# Package ‘fixr’

April 20, 2023

**Title** Fixing Data Made Easy for Statistical Analysis

**Version** 0.1.0

**Description** A set of functions that facilitate basic data manipulation and cleaning for statistical analysis including functions for finding and fixing duplicate rows and columns, missing values, outliers, and special characters in column and row names and functions for checking data consistency, distribution, quality, reliability, and structure.

**License** GPL-3

**URL** https://github.com/ambuvjyn/fixr

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Imports** RCurl, httr, xml2

**NeedsCompilation** no

**Author** Ambu Vijayan [aut, cre] (<https://orcid.org/0000-0001-8924-5685>),
J. Sreekumar [aut] (<https://orcid.org/0000-0002-4253-6378>, ICAR - Central Tuber Crops Research Institute)

**Maintainer** Ambu Vijayan <ambuvjyn@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-04-20 16:32:31 UTC

## R topics documented:

---

## check_data_consistency *Check Data Consistency Between Two Data Frames*

---

### Description

This function compares the column names and number of rows in two data frames and returns a message indicating whether the data is consistent or not.

### Usage

```
check_data_consistency(df1, df2)
```

### Arguments

| | |
|---|---|
| df1 | First data frame to compare |
| df2 | Second data frame to compare |

### Value

A message indicating whether the data is consistent or not.

## Examples

```
df1 <- data.frame(x = c(1,2,3), y = c(4,5,6))
df2 <- data.frame(x = c(1,2,3), y = c(4,5,6))
check_data_consistency(df1, df2)
# Data is consistent across the two sources.
df3 <- data.frame(a = c(1,2,3), b = c(4,5,6))
check_data_consistency(df1, df3)
# Data is not consistent across the two sources.
```

# check_data_distribution *Check the data distribution of a data frame*

## Description

This function checks if the data is normally distributed for each numeric column in a data frame.

## Usage

```
check_data_distribution(df)
```

## Arguments

| | |
|---|---|
| df | A data frame |

## Value

This function does not return anything, it only prints messages to the console.

## Examples

```
df <- data.frame(x = c("a", "b", "c"), y = c(4, 5, 6), z = c(7, 8, 9))

check_data_distribution(df)
```

# check_data_quality *Check Data Quality*

## Description

This function performs a series of data quality checks on a given dataframe, including checking the data structure, missing values, data accuracy, negative values, outliers, sample size, duplicate rows, and duplicate columns.

### Usage

```
check_data_quality(df)
```

### Arguments

df A dataframe.

### Value

A message indicating the results of each data quality check.

### Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, NA, -6, 4), z = c(7, 8, 180, 7))

# Check the data quality of the example dataframe
check_data_quality(df)
```

## check_data_reliability *Check inter-rater or test-retest reliability between numeric columns*

### Description

This function checks for inter-rater or test-retest reliability between all pairs of numeric columns in a data frame by computing the correlation between each pair and reporting if it is less than 0.8.

### Usage

```
check_data_reliability(df)
```

### Arguments

df A data frame

### Value

A message indicating whether the data is reliable or not between each pair of columns.

### Examples

```
df <- data.frame(x = c("a", "b", "c"), y = c(4, 5, 6), z = c(7, 8, 180))
check_data_reliability(df)
```

## `check_data_structure` *Check the structure of the data*

### Description

This function checks the structure of the given data frame and prints the number of rows, number of columns, column names, column data types, and number of missing values.

### Usage

```
check_data_structure(df)
```

### Arguments

| | |
|---|---|
| `df` | The data frame to be checked. |

### Value

None

### Examples

```
df <- data.frame(id = 1:10,
gender = c("male", "female", "male", "male", "male", "male", "male", "male", "female", "female"),
age = c(25, 32, 45, 19, 27, 56, 38, 42, 33, NA),
salary = c(50000, 60000, 75000, 45000, 55000, 90000, NA, 80000, 65000, 70000))

# Check the data structure of the example dataframe
check_data_structure(df)
```

## `check_for_negative_values` *Check if a data frame contains negative values.*

### Description

This function checks if a data frame contains negative values and returns their indices if any are found.

### Usage

```
check_for_negative_values(df)
```

### Arguments

| | |
|---|---|
| `df` | The data frame to check for negative values. |

### Value

If negative values are found, the function returns their indices as an array index object. If no negative values are found, NULL is returned.

### Examples

```
df <- data.frame(a = c(1, 2, 3), b = c(-1, 0, 1))
check_for_negative_values(df)
# [1] "Data frame contains negative values."
#      row col
# [1,]   2   1"
```

## check_missing_values *Check for Missing Values in Data Frame*

### Description

This function checks for missing values in a data frame and prints out the names of the columns with missing values and their counts.

### Usage

```
check_missing_values(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame to check for missing values. |

### Value

A message indicating if missing values were found or not.

### Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, 5, -6, 4), z = c(7, 8, NA, 7))
check_missing_values(df)
```

## check_outliers *Check for Outliers or Extreme Values in Data*

### Description

This function checks for outliers or extreme values in a given dataframe.

### Usage

```
check_outliers(df)
```

### Arguments

| | |
|---|---|
| df | A dataframe. |

### Value

A message indicating whether or not extreme values were found.

### Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, 5, -6, 4), z = c(7, 8, NA, 7))

check_outliers(df)
```

## check_sample_size *Check if sample size is adequate*

### Description

This function checks if the sample size of a data frame is adequate for statistical analysis.

### Usage

```
check_sample_size(df)
```

### Arguments

| | |
|---|---|
| df | A data frame to be checked |

### Value

A message indicating if the sample size is adequate or too small

### Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, 5, -6, 4), z = c(7, 8, 18, 7))
check_sample_size(df)
```

## find.packages *Find R packages that can import a file format*

### Description

This function searches the CRAN repository for R packages that can be used to import a file format

### Usage

```
find.packages(file_extension)
```

### Arguments

`file_extension` The file extension for the file format to search for packages to import

### Value

A character vector of package names that can be used to import the file format

## find.packages_path *Find the R Packages to Import a File Format*

### Description

This function takes a file path as input and searches the CRAN repository for R packages that can import the file format.

### Usage

```
find.packages_path(file_path)
```

### Arguments

`file_path` A character string specifying the file path of the file to be imported.

### Value

A character string that lists the R packages that can be used to import the file format of the input file.

## Examples

```
# Search for packages that can import a CSV file
find.packages_path("sample.csv")

# Search for packages that can import a JSON file
find.packages_path("sample.json")
```

---

# find_duplicate_cols *Find Duplicate Columns*

---

## Description

This function takes a data frame as input and checks for duplicate columns. A column is considered a duplicate of another column if all values in both columns are the same. If any duplicate columns are found, the function prints a message indicating which columns are duplicates of which other columns. If no duplicate columns are found, the function prints a message indicating that no duplicates were found.

## Usage

```
find_duplicate_cols(df)
```

## Arguments

| | |
|---|---|
| df | A data frame |

## Value

A message indicating which columns are duplicates of which other columns

## Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, NA, -6, 4), z = c(7, 8, 180, 7))
find_duplicate_cols(df)
# Column 'c' is a duplicate of column 'a'
```

## find_duplicate_rows *Find duplicate rows in a data frame*

### Description

This function identifies and reports duplicate rows in a given data frame. It first removes any rows with no values in all cells, and then compares each row to subsequent rows to check for duplicates. Duplicate rows are identified by having the same values in all columns. The function returns a message stating whether or not duplicate rows were found, and if so, the row numbers of the duplicate and original rows.

### Usage

```
find_duplicate_rows(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame to check for duplicate rows. |

### Value

A message stating whether or not duplicate rows were found, and if so, the row numbers of the duplicate and original rows.

### Examples

```
# Create example data frame
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "a"),
                 y = c(4, 5, -6, 4), z = c(7, 8, NA, 7))
# Find duplicate rows
find_duplicate_rows(df)
```

## fix.data *Fix data frame column and row names and remove symbols and blanks*

### Description

This function applies several data cleaning functions from the `fixr` package to a given data frame. The `fix_data_names`, `remove_spaces`, `remove_symbols_data`, and `replace_blanks_with_na` functions are used to add "X_" before column and row names that start with a number, remove leading/trailing spaces, remove non-alphanumeric characters from the data, replace spaces with underscores in column and row names, and replace empty string values with NAs, respectively.

### Usage

```
fix.data(df)
```

### Arguments

`df` A data frame to be processed.

### Value

The cleaned data frame.

### Examples

```
df <- data.frame(" 1st col " = c("", "foo", ""), "2nd col" = c(" ", " ", "bar"),
                 "3rd col" = c(1, 2, 3))
fix.data(df)
```

## `fix_blanks_with_na` *Replace blanks with NA in a data frame*

### Description

This function replaces all empty string values ("") in a given data frame with NA values.

### Usage

```
fix_blanks_with_na(df)
```

### Arguments

`df` A data frame to be processed.

### Value

The data frame with empty string values replaced with NAs.

### Examples

```
df <- data.frame(x = c("", "foo", ""), y = c("", "", "bar"), z = c(1, 2, 3))
fix_blanks_with_na(df)
```

## `fix_column_names` *Fix Column Names*

### Description

This function removes "X." or "X" from the beginning of column names and replaces any "." with "_". It also removes leading/trailing symbols and spaces, and ensures that there is only one underscore between two words. If there are duplicate column names, it appends a number to each duplicate column name to make it unique.

### Usage

```
fix_column_names(data)
```

### Arguments

`data` A data frame with improperly formatted column names.

### Value

The modified data frame with fixed column names.

### Examples

```
my_data <- data.frame(” Col1” = c(1, 2, 3), "Col.2" = c(4, 5, 6), check.names = FALSE)
fix_column_names(my_data)
```

## `fix_col_spaces` *Replace spaces in column names with underscores*

### Description

This function takes a data frame as an argument and replaces all spaces in the column names with underscores.

### Usage

```
fix_col_spaces(df)
```

### Arguments

`df` A data frame

### Value

A modified data frame with spaces in column names replaced by underscores.

### Examples

```
my_data <- data.frame("Column Name 1" = c(1, 2, 3), "Column Name 2" = c(4, 5, 6))

fix_col_spaces(my_data)
# Returns a data frame with column names where spaces are replaced by underscores.
```

## `fix_data_names` *Fix row and column names of a data frame*

### Description

This function fixes the row and column names of a data frame by removing leading and trailing spaces, replacing spaces with underscores, and modifying duplicate names.

### Usage

```
fix_data_names(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame to be fixed |

### Value

A fixed data frame with modified row and column names

### Examples

```
my_data <- data.frame(" Col1" = c(1, 2, 3), "Col.2" = c(4, 5, 6), check.names = FALSE)
rownames(my_data) <- c(" Row1", " Row.2", "Row.3 ")
fix_column_names(fix_row_names(my_data))
```

## `fix_duplicate_cols` *Remove duplicate columns from a data frame*

### Description

This function removes duplicate columns from a data frame.

### Usage

```
fix_duplicate_cols(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame |

### Value

A data frame with duplicate columns removed

### Examples

```
df <- data.frame(a = c(1, 1, 2), b = c(2, 2, 3))
fix_duplicate_cols(df)
```

## `fix_duplicate_rows` *Remove duplicate rows from a data frame*

### Description

This function removes duplicate rows from a data frame.

### Usage

```
fix_duplicate_rows(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame |

### Value

A data frame with duplicate rows removed

### Examples

```
df <- data.frame(a = c(1, 1, 2), b = c(2, 2, 3))
fix_duplicate_rows(df)
```

## fix_missing_alphanumeric_values

*Fill missing values in alphanumeric columns*

### Description

This function imputes missing values in alphanumeric columns of a data frame. If a column is numeric, missing values are imputed with the column mean. Otherwise, missing values are imputed with the column mode (most common value).

### Usage

```
fix_missing_alphanumeric_values(df)
```

### Arguments

| | |
|---|---|
| df | A data frame with missing values. |

### Value

A data frame with imputed missing values.

### Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", NA),
                 y = c(4, 5, -6, 4), z = c(7, 8, NA, 7))
fix_missing_alphanumeric_values(df)
```

## fix_missing_numeric_values

*fill_missing_numeric_values*

### Description

A function to fill missing values in numeric columns of a data frame with the mean of the column.

### Usage

```
fix_missing_numeric_values(df)
```

### Arguments

| | |
|---|---|
| df | A data frame with missing values. |

## Value

A data frame with missing numeric values filled with the column mean.

## Examples

```
df <- data.frame(w = c(7, 8, 180, 7), x = c("a", "b", "c", "d"),
                 y = c(4, 5, -6, 4), z = c(7, 8, NA, 7))
fix_missing_numeric_values(df)
```

---

# fix_outliers *Remove Outliers from a Data Frame*

---

## Description

This function removes outlier rows from a data frame by identifying rows with values that are more than 2 standard deviations away from the mean in any column.

## Usage

```
fix_outliers(df)
```

## Arguments

| | |
|---|---|
| df | A data frame to clean |

## Value

A cleaned data frame with outlier rows removed

## Examples

```
df <- data.frame(x = c(1,2,3,4,5,6,7,8,9,10),
                 y = c(1,1,1,1,1,1,1,100,1,1))
fix_outliers(df)
```

## `fix_row_names` *Fix row names of a data frame*

### Description

This function removes any leading "X." or "X" from the row names of a data frame, replaces any "." with "_", removes any leading or trailing symbols and spaces, and ensures that there is only one underscore between two words. Additionally, if there are duplicate row names, the function appends a number to each duplicate row name to make it unique.

### Usage

```
fix_row_names(data)
```

### Arguments

| | |
|---|---|
| `data` | a data frame with improperly formatted row names |

### Value

a modified data frame with fixed row names

### Examples

```
my_data <- data.frame(" Col1" = c(1, 2, 3), "Col.2" = c(4, 5, 6), check.names = FALSE)
rownames(my_data) <- c(" Row1", " Row.2", "Row.3 ")
fix_row_names(my_data)
```

## `fix_row_spaces` *Replace spaces in row names with underscores*

### Description

This function takes a data frame as an argument and replaces all spaces in the row names with underscores.

### Usage

```
fix_row_spaces(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame |

### Value

A modified data frame with spaces in row names replaced by underscores.

### Examples

```
my_data <- data.frame("Column Name 1" = c(1, 2, 3), "Column Name 2" = c(4, 5, 6))
rownames(my_data) <- c("Row Name 1", "Row Name 2", "Row Name 3")

fix_row_spaces(my_data)
# Returns a data frame with row names where spaces are replaced by underscores.
```

## fix_special_characters_in_data
*Remove Non-Alphanumeric Characters from Data Frame*

### Description

This function removes non-alphanumeric characters from all non-numeric columns in a data frame. The columns are modified in-place.

### Usage

```
fix_special_characters_in_data(df)
```

### Arguments

| | |
|---|---|
| `df` | A data frame. |

### Value

A modified data frame where all non-numeric columns have had non-alphanumeric characters removed.

### Examples

```
df <- data.frame(a = c("A*B", "C&D"), b = c("1.2", "3.4"))
fix_special_characters_in_data(df)
# Output:
#   a   b
# 1 AB  1.2
# 2 CD  3.4
```

## fix_special_characters_in_names

*Remove Special Characters from Data Frame Column and Row Names*

### Description

This function removes any non-alphanumeric characters from both the row and column names of a given data frame.

### Usage

```
fix_special_characters_in_names(df)
```

### Arguments

| | |
|---|---|
| df | A data frame with non-alphanumeric characters in the column or row names. |

### Value

A data frame with all non-alphanumeric characters removed from the column and row names.

### Examples

```
df <- data.frame("Col1!" = c(1, 2, 3), "Col2?" = c(4, 5, 6))
rownames(df) <- c("Row1@", "Row2#", "Row3$")
fix_special_characters_in_names(df)
```

# Index