

# Package ‘wrangle’

August 24, 2023

**Type** Package

**Title** A Systematic Data Wrangling Idiom

**Version** 0.6.3

**Author** Tim Bergsma

**Maintainer** Tim Bergsma <bergsmat@gmail.com>

**Description** Supports systematic scrutiny, modification, and integration of data. The function `status()` counts rows that have missing values in grouping columns (returned by `na()`), have non-unique combinations of grouping columns (returned by `dup()`), and that are not locally sorted (returned by `unsorted()`). Functions `enumerate()` and `itemize()` give sorted unique combinations of columns, with or without occurrence counts, respectively. Function `ignore()` drops columns in `x` that are present in `y`, and `informative()` drops columns in `x` that are entirely NA; `constant()` returns values that are constant, given a key. Data that have defined unique combinations of grouping values behave more predictably during merge operations.

**License** GPL-3

**BugReports** <https://github.com/bergsmat/wrangle/issues>

**Imports** dplyr (>= 1.0.2), tidyr, magrittr, rlang

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-08-24 03:00:02 UTC

## R topics documented:

<code>constant.data.frame</code> . . . . .	2
<code>dup.data.frame</code> . . . . .	3
<code>dupGroups.data.frame</code> . . . . .	4
<code>enumerate</code> . . . . .	4
<code>ignore</code> . . . . .	5
<code>informative</code> . . . . .	5

informative.data.frame . . . . .	6
itemize . . . . .	7
misplaced.data.frame . . . . .	7
na.data.frame . . . . .	8
naGroups.data.frame . . . . .	8
safe_join.data.frame . . . . .	9
sort.grouped_df . . . . .	10
static . . . . .	10
status.data.frame . . . . .	11
unsorted.data.frame . . . . .	12
weak.data.frame . . . . .	12

## Index 14

---

constant.data.frame    *Identify Constant Features of a Data Frame*

---

### Description

Returns columns of a data.frame whose values do not vary within subsets defined by columns named in .... Defaults to groups(x) if none supplied, or all columns otherwise.

### Usage

```
## S3 method for class 'data.frame'
constant(x, ...)
```

### Arguments

x                    object  
 ...                optional grouping columns (named arguments are ignored)

### Value

data.frame (should be same class as x)

### See Also

Other constant: [constant\(\)](#)

### Examples

```
library(dplyr)
constant(Theoph)                    # data frame with 0 columns and 1 row
constant(Theoph, Subject)        # Subject Wt Dose Study
Theoph$Study <- 1
constant(Theoph)                   # Study
constant(Theoph, Study)            # Study
constant(Theoph, Study, Subject)  # Subject Wt Dose Study
```

```
Theoph <- group_by(Theoph, Subject)
constant(Theoph)           # Subject Wt Dose Study
constant(Theoph, Study)    # Study
```

---

dup.data.frame	<i>Show records with duplicate or duplicated values of grouping variables.</i>
----------------	--

---

## Description

Shows records with duplicate or duplicated values of grouping variables.

## Usage

```
## S3 method for class 'data.frame'
dup(x, ...)
```

## Arguments

x	data.frame
...	optional grouping columns (named arguments are ignored)

## Value

data.frame

## See Also

Other dup: [dup\(\)](#)

## Examples

```
library(dplyr)
dupGroups(mtcars)
dupGroups(group_by(mtcars, mpg))
dup(group_by(mtcars, mpg))
```

---

`dupGroups.data.frame` *Index records with with duplicate or duplicated values of grouping variables.*

---

### Description

Indexes records with with duplicate or duplicated values of grouping variables. If b follows a and and is the same, then b is a duplicate, a is duplicated, and both are shown.

### Usage

```
## S3 method for class 'data.frame'
dupGroups(x, ...)
```

### Arguments

x                    data.frame  
 ...                  optional grouping columns (named arguments are ignored)

### Value

grouped\_df  
 logical

### See Also

Other dupGroups: [dupGroups\(\)](#)

---

`enumerate`                    *Count unique combinations of items in specified columns.*

---

### Description

Counts unique combinations of items in specified columns (unquoted).

### Usage

```
enumerate(x, ...)
```

### Arguments

x                    data.frame  
 ...                  columns to show

**Value**

grouped\_df

**See Also**

Other util: [detect\(\)](#), [itemize\(\)](#), [static\(\)](#)

**Examples**

```
enumerate(mtcars, cyl, gear, carb)
```

---

ignore	<i>Drop columns in x that are present in y.</i>
--------	---

---

**Description**

Drops columns in x that are present in y.

**Usage**

```
ignore(x, y, ...)
```

**Arguments**

x	data.frame
y	data.frame
...	ingored

**Value**

data.frame

---

informative	<i>Drop columns in x that are entirely NA.</i>
-------------	--

---

**Description**

Drops columns in x that are entirely NA.

**Usage**

```
informative(x, ...)
```

**Arguments**

x                    object of dispatch  
...                   passed

**See Also**

[informative.data.frame](#)

Other informative: [informative.data.frame\(\)](#)

**Examples**

```
head(Theoph)  
Theoph$Dose <- NA  
head(informative(Theoph))
```

---

informative.data.frame

*Drop columns in x that are entirely NA.*

---

**Description**

Drops columns in x that are entirely NA.

**Usage**

```
## S3 method for class 'data.frame'  
informative(x, ...)
```

**Arguments**

x                    data.frame  
...                   ingored

**Value**

data.frame

**See Also**

Other informative: [informative\(\)](#)

---

itemize	<i>Show unique combinations of items in specified columns</i>
---------	---

---

**Description**

Shows unique combinations of items in specified columns (unquoted).

**Usage**

```
itemize(x, ...)
```

**Arguments**

x	data.frame
...	columns to show

**Value**

grouped\_df

**See Also**

Other util: [detect\(\)](#), [enumerate\(\)](#), [static\(\)](#)

**Examples**

```
itemize(mtcars, cyl, gear, carb)
```

---

<code>misplaced.data.frame</code>	<i>Index records whose relative positions would change if sorted.</i>
-----------------------------------	---

---

**Description**

Indexes records whose relative positions would change if sorted, i.e. records that would not have the same nearest neighbors (before and after). `unsorted()` returns the records corresponding to this index.

**Usage**

```
## S3 method for class 'data.frame'  
misplaced(x, ...)
```

**Arguments**

x	data.frame
...	optional grouping columns (named arguments are ignored)

**Value**

logical with length nrow(x)

**See Also**

[na dup](#)

Other unsorted: [misplaced\(\)](#), [unsorted.data.frame\(\)](#), [unsorted\(\)](#)

---

na.data.frame	<i>Show records with NA values of grouping variables.</i>
---------------	---

---

**Description**

Shows records with NA values of grouping variables.

**Usage**

```
## S3 method for class 'data.frame'  
na(x, ...)
```

**Arguments**

x	data.frame
...	optional grouping columns (named arguments are ignored)

**Value**

data.frame

**See Also**

Other na: [na\(\)](#)

---

naGroups.data.frame	<i>Index records with NA values of grouping variables.</i>
---------------------	--

---

**Description**

Indexes records with NA values of grouping variables.

**Usage**

```
## S3 method for class 'data.frame'  
naGroups(x, ...)
```



**Arguments**

x data.frame  
... optional grouping columns (named arguments are ignored)

**Value**

logical

**See Also**

Other naGroups: [naGroups\(\)](#)

---

safe\_join.data.frame *Join Data Frames Safely*

---

**Description**

Joins data frames safely. I.e., a left join that cannot alter row order or number. Supports the case where you only intend to augment existing rows with additional columns and are expecting singular matches. Gives an error if row order or number would have been altered by a left join.

**Usage**

```
## S3 method for class 'data.frame'  
safe_join(x, y, ...)
```

**Arguments**

x data.frame  
y data.frame  
... passed to `dplyr::left_join`

**See Also**

Other safe\_join: [safe\\_join\(\)](#)

**Examples**

```
library(magrittr)  
x <- data.frame(code = c('a','b','c'), value = c(1:3))  
y <- data.frame(code = c('a','b','c'), roman = c('I','II','III'))  
x %>% safe_join(y)  
try(  
  x %>% safe_join(rbind(y,y))  
)
```

---

<code>sort.grouped_df</code>	<i>Arrange by groups.</i>
------------------------------	---------------------------

---

### Description

As of 0.5, `dplyr::arrange` ignores groups. This function gives the old behavior as a method for generic `base::sort`. Borrowed from Ax3man at <https://github.com/hadley/dplyr/issues/1206>.

### Usage

```
## S3 method for class 'grouped_df'
sort(x, decreasing = FALSE, ...)
```

### Arguments

<code>x</code>	<code>grouped_df</code>
<code>decreasing</code>	logical (ignored)
<code>...</code>	further sort criteria

### Value

`grouped_df`

### Examples

```
library(dplyr)
head(sort(group_by(Theoph, Subject, Time)))
```

---

<code>static</code>	<i>Find unique records for subset of columns with one unique value.</i>
---------------------	---

---

### Description

Finds unique records for subset of columns with one unique value.

### Usage

```
static(x, ...)
```

### Arguments

<code>x</code>	<code>data.frame</code>
<code>...</code>	ignored

**Value**

data.frame

**See Also**

Other util: [detect\(\)](#), [enumerate\(\)](#), [itemize\(\)](#)

---

status.data.frame      *Report status with respect to grouping variables.*

---

**Description**

Reports status with respect to grouping variables.

**Usage**

```
## S3 method for class 'data.frame'  
status(x, ...)
```

**Arguments**

x	data.frame
...	optional grouping columns (named arguments are ignored)

**Value**

returns x invisibly (as originally grouped)

**See Also**

[na dup unsorted informative ignore itemize enumerate sort.grouped\\_df](#)

Other status: [status\(\)](#)

**Examples**

```
library(dplyr)  
status(Theoph)  
status(Theoph, Subject)  
status(group_by(Theoph, Subject, Time))
```

---

unsorted.data.frame     *Extract records whose relative positions would change if sorted.*

---

### Description

Extracts records whose relative positions would change if sorted, i.e. records that would not have the same nearest neighbors (before and after). `misplaced()` returns the index that extracts these records.

### Usage

```
## S3 method for class 'data.frame'
unsorted(x, ...)
```

### Arguments

`x`                    data.frame  
`...`                optional grouping columns (named arguments are ignored)

### Value

data.frame, possibly grouped\_df

### See Also

[na dup](#)

Other unsorted: [misplaced.data.frame\(\)](#), [misplaced\(\)](#), [unsorted\(\)](#)

---

weak.data.frame     *Show records with NA, duplicate or duplicated values of grouping variables.*

---

### Description

Shows records with NA, duplicate or duplicated values of grouping variables.

### Usage

```
## S3 method for class 'data.frame'
weak(x, ...)
```

### Arguments

`x`                    data.frame  
`...`                optional grouping columns (named arguments are ignored)

**Value**

data.frame

**See Also**

Other weak: [weak\(\)](#)

# Index

- \* **constant**
    - constant.data.frame, 2
  - \* **dupGroups**
    - dupGroups.data.frame, 4
  - \* **dup**
    - dup.data.frame, 3
  - \* **ignore**
    - ignore, 5
  - \* **informative**
    - informative, 5
    - informative.data.frame, 6
  - \* **naGroups**
    - naGroups.data.frame, 8
  - \* **na**
    - na.data.frame, 8
  - \* **safe\_join**
    - safe\_join.data.frame, 9
  - \* **sort**
    - sort.grouped\_df, 10
  - \* **status**
    - status.data.frame, 11
  - \* **unsorted**
    - misplaced.data.frame, 7
    - unsorted.data.frame, 12
  - \* **util**
    - enumerate, 4
    - itemize, 7
    - static, 10
  - \* **weak**
    - weak.data.frame, 12
- constant, 2  
constant.data.frame, 2
- detect, 5, 7, 11  
dup, 3, 8, 11, 12  
dup.data.frame, 3  
dupGroups, 4  
dupGroups.data.frame, 4
- enumerate, 4, 7, 11  
ignore, 5, 11  
informative, 5, 6, 11  
informative.data.frame, 6, 6  
itemize, 5, 7, 11
- misplaced, 8, 12  
misplaced.data.frame, 7, 12
- na, 8, 11, 12  
na.data.frame, 8  
naGroups, 9  
naGroups.data.frame, 8
- safe\_join, 9  
safe\_join.data.frame, 9  
sort.grouped\_df, 10, 11  
static, 5, 7, 10  
status, 11  
status.data.frame, 11
- unsorted, 8, 11, 12  
unsorted.data.frame, 8, 12
- weak, 13  
weak.data.frame, 12