Package: speedycode (via r-universe)

August 22, 2024

Version 0.3.0

Date 2022-03-30

Title Automate Code for Adding Labels, Recoding and Renaming Variables, and Converting ASCII Files

Author Jacob Harris [aut, cre]

Suggests labelled, readroper

Description Label, recode, rename, and convert datasets and ASCII files more efficiently. 'speedycode' automates the code necessary for labeling variables with the 'labelled' package, recoding and renaming variables with 'dplyr' syntax, and converting ASCII files with the 'readroper' package. Most functions require only the name of the dataset and the code will be automatically written. Some convenience functions useful for converting ASCII files are also included.

License GPL-3

Imports dplyr, stringr, purrr

Encoding UTF-8

NeedsCompilation no

Maintainer Jacob Harris <jh2689@cornell.edu>

Date/Publication 2022-03-31 06:40:05 UTC

Repository https://jacobwharris.r-universe.dev

RemoteUrl https://github.com/cran/speedycode

RemoteRef HEAD

RemoteSha 4f3724df9c366ee278e46d3a72c0bcc3f6bc1cdf

Contents

`debug_ascii` *Debug errors when converting ASCII files with the readroper package*

Description

A common issue when converting ASCII files is getting the column positions, variable widths, and new variable names to align. A simple way to debug errors is to compare the lengths of each which must be equivalent to convert ASCII files with the readroper package. debug_ascii calculates the lengths of each of these inputs so you can quickly diagnose errors. Each argument can be copied directly from the 'read_rpr' function within the readroper package.

Usage

```
debug_ascii(col_positions_input, widths_input, col_names_input)
```

Arguments

`col_positions_input`	The col_positions argument in the read_rpr function
`widths_input`	The widths argument in the read_rpr function
`col_names_input`	The col_names argument in the read_rpr function

Value

A dataframe containing the lengths of the col_positions, widths, and col_names arguments

Author(s)

Jacob Harris

Examples

```
## Not run: For ease of replicability, the examples here
come from generated data rather than data from
an ASCII file.
## End(Not run)

col_positions <- c(1, 2, 3, 5, 8)
widths <- c(1, 1, 2, 3, 1)
col_names <- c("Q1", "Q2", "Q3", "Q4", "Q5")

debug_ascii(col_positions, widths, col_names)
```

```
## Not run: Now, if the lengths differ from an error
(see the missing "Q4" variable), the function
will throw a warning.
## End(Not run)

col_positions <- c(1, 2, 3, 5, 8)
widths <- c(1, 1, 2, 3, 1)
col_names <- c("Q1", "Q2", "Q3", "Q5")

debug_ascii(col_positions, widths, col_names)
```

---

speedy_classes *Automate code for changing variable classes*

Description

'speedy_classes' automates the code for changing variable classes for many variables at time. The default is for each variable to be saved back to the its original class so the new classes are the only required input. Variables that do not need to be changed may be kept or removed from the code.

Usage

```
speedy_classes(data, path = "")
```

Arguments

data	name of dataset
path	If saving code to a new R script file, specify the file path here. Leave blank if not saving code.

Details

'speedy_classes' automates the code for changing the classes of a large number of variables at once. The code may be copied and pasted from the console or saved out to a separate R script. The 'dplyr' package is not required to run the package, but the automated code uses 'dplyr' syntax so you will need to load it to run the code.

Value

Formatted code written with 'dplyr' syntax for changing the classes of all variables in a dataset. The automated code maintains the original class for each variable so the only required input is a different class for the variables that need it. You can run the entire code chuck created by 'speedy_classes' without changing anything in the generated code.

Author(s)

Jacob Harris

Examples

```
speedy_classes(data = iris)

## Not run: speedy_classes(data = iris, path = "~/INPUT-FILE-PATH")
```

`speedy_labels` *Automate code for labelling variables and values with the 'labelled' package*

Description

'speedy_labels' automates the code for labelling variables and values. With 'speedy_labels', all the code is automatically written other than the new labels. It is primarily designed for working with survey data but can be used for any data that requires labels.

Usage

```
speedy_labels(data, nrows = 5, path = "")
```

Arguments

`data`	name of dataset
`nrows`	Number of rows for value labels of each variable. The minimum number of rows allowed is 2 and the maximum is 10. Extra rows without values are set to NA.
`path`	If saving code to a new R script file, specify the file path here. Leave blank if not saving code.

Details

'speedy_labels' automatically writes all the code necessary to add labels to a dataset. The code may be copied and pasted from the console or saved out to a separate R script. The 'dplyr' and 'labelled' packages are not required to run the package, but the automated code uses syntax from these packages so you will need to load them to run the code.

Value

Formatted code written with 'dplyr' syntax for labelling variables with the 'labelled' package. The automated code maintains the original class for each variable so the only required input is a different class for the variables that need it. Replace the final comma with a parenthese and press "Cmd + I" (or Ctrl + I for PC users) to format the code indentations.

Note

For ease, the iris dataset is used as an example. However, the usage of 'speedy_labels' is more intuitive with actual survey data with categorical responses since it usually doesn't make sense to label continuous values.

Author(s)

Jacob Harris

See Also

This function is particularly useful for adding labels to data files that were converted from ASCII format. Click here to learn more about ASCII converions in R.

Examples

```
speedy_labels(iris, nrows = 5)

## Not run: speedy_labels(data = iris, nrows = 5, path = "~/INPUT-FILE-PATH")
```

`speedy_mutate` *Automate code for recoding variables with the 'mutate' and 'case_when' functions*

Description

'speedy_mutate' automates the code for quickly recoding variables with a large number of unique levels with 'dplyr' syntax. The user only needs to supply the variable to recode and whether or not those variables should be quoted or not.

Usage

```
speedy_mutate(data, var, var_classes = "sn", path = "")
```

Arguments

`data`	Name of dataset
`var`	String of the name of the variable being recoded
`var_classes`	Specifies whether or not the current variable and the new variable being created should have quotes around them. There are four possible inputs ("ss", "sn", "nn", "ns"). "ss" means the current and new variable with both have quotes. "sn" means the first will have quotes and the second will not and so forth.
`path`	If saving code to a new R script file, specify the file path here. Leave blank if not saving code.

Details

'speedy_mutate' generates a formatted chunk of the code for creating a new variable using the 'mutate' and 'case_when' functions. The code may be copied and pasted from the console or saved out to a separate R script. This is useful when a new variable needs to be created with many different levels based on the values in another variable.

Value

Formatted code written with 'dplyr' syntax for recoding variables with 'mutate' and 'case_when'.

Examples

```
## Not run: A simple applications is to add geographical FIPS codes to U.S. states

states <- as.data.frame(state.abb)
speedy_mutate(data = states, var = "state.abb")
```

`speedy_rename` *Automate code for renaming variables*

Description

'speedy_rename' automates the code for renaming variables. With 'speedy_rename', all the code is automatically written with dplyr syntax. The user only needs to provide the new variable names.

Usage

```
speedy_rename(data, path = "")
```

Arguments

`data`	name of dataset
`path`	If saving code to a new R script file, specify the file path here. Leave blank if not saving code.

Details

'speedy_rename' automatically writes all the code necessary to rename a large number of variables at once with the exception of the new variable names. The code may be copied and pasted from the console or saved out to a separate R script. The dplyr package is not required to run the package, but the automated code uses 'dplyr' syntax so you will need to load it to run the code.

Value

Formatted code written with 'dplyr' syntax for renaming all the variables in a dataset which may be copied and pasted from the console or saved out to a separate R script.

Author(s)

Jacob Harris <jh2689@cornell.edu>

Examples

```
speedy_rename(data = iris)

## Not run: speedy_rename(data = iris, path = "~/INPUT-FILE-PATH")
```

`speedy_varnames` *Quickly generate a vector of new variable names*

Description

speedy_varnames generates a vector of new, generic variable names beginning with a given character value through the number of new names desired. This is espeically useful for converting ASCII files when a large number of new variable names must be created.

Usage

```
speedy_varnames(prefix = "Q", first_number = 1, last_number = 25)
```

Arguments

`prefix`	A character value to precede all of the new variable names
`first_number`	The beginning number of the new variable name vector
`last_number`	The final number of the new variable name vector

Value

A vector of new variable names

Note

If using this function for ASCII conversions, you can paste the new vector of names into the col_names argument of the 'read_rpr' function.

Author(s)

Jacob Harris

Examples

```
speedy_varnames("Q", 1, 25)
```

Index