# Package 'messy'

December 3, 2024

**Title** Create Messy Data from Clean Data Frames

**Version** 0.1.0

**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

**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

**NeedsCompilation** no

**Author** Nicola Rennie [aut, cre, cph] (<https://orcid.org/0000-0003-4797-557X>)

**Maintainer** Nicola Rennie <nrennie35@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-12-03 18:50:02 UTC

## Contents

---

`add_special_chars` *Add special characters to strings*

---

## 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` *Add whitespaces*

---

## 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 *Change case*

---

### 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)
```

## duplicate_rows *Duplicate rows and insert them into the dataframe in order or at random*

### Description

Duplicate rows and insert them into the dataframe in order or at random

### 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

### Examples

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

## make_missing *Make missing*

### 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")
```

### Arguments

| | |
|---|---|
| data | input dataframe |
| messiness | Percentage of values to change per function. 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. |
| case_type | Whether the case should change based on the "word" or "letter". |

### Value

a dataframe the same size as the input data.

### Examples

```
messy(mtcars)
```

`messy_colnames` *Make column names messy*

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `data` | data.frame to alter column names |
| `messiness` | 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)
```

`messy_datetime_formats` *Make date(time) formats inconsistent*

## 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$dates
attr(data$dates, "tzone")

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

# Index