be transformed into the GCT format. The transformed texts are printed on the standard output or in specified files.

If the input matrix has NULL as row names, and the feat.name option is left missing, a warning message will be print and the NAME column of the gct file will use integer indices starting from 1.

feat.desc specifies feature descriptions. Leaving is missing, or assigning it to NA or NULL will output a description column filled with empty strings.

Value

Texts printed in stdout() or in output file.

Note

From version 1.0-22, write_gct is able to handle zero-row matrix (see examples below)

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

read_gct_matrix to read matrix from GCT files.

Examples

```
tmpMatrix <- matrix(rnorm(15), nrow=3L, ncol=5L,
dimnames=list(LETTERS[1:3L], letters[1:5L]))

write_gct(tmpMatrix)
write_gct(tmpMatrix, file=tempfile())

## specify feature names
write_gct(tmpMatrix, feat.name=c("F1", "F2", "F3"))
write_gct(tmpMatrix, feat.name=c("F1", "F2", "F3"), feat.desc=NULL)
write_gct(tmpMatrix, feat.name=c("F1", "F2", "F3"), feat.desc=NA)
```

```
## specify feature names and descriptions
write_gct(tmpMatrix, feat.name=c("F1", "F2", "F3"), feat.desc=
c("Feature 1", "Feature 2", "Feature 3"))

## special case: 0-row matrix
write_gct(tmpMatrix[c(FALSE,FALSE,FALSE),,drop=FALSE])
```

`write_gmt` *Write a list of gene sets into a GMT file*

Description

Write gene-sets in a GMT-list form into GMT files.

Usage

```
write_gmt(gmt, file, description = NULL)
```

Arguments

`gmt`	A list of gene sets. It can be either (1) a list with each item is a list of three components, named 'name', 'description' and 'genes', or (2) a list of gene identifiers.
`file`	The GMT file to create
`description`	Description, used in case `gmt` is a list of gene identifiers (e.g. without description).

Details

This function can be used, for instance, to combine multiple GMT files into a new one.

Value

Invisible NULL when the file is successfully created. Otherwise an error message will be printed.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```
idir <- system.file("extdata", package="ribiosIO")
sample.gmt.file <- file.path(idir, "test.gmt")

test.gmt <- read_gmt_list(sample.gmt.file)

outgmt.file <- paste(tempfile(), ".gmt", sep="")
```

```
write_gmt(test.gmt[1:2], file=outgmt.file)

## a list of identifiers
testList <- list(A=LETTERS[3:5], B=LETTERS[4:7], C=12:9)
write_gmt(testList, file=outgmt.file)
```

`writeMatrix` *Export matrix into a commonly used tab-delimited format inside Roche Bioinformatics*

Description

`writeMatrix` writes a matrix into a non-quoted, tab-delimited file.

Usage

```
writeMatrix(x, file, row.names = TRUE)
```

Arguments

`x`	a matrix
`file`	file to be written to
`row.names`	logical, whether row.names is appended. Default: `TRUE`

Details

Different from the default behaviour of `write.table`, an empty cell is inserted as the header of row names (equivalent to setting `col.names` to `NA`

Value

No return value, called for side effects (writes to file).

See Also

`readMatrix` to read in matrix

Examples

```
test.mat <- matrix(rnorm(1000), nrow=10)
writeMatrix(test.mat, tempfile())
```

`writeMatrix.tableList` *Write a list of data.frames (tables) into file with writeMatrix*

Description

Write a list of data.frames (tables) into file with writeMatrix

Usage

```
writeMatrix.tableList(list, file.names, row.names = TRUE, ...)
```

Arguments

`list`	A list of data frames
`file.names`	File names. If missing, the names of the list will be used. Must be of the same length as the list
`row.names`	Logical, whether row.names should be in the first, unnamed column of the output files
`...`	Other parameters that are passed to `writeMatrix`

Value

Side-effects are used

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`writeMatrix`

Examples

```
td <- tempdir()
cwd <- getwd()
setwd(td)
df1 <- data.frame(name=c("A", "B", "C"), value=1:3)
df2 <- data.frame(name=c("C", "D", "E"), value=seq(9,3,-3))
dflist <- list(file1=df1, file2=df2)
writeMatrix.tableList(dflist) ## two files, file1 and file2, are written
dir()
writeMatrix.tableList(dflist, file.names=c("file1.txt", "file2.txt"))
dir()
setwd(cwd)
```

writeStrList *Write a list of strings in a tab-delimited file*

Description

Write a list of strings in a tab-delimited file

Usage

```
writeStrList(
  list,
  file,
  names = NULL,
  type = c("column", "row"),
  index = FALSE
)
```

Arguments

list	A list of character strings
file	A filename
names	Names of the list; by default the names of the list
type	Should list items written in columns or rows?
index	Logical, should integer index be printed along the elements?

Value

No return value, called for side effects (writes to file).

Examples

```
myList <- list("A"=LETTERS[3:5], "B"=LETTERS[4])
writeStrList(myList, file=stdout())
writeStrList(myList, file=stdout(), names=c("ListA", "ListB"))
writeStrList(myList, file=stdout(), names=c("ListA", "ListB"), type="row")
writeStrList(myList, file=stdout(), names=c("ListA", "ListB"), type="row", index=TRUE)
writeStrList(myList, file=stdout(), names=c("ListA", "ListB"), type="column", index=TRUE)
```

Index