

Package: messy (via r-universe)

October 1, 2026

Title Create Messy Data from Clean Data Frames

Version 0.1.0.9003

Description For the purposes of teaching, it is often desirable to show examples of working with messy data and how to clean it. This R package creates messy data from clean, tidy data frames so that students have a clean example to work towards.

Depends R (>= 4.1)

Imports dplyr, rlang, stringr

License CC BY 4.0

Suggests lubridate, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/Needs/website nrennie/nrenniepkgdown

URL <https://nrennie.rbind.io/messy/>, <https://github.com/nrennie/messy>

BugReports <https://github.com/nrennie/messy/issues>

Config/pak/sysreqs libicu-dev

Repository <https://nrennie.r-universe.dev>

Date/Publication 2025-08-29 18:09:12 UTC

RemoteUrl <https://github.com/nrennie/messy>

RemoteRef HEAD

RemoteSha 7cd51711094b8db0d40c268d367d81cb051a78a3

Contents

add_special_chars	2
add_whitespace	3
change_case	3

change_separators	4
duplicate_columns	5
duplicate_rows	5
make_missing	6
messy	7
messy_colnames	8
messy_datetime_formats	8
messy_datetime_tzones	9
split_datetimes	10
unjoin	11
unrbind	13
Index	15

add_special_chars	<i>Add special characters to strings</i>
-------------------	--

Description

Add special characters to strings

Usage

add_special_chars(data, cols = NULL, messiness = 0.1)

Arguments

- data input dataframe
- cols set of columns to apply transformation to. If NULL will apply to all columns. Default NULL.
- messiness Percentage of values to change. Must be between 0 and 1. Default 0.1.

Value

a dataframe the same size as the input data.

Examples

add_special_chars(mtcars)

add_whitespace	<i>Add whitespaces</i>
----------------	------------------------

Description

Randomly add whitespaces to the end of some values in all or a subset of columns.

Usage

```
add_whitespace(data, cols = NULL, messiness = 0.1)
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all columns. Default NULL.
messiness	Percentage of values to change. Must be between 0 and 1. Default 0.1.

Value

a dataframe the same size as the input data.

Examples

```
add_whitespace(mtcars)
```

change_case	<i>Change case</i>
-------------	--------------------

Description

Randomly switch between title case and lowercase for character strings

Usage

```
change_case(data, cols = NULL, messiness = 0.1, case_type = "word")
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all columns. Default NULL.
messiness	Percentage of values to change. Must be between 0 and 1. Default 0.1.
case_type	Whether the case should change based on the "word" or "letter".

Value

a dataframe the same size as the input data.

Examples

```
change_case(mtcars)
```

change_separators	<i>Change separators</i>
-------------------	--------------------------

Description

Randomly change the separators in character strings through random replacement

Usage

```
change_separators(  
  data,  
  cols = NULL,  
  messiness = 0.1,  
  sep_in = c("-", "_", " ", " " ),  
  sep_out = c("-", "_", " ", " ")  
)
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all columns. Default NULL.
messiness	Percentage of values to change. Must be between 0 and 1. Default 0.1.
sep_in	A single value, or vector, or list of what is considered a separator in the input data. Default c("-", "_", " ", " ").
sep_out	A single value, or vector, or list of what the separators may be randomly with. Default c("-", "_", " ", " ").

Value

a dataframe the same size as the input data.

Examples

```
change_separators(mtcars)
```

duplicate_columns	<i>Duplicate columns and insert them into the dataframe at random</i>
-------------------	---

Description

Duplicate columns and insert them into the dataframe at random

Usage

```
duplicate_columns(data, messiness = 0.1, random = TRUE, name_sep = "")
```

Arguments

data	input dataframe
messiness	Probability that each column is duplicated. Must be between 0 and 1. Default 0.1.
random	Whether duplicated column names should be randomly selected from other column names, or maintain the original. Default TRUE.
name_sep	Separator to use for adding numbers to end of names. Default "".

Value

A dataframe with duplicated rows inserted

Author(s)

Jordi Rosell

Examples

```
duplicate_columns(mtcars, messiness = 0.1)
```

duplicate_rows	<i>Duplicate rows and insert them into the dataframe in order or at random. May result in numbers being added to the end of row names.</i>
----------------	--

Description

Duplicate rows and insert them into the dataframe in order or at random. May result in numbers being added to the end of row names.

Usage

```
duplicate_rows(data, messiness = 0.1, shuffle = FALSE)
```

Arguments

data	input dataframe
messiness	Percentage of rows to duplicate. Must be between 0 and 1. Default 0.1.
shuffle	Insert duplicated data underneath original data or insert randomly

Value

A dataframe with duplicated rows inserted

Author(s)

Philip Leftwich, Barry Rowlingson

Examples

```
duplicate_rows(mtcars, messiness = 0.1)
```

make_missing	<i>Make missing</i>
--------------	---------------------

Description

Randomly make values missing in all data columns, or a subset of columns

Usage

```
make_missing(data, cols = NULL, messiness = 0.1, missing = NA)
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all columns. Default NULL.
messiness	Percentage of values to change. Must be between 0 and 1. Default 0.1.
missing	A single value, vector, or list of what the missing values will be replaced with. If length is greater than 1, values will be replaced randomly. Default NA.

Value

a dataframe the same size as the input data.

Examples

```
make_missing(mtcars)
```

*messy**Messy*

Description

Make a data frame messier.

Usage

```
messy(  
  data,  
  messiness = 0.1,  
  missing = NA,  
  case_type = "word",  
  sep_in = c("-", "_", " ", " "),  
  sep_out = c("-", "_", " ", " ")  
)
```

Arguments

<code>data</code>	input dataframe
<code>messiness</code>	Percentage of values to change per function. Must be between 0 and 1. Default 0.1.
<code>missing</code>	A single value, vector, or list of what the missing values will be replaced with. If length is greater than 1, values will be replaced randomly. Default NA.
<code>case_type</code>	Whether the case should change based on the "word" or "letter".
<code>sep_in</code>	A single value, or vector, or list of what is considered a separator in the input data. Default <code>c("-", "_", " ", " ")</code> .
<code>sep_out</code>	A single value, or vector, or list of what the separators may be randomly with. Default <code>c("-", "_", " ", " ")</code> .

Value

a dataframe the same size as the input data.

Examples

```
messy(mtcars)
```

<code>messy_colnames</code>	<i>Make column names messy</i>
-----------------------------	--------------------------------

Description

Adds special characters and randomly capitalises characters in the column names of a data frame.

Usage

```
messy_colnames(data, messiness = 0.2)
```

Arguments

<code>data</code>	data.frame to alter column names
<code>messiness</code>	Percentage of values to change per function. Must be between 0 and 1. Default 0.1.

Value

data.frame with messy column names

Author(s)

Athanasia Monika Mowinckel

Examples

```
messy_colnames(mtcars)
```

<code>messy_datetime_formats</code>	<i>Make date(time) formats inconsistent</i>
-------------------------------------	---

Description

Takes any date(times) column and transforms it into a character column, sampling from any number of random of valid character representations.

Usage

```

messy_datetime_formats(
  data,
  cols = NULL,
  formats = c("%Y/%m/%d %H:%M:%S", "%d/%m/%Y %H:%M:%S")
)

messy_date_formats(
  data,
  cols = NULL,
  formats = c("%Y/%m/%d", "%d/%m/%Y")
)

```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all POSIXt columns (for messy_datetime_formats()) or Date columns (for messy_date_formats()).
formats	A vector of any number of valid strptime() formats. Multiple formats will be sampled at random.

Value

a dataframe the same size as the input data.

Author(s)

Jack Davison

See Also

Other Messy date(time) functions: [messy_datetime_tzones\(\)](#), [split_datetimes\(\)](#)

Examples

```

data <- data.frame(dates = rep(Sys.Date(), 10))
messy_date_formats(data)

```

`messy_datetime_tzones` *Change the timezone of datetime columns*

Description

Takes any number of datetime columns and changes their timezones either totally at random, or from a user-provided list of timezones.

Usage

```
messy_datetime_tzones(data, cols = NULL, tzones = OlsonNames(), force = FALSE)
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all POSIXt columns.
tzones	Valid time zones to sample from. By default samples from all OlsonNames() , but can be set to options more relevant to the data.
force	By default (force = FALSE) the datetimes will have their actual hour/minute values changed along with the timezones. If force = TRUE, which requires lubridate , the datetime values will remain the same and only the timezone will differ.

Value

a dataframe the same size as the input data.

Author(s)

Jack Davison

See Also

Other Messy date(time) functions: [messy_datetime_formats\(\)](#), [split_datetimes\(\)](#)

Examples

```
data <- data.frame(dates = rep(Sys.time(), 10))

data$dates
attr(,"tzone")

messy <- messy_datetime_tzones(data, tzones = "Poland")
messy$dates
attr(,"tzone")
```

split_datetimes

Splits date(time) column(s) into multiple columns

Description

These functions can split the "date" and "time" components of POSIXt columns and the "hour", "month", and "day" components of Date columns into multiple columns.

Usage

```
split_datetimes(data, cols = NULL, class = c("character", "date"))

split_dates(data, cols = NULL)
```

Arguments

data	input dataframe
cols	set of columns to apply transformation to. If NULL will apply to all POSIXt columns (for split_datetimes()) or Date columns (for split_dates()).
class	For split_datetimes() . The desired output of the separate "date" and "time" columns. "character" leaves the columns as character vectors. "date" will reformat the date as a "Date" and the time as a "POSIXct" object, with a dummy date appended to it. In split_dates() , all returned columns are integers.

Value

a dataframe

Author(s)

Jack Davison

See Also

Other Messy date(time) functions: [messy_datetime_formats\(\)](#), [messy_datetime_tzones\(\)](#)

Examples

```
# split datetimes
data <- data.frame(today = Sys.time())
split_datetimes(data)
# split dates
data <- data.frame(today = Sys.Date())
data
split_dates(data)
```

unjoin	<i>Splits a dataframe into two, such that it could be reassembled with a mutating join</i>
--------	--

Description

This function takes an arbitrary number of 'joining' columns and any number of additional column names and splits a dataframe in two such that a user could then re-join using [merge\(\)](#) or [dplyr::left_join\(\)](#). The user may find it appropriate to go on and apply [messy\(\)](#) to each new dataframe independently to impede rejoining.

Usage

```
unjoin(data, by, cols, distinct = "none", names = c("left", "right"))
```

Arguments

data	input dataframe
by	a vector of column names which will be present in both outputs, to rejoin the dataframes
cols	specific columns to be present in the 'right' dataframe. implicitly, all other columns not in 'cols' will be present in the 'left' dataframe.
distinct	Apply <code>dplyr::distinct()</code> to "both" dataframes, the "left" or "right" dataframes, or "none" of the dataframes. This may be useful if one table is a 'lookup' or metadata table that has its values repeated many times in data.
names	The names of the output list. If NULL the list will be unnamed.

Details

Real data is often found across multiple datasets. For example, in environmental monitoring, measurements at a monitoring station may need to be bound with metadata about the station such as geographic coordinates, or even meteorological data from an external source, to produce desired outputs. In clinical research it may be necessary to combine the results of a clinical trial with relevant patient information, such as weight or sex. This function undoes existing joins to present learners with an authentic problem to solve; joining two independent datasets to achieve some goal.

Value

A list of two dataframes

Author(s)

Jack Davison

See Also

Other data deconstructors: `unrbind()`

Examples

```
dummy <-
  dplyr::tibble(
    patient_id = c(1, 1, 1, 2, 2, 2, 3, 3, 3),
    test = c(1, 2, 3, 1, 2, 3, 1, 2, 3),
    result = c("++", "+", "-", "--", "+", "-", "+", "+", "-"),
    sex = c("M", "M", "M", "M", "M", "M", "F", "F", "F"),
    age = c(50, 50, 50, 25, 25, 25, 30, 30, 30)
  )

unjoin(
  dummy,
```

```

    by = "patient_id",
    cols = c("sex", "age"),
    distinct = "right",
    names = c("tests", "patient_info")
  )

```

unrbind	<i>Splits a dataframe row-wise or col-wise into any arbitrary number of dataframes</i>
---------	--

Description

This function splits a dataframe into any number of dataframes such that they can be rejoined by using `rbind()/dplyr::bind_rows()` for `unrbind()` or `cbind()/dplyr::bind_cols()` for `uncbind()`. The user may find it appropriate to go on and apply `messy()` to each new dataframe independently to impede rejoining.

Usage

```
unrbind(data, sizes = NULL, probs = NULL, names = NULL, shuffle = TRUE)
```

```
uncbind(data, sizes = NULL, probs = NULL, names = NULL, shuffle = TRUE)
```

Arguments

data	input dataframe
sizes	A vector of numeric inputs summing to <code>nrow(data)</code> for <code>unrbind()</code> or <code>ncol(data)</code> for <code>uncbind()</code> ; the number of rows of each resulting dataframe. See <code>probs</code> for an alternative approach. If neither are provided, the dataframe will be split roughly in half.
probs	A vector of numeric inputs summing to 1; the proportion of rows/columns in each resulting dataframe. An alternative to <code>sizes</code> .
names	The names of the output list. If <code>NULL</code> the list will be unnamed.
shuffle	Shuffle rows in <code>unrbind()</code> or columns in <code>uncbind()</code> ? Defaults to <code>TRUE</code> .

Details

Real data can often be found in disparate files. For example, data reports may come in monthly and require row-binding together to obtain a complete annual time series. Scientific results may arrive from different laboratories and require binding together for further analysis and comparisons. This function may simulate a single dataframe having come from different sources and requiring binding back together. Base R's `split()` offers an alternative to `unrbind()`, but requires a pre-existing factor column to split by and cannot as easily create random splits in the data.

Value

A list of dataframes

Author(s)

Jack Davison

See Also

Other data deconstructors: [unjoin\(\)](#)

Examples

```
unrbind(dplyr::tibble(mtcars), probs = c(0.5, 0.3, 0.2))
```

```
uncbind(dplyr::tibble(mtcars), probs = c(0.5, 0.3, 0.2))
```

Index

* Messy date(time) functions

messy_datetime_formats, 8

messy_datetime_tzones, 9

split_datetimes, 10

* data deconstructors

unjoin, 11

unrbind, 13

add_special_chars, 2

add_whitespace, 3

cbind(), 13

change_case, 3

change_separators, 4

dplyr::bind_cols(), 13

dplyr::bind_rows(), 13

dplyr::distinct(), 12

dplyr::left_join(), 11

duplicate_columns, 5

duplicate_rows, 5

lubridate, 10

make_missing, 6

merge(), 11

messy, 7

messy(), 11, 13

messy_colnames, 8

messy_date_formats
(messy_datetime_formats), 8

messy_date_formats(), 9

messy_datetime_formats, 8, 10, 11

messy_datetime_formats(), 9

messy_datetime_tzones, 9, 9, 11

OlsonNames(), 10

rbind(), 13

split(), 13

split_dates(split_datetimes), 10

split_dates(), 11

split_datetimes, 9, 10, 10

split_datetimes(), 11

strptime(), 9

uncbind(unrbind), 13

uncbind(), 13

unjoin, 11, 14

unrbind, 12, 13

unrbind(), 13