

# Package ‘fastDummies’

September 16, 2020

**Type** Package

**Title** Fast Creation of Dummy (Binary) Columns and Rows from  
Categorical Variables

**Version** 1.6.2

**Description** Creates dummy columns from columns that have categorical variables (character or factor types). You can also specify which columns to make dummies out of, or which columns to ignore. Also creates dummy rows from character, factor, and Date columns. This package provides a significant speed increase from creating dummy variables through `model.matrix()`.

**Depends** R (>= 2.10)

**Imports** data.table, tibble, stringr

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/jacobkap/fastDummies>

**BugReports** <https://github.com/jacobkap/fastDummies/issues>

**RoxygenNote** 7.1.1

**Suggests** testthat (>= 2.1.0), knitr, rmarkdown, covr, spelling

**VignetteBuilder** knitr

**Language** en-US

**NeedsCompilation** no

**Author** Jacob Kaplan [aut, cre] (<<https://orcid.org/0000-0002-0601-0387>>),  
Benjamin Schlegel [ctb]

**Maintainer** Jacob Kaplan <jkkaplan6@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-09-16 21:10:09 UTC

## R topics documented:

<code>dummy_cols</code> . . . . .	2
<code>dummy_columns</code> . . . . .	3
<code>dummy_rows</code> . . . . .	4

---

dummy_cols	<i>Fast creation of dummy variables</i>
------------	---

---

## Description

Quickly create dummy (binary) columns from character and factor type columns in the inputted data (and numeric columns if specified.) This function is useful for statistical analysis when you want binary columns rather than character columns.

## Usage

```
dummy_cols(
  .data,
  select_columns = NULL,
  remove_first_dummy = FALSE,
  remove_most_frequent_dummy = FALSE,
  ignore_na = FALSE,
  split = NULL,
  remove_selected_columns = FALSE
)
```

## Arguments

<code>.data</code>	An object with the data set you want to make dummy columns from.
<code>select_columns</code>	Vector of column names that you want to create dummy variables from. If NULL (default), uses all character and factor columns.
<code>remove_first_dummy</code>	Removes the first dummy of every variable such that only n-1 dummies remain. This avoids multicollinearity issues in models.
<code>remove_most_frequent_dummy</code>	Removes the most frequently observed category such that only n-1 dummies remain. If there is a tie for most frequent, will remove the first (by alphabetical order) category that is tied for most frequent.
<code>ignore_na</code>	If TRUE, ignores any NA values in the column. If FALSE (default), then it will make a dummy column for value_NA and give a 1 in any row which has a NA value.
<code>split</code>	A string to split a column when multiple categories are in the cell. For example, if a variable is Pets and the rows are "cat", "dog", and "turtle", each of these pets would become its own dummy column. If one row is "cat, dog", then a split value of "," this row would have a value of 1 for both the cat and dog dummy columns.
<code>remove_selected_columns</code>	If TRUE (not default), removes the columns used to generate the dummy columns.

**Value**

A data.frame (or tibble or data.table, depending on input data type) with same number of rows as inputted data and original columns plus the newly created dummy columns.

**See Also**

[dummy\\_rows](#) For creating dummy rows

Other dummy functions: [dummy\\_columns\(\)](#), [dummy\\_rows\(\)](#)

**Examples**

```
crime <- data.frame(city = c("SF", "SF", "NYC"),
  year = c(1990, 2000, 1990),
  crime = 1:3)
dummy_cols(crime)
# Include year column
dummy_cols(crime, select_columns = c("city", "year"))
# Remove first dummy for each pair of dummy columns made
dummy_cols(crime, select_columns = c("city", "year"),
  remove_first_dummy = TRUE)
```

---

dummy\_columns

*Fast creation of dummy variables*


---

**Description**

`dummy_columns()` quickly creates dummy (binary) columns from character and factor type columns in the inputted data. This function is useful for statistical analysis when you want binary columns rather than character columns.

**Usage**

```
dummy_columns(
  .data,
  select_columns = NULL,
  remove_first_dummy = FALSE,
  remove_most_frequent_dummy = FALSE,
  ignore_na = FALSE,
  split = NULL,
  remove_selected_columns = FALSE
)
```

**Arguments**

`.data` An object with the data set you want to make dummy columns from.

`select_columns` Vector of column names that you want to create dummy variables from. If NULL (default), uses all character and factor columns.

<code>remove_first_dummy</code>	Removes the first dummy of every variable such that only n-1 dummies remain. This avoids multicollinearity issues in models.
<code>remove_most_frequent_dummy</code>	Removes the most frequently observed category such that only n-1 dummies remain. If there is a tie for most frequent, will remove the first (by alphabetical order) category that is tied for most frequent.
<code>ignore_na</code>	If TRUE, ignores any NA values in the column. If FALSE (default), then it will make a dummy column for value_NA and give a 1 in any row which has a NA value.
<code>split</code>	A string to split a column when multiple categories are in the cell. For example, if a variable is Pets and the rows are "cat", "dog", and "turtle", each of these pets would become its own dummy column. If one row is "cat, dog", then a split value of "," this row would have a value of 1 for both the cat and dog dummy columns.
<code>remove_selected_columns</code>	If TRUE (not default), removes the columns used to generate the dummy columns.

### See Also

[dummy\\_rows](#) For creating dummy rows

Other dummy functions: [dummy\\_cols\(\)](#), [dummy\\_rows\(\)](#)

### Examples

```
crime <- data.frame(city = c("SF", "SF", "NYC"),
  year = c(1990, 2000, 1990),
  crime = 1:3)
dummy_cols(crime)
# Include year column
dummy_cols(crime, select_columns = c("city", "year"))
# Remove first dummy for each pair of dummy columns made
dummy_cols(crime, select_columns = c("city", "year"),
  remove_first_dummy = TRUE)
```

---

dummy\_rows

*Fast creation of dummy rows*

---

### Description

`dummy_rows()` quickly creates dummy rows to fill in missing rows based on all combinations of available character, factor, and date columns (if not otherwise specified). This is useful for creating balanced panel data. Columns that are not character, factor, or dates are filled in with NA (or whatever value you specify).

**Usage**

```
dummy_rows(  
  .data,  
  select_columns = NULL,  
  dummy_value = NA,  
  dummy_indicator = FALSE  
)
```

**Arguments**

.data	An object with the data set you want to make dummy columns from.
select_columns	If NULL (default), uses all character, factor, and Date columns to produce categories to make the dummy rows by. If not NULL, you manually enter a string or vector of strings of columns name(s).
dummy_value	Value of the row for columns that are not selected. Default is a value of NA.
dummy_indicator	Adds binary column to say if row is dummy or not (i.e. included in original data or not)

**Value**

A data.frame (or tibble or data.table, depending on input data type) with same number of columns as inputted data and original rows plus the newly created dummy rows

**See Also**

[dummy\\_cols](#) For creating dummy columns

Other dummy functions: [dummy\\_cols\(\)](#), [dummy\\_columns\(\)](#)

**Examples**

```
crime <- data.frame(city = c("SF", "SF", "NYC"),  
  year = c(1990, 2000, 1990),  
  crime = 1:3)  
  
dummy_rows(crime)  
# Include year column  
dummy_rows(crime, select_columns = c("city", "year"))  
# m=Make dummy value 0  
dummy_rows(crime, select_columns = c("city", "year"),  
  dummy_value = 0)  
# Add a dummy indicator  
dummy_rows(crime, select_columns = c("city", "year"),  
  dummy_indicator = TRUE)
```

# Index

## \* **dummy functions**

- dummy\_cols, [2](#)
- dummy\_columns, [3](#)
- dummy\_rows, [4](#)

- dummy\_cols, [2](#), [4](#), [5](#)
- dummy\_columns, [3](#), [3](#), [5](#)
- dummy\_rows, [3](#), [4](#), [4](#)