

# Package ‘handyFunctions’

August 16, 2022

**Type** Package

**Title** Useful functions for handfully manipulating and analyzing data with data.frame format

**Version** 0.1.0

**Author** Hongfei Liu

**Maintainer** Hongfei Liu <lhf563@126.com>

**URL** <https://github.com/LuffyLouis/handyFunctions>

**BugReports** <https://github.com/LuffyLouis/handyFunctions/issues>

## Description

Some useful functions for simply manipulating and analyzing data with data.frame format.

It mainly includes the following sections:

ReformatDataframe

modifyColnames: modify the colnames of raw data.frame

modifyColTypes: automatically recognize and change the data types of each col

modifyRownames: modify the rownames of raw data.frame

unifyDataframe: reformat dataframe with the above all modifiers simultaneously

splitCol: return specific-

indexed vector according to given delimitator/separator by splitting one col in data.frame

InteractDataframe

mergeCustom: merge two data.frame based on xcol and ycol

matchIndex: return index of x data.frame with the given vec-

tor/list or ycol in data.frame (if set the accurate match or not)

Post-VCF

plotSNVdensity: Function of showing SNP density at chromosome level

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**Depends** R (>= 2.10)

**Imports** ggplot2 (>= 3.0),

stats (>= 3.0),

stringr (>= 1.0)

**Suggests** covr,

testthat (>= 3.0.0)

**Config/testthat/edition** 3

## R topics documented:

checkCols . . . . .	2
checkDtype . . . . .	3
grade . . . . .	3
matchIndex . . . . .	4
mergeCustom . . . . .	4
modifyColNames . . . . .	5
modifyColTypes . . . . .	6
modifyRowNames . . . . .	6
people . . . . .	7
queryingInfo . . . . .	8
ShowSNPDensityPlot . . . . .	8
SNV_1MB_density_data . . . . .	9
splitCol . . . . .	10
unifyDataframe . . . . .	10
<b>Index</b>	<b>12</b>

---

checkCols	<i>check the validation and return index of cols given from input in raw-DataFrame</i>
-----------	--

---

### Description

check the validation and return index of cols given from input in rawDataFrame

### Usage

```
checkCols(rawDataFrame, cols)
```

### Arguments

rawDataFrame raw data.frame  
cols specific cols given from input

### Value

return validation (only FALSE if invaild cols input) or index of cols

### Examples

```
library(handyFunctions)
data(people)
checkCols(people, c("..name", "..sex"))
# OR
checkCols(people, c(1, 2))
```

---

checkDtype	<i>Return suggested dtype of vector input</i>
------------	---

---

**Description**

Return suggested dtype of vector input

**Usage**

```
checkDtype(vector)
```

**Arguments**

vector                  vector/list input

**Value**

Return suggested dtypes of vector

**Examples**

```
library(handyFunctions)
vector <- c(1, 2, 3, "", NA, " ", "four", "NA", 5)
checkDtype(vector)
```

---

grade	<i>Grade records of virtual persons in high school</i>
-------	--

---

**Description**

A dataset containing the personal grade information (chinese, math, english, physics, biology, chemistry) of virtual persons.

**Usage**

```
grade
```

**Format**

A data frame with 6 rows and 7 variables:

**name** name, chinese or foreigner, in carats

**chinese** grade of the chinese, in numbers

**math** grade of the math, in numbers

**english** grade of the english, in numbers

**physics** grade of the physics, in numbers

**biology** grade of the biology, in numbers

**chemistry** grade of the chemistry, in numbers ...

**Source**

"simulated dataset"

---

matchIndex	<i>Return the index of source vector matched with query vector</i>
------------	--

---

**Description**

Return the index of source vector matched with query vector

**Usage**

```
matchIndex(SourceInfo, queryInfo, queryType = TRUE)
```

**Arguments**

SourceInfo    the source vector  
 queryInfo    the query vector  
 queryType    logical If set it to accurate match (default: TRUE)

**Value**

the index of source vector matched with query vector

**Examples**

```
library(handyFunctions)
data(grade)
matchIndex(grade[, "name"], c("Ming Li", "Bang Wei"))
```

---

mergeCustom	<i>merge two data.frame based on xcol and ycol</i>
-------------	--

---

**Description**

merge two data.frame based on xcol and ycol

**Usage**

```
mergeCustom(x, y, xcol, ycol)
```

**Arguments**

x                the first data.frame  
 y                the second data.frame  
 xcol            colnames which you want to merged in first data.frame  
 ycol            colnames which you want to merged in second data.frame

**Value**

return the new data.frame merged

**Examples**

```
library(handyFunctions)
data(people)
data(grade)
mergeCustom(people, grade, "..name", "name")
```

---

modifyColNames	<i>Return reformatted data.frame with standard col names</i>
----------------	--

---

**Description**

Return reformatted data.frame with standard col names

**Usage**

```
modifyColNames(rawDataFrame, cols = TRUE, rawSep = "..", sep = "_")
```

**Arguments**

rawDataFrame	Raw data.frame input
cols	Specific col names or indexes what you want to reformat (default: TRUE, use all cols)
rawSep	Raw odd separation symbol in col names of raw data.frame. Note: it supports regEx (regular expression), so "." means all possible symbols. If you want to use the "." dot notation, please use "[.]".
sep	Separation symbol in col names of modified data.frame

**Value**

A modified data.frame with col names separated by your given delimitator

**Examples**

```
library(handyFunctions)
data(people)
modified_people <- modifyColNames(people, rawSep = "[.][.]")
```

---

modifyColTypes	<i>Return suggested appropriate dtypes for each column in raw-DataFrame</i>
----------------	---

---

### Description

Return suggested appropriate dtypes for each column in rawDataFrame

### Usage

```
modifyColTypes(rawDataFrame, cols = TRUE, dtype = FALSE, custom = FALSE)
```

### Arguments

rawDataFrame	Raw data.frame
cols	Specify cols which you want to change its dtypes when custom is FALSE (default: TRUE, for all cols)
dtype	Specify indexed matched dtypes whcih you want to update when custom is FALSE (default: FALSE, for automatically update)
custom	Option whether set to auto/custom , you can specify your custom dtypes for cols given when setting to TRUE (default: FALSE, for auto)

### Value

Return a new data.frame with appropriate dtypes suggested for each cols

### Examples

```
library(handyFunctions)
data(people)
modifyColTypes(people)
```

---

modifyRowNames	<i>Return reformatted data.frame with standard row names</i>
----------------	--

---

### Description

Return reformatted data.frame with standard row names

### Usage

```
modifyRowNames(rawDataFrame, rows = TRUE, rowSep = "..", sep = "_")
```

### Arguments

<code>rawDataFrame</code>	Raw data.frame input
<code>rows</code>	Specific row names or indexes what you want to reformat (default: TRUE, use all row)
<code>rawSep</code>	Raw odd separation symbol in row names of raw data.frame. Note: it supports regEx (regular expression), so "." means all possible symbols. If you want to use the "." dot notation, please use "[.]".
<code>sep</code>	Separation symbol in row names of modified data.frame

### Value

A modified data.frame with row names separated by your given delimitator

### Examples

```
library(handyFunctions)
data(people)
modifyRowNames(people)
```

---

people

*Basic information of virtual persons*

---

### Description

A dataset containing the personal basic information (name, sex, age, and death\_age) of virtual persons.

### Usage

```
people
```

### Format

A data frame with 6 rows and 4 variables:

- ..name** name, chinese or foreigner, in carats
- ..sex** sex of the person, in carats
- ..age** living age in final record, in numbers
- ..death..age** final age when a person is dead, in numbers ...

### Source

"simulated dataset"

---

queryingInfo	<i>return index of x data.frame with the given vector/list or ycol in data.frame (if set the accurate match or not)</i>
--------------	---

---

### Description

return index of x data.frame with the given vector/list or ycol in data.frame (if set the accurate match or not)

### Usage

```
queryingInfo(SourceData, sourceCol, queryCol, queryInfo, queryType = TRUE)
```

### Arguments

SourceData	the source data.frame which you want to query
sourceCol	the col names or index of query field in source data.frame
queryCol	the col names or index of return field in source data.frame
queryInfo	vector/list the query info
queryType	logical if set it to accurate match (default: TRUE)

### Value

a vector in query field matched with query info in source data

### Examples

```
library(handyFunctions)
data(grade)
queryingInfo(grade, "name", "chinese", c("Ming Li", "Bang Wei"))
```

---

ShowSNPDensityPlot *Function of showing SNP density at chromosome level*

---

### Description

Function of showing SNP density at chromosome level

### Usage

```
ShowSNPDensityPlot(
  densityData,
  binSize,
  densityColorBar = c("grey", "darkgreen", "yellow", "red"),
  chromSet = c(1:22),
  withchr = FALSE
)
```



**Arguments**

densityData	the raw density data generated from vcftools
binSize	the bin size set while generating density data
densityColorBar	vector Specific the color bar for plotting density plot (generally four colors)
chromSet	vector Filtered chrom set which you want to plot (it must be matched with the CHROM column in densityData)
withchr	logical If the chromosome labels of density plot is prefixed with "chr". Note: it cannot work when the filtered chrom set contain other uncommon chrom symbols (e.g. NC0*, etc)

**Value**

A ggplot2 object for SNP density plot

**Examples**

```
library(handyFunctions)
data(SNV_1MB_density_data)
ShowSNPDensityPlot(SNV_1MB_density_data, binSize = 1e6, chromSet = c(38:1))
```

---

SNV\_1MB\_density\_data

*The SNP number within 1Mb bins at chromosome levels generated from transcriptome dataset of two dog populations (including wild wolf and domesticated dogs).*

---

**Description**

A dataset containing the SNV number within 1Mb bins called from transcriptome dataset of wild wolf and domesticated dogs.

**Usage**

```
SNV_1MB_density_data
```

**Format**

A data frame with 2544 rows and 4 variables:

**CHROM** chrom id, reference genome of CanFam3.1, in numbers/carats

**BIN\_START** the start genomic coordinate for one bin at relevant chromosome, in numbers

**SNP\_COUNT** the end genomic coordinate for one bin at relevant chromosome, in numbers

**VARIANTS.KB** SNV(variants) number within one bin per KB, in numbers ...

**Source**

"real dataset"

---

<code>splitCol</code>	<i>Return specific-indexed vector according to given delimiter/separator by splitting one col in data.frame</i>
-----------------------	---

---

### Description

Return specific-indexed vector according to given delimiter/separator by splitting one col in data.frame

### Usage

```
splitCol(data, col = FALSE, sep, index, fixed = TRUE)
```

### Arguments

<code>data</code>	vector or data.frame input
<code>col</code>	the col names or indexes if data.frame input
<code>sep</code>	separation deliminators
<code>index</code>	the index of symbol which you want
<code>fixed</code>	logical. If TRUE match split exactly, otherwise use regular expressions, detailed info can be seen in <a href="#">strsplit</a> .

### Value

specific-indexed vector or factor

### Examples

```
library(handyFunctions)
data(people)
splitCol(people, col = 1, sep = " ", index = 2)
```

---

<code>unifyDataframe</code>	<i>Reformat dataframe with the all modifiers simultaneously (colNames, rowNames and dtypes)</i>
-----------------------------	---

---

### Description

Reformat dataframe with the all modifiers simultaneously (colNames, rowNames and dtypes)

### Usage

```
unifyDataframe(
  rawDataFrame,
  rawRowSep = "..",
  rowSep = "_",
  rawColSep = "..",
  colSep = "_",
  changeDtype = TRUE
)
```

**Arguments**

<code>rawDataFrame</code>	raw data.frame
<code>rawRowSep</code>	raw separation delimitator of row names in raw data.frame
<code>rowSep</code>	the new separation delimitator of row names
<code>rawColSep</code>	raw separation delimitator of col names in raw data.frame
<code>colSep</code>	the new separation delimitator of col names
<code>changeDtype</code>	if change the dtypes of cols

**Value**

A modified data.frame with applied to above all modifiers

**Examples**

```
library(handyFunctions)
data(people)
unifyDataframe(people, rawColSep = "[.][.]")
```

# Index

## \*Topic **datasets**

- grade, [3](#)
- people, [7](#)
- SNV\_1MB\_density\_data, [9](#)

- checkCols, [2](#)
- checkDtype, [3](#)

- grade, [3](#)

- matchIndex, [4](#)
- mergeCustom, [4](#)
- modifyColNames, [5](#)
- modifyColTypes, [6](#)
- modifyRowNames, [6](#)

- people, [7](#)

- queryingInfo, [8](#)

- ShowSNPDensityPlot, [8](#)
- SNV\_1MB\_density\_data, [9](#)
- splitCol, [10](#)

- unifyDataframe, [10](#)