Package 'blaise'

August 18, 2023

Type Package

Title Read and Write FWF Files in the 'Blaise' Format

Version 1.3.10

Description Can be used to read and write a fwf with an accompanying 'Blaise' datamodel.
Blaise is the software suite built by Statistics Netherlands (CBS). It is essentially a
way to write and collect surveys and perform statistical analysis on the data. It stores its data in
fixed width format with an accompanying metadata file, this is the Blaise format. The package automatically
interprets this metadata and reads the file into an R dataframe.
When supplying a datamodel for writing, the dataframe will be automatically converted
to that format and checked for compatibility.
Supports dataframes, tibbles and LaF objects.
For more information about 'Blaise', see <https://blaise.com/products/general-information>.

License GPL-3

Encoding UTF-8

Imports dplyr (>= 0.7.2), readr (>= 1.1.1), stringr (>= 1.2.0), utils (>= 3.4.1), tibble (>= 1.3.3), tools (>= 3.4.1), methods (>= 3.4.1), stats (>= 3.4.1)

Suggests testthat, LaF (>= 0.6.3), knitr, rmarkdown

RoxygenNote 7.2.3

Collate 'clean_model.R' 'generics.R' 'utils.R' 'variable.R' 'model.R' 'convert_df.R' 'convert_type.R' 'get_model.R' 'read_custom_types.R' 'read_data.R' 'read_data_laf.R' 'read_fwf_blaise.R' 'read_model.R' 'variable_custom.R' 'variable_date.R' 'variable_dummy.R' 'variable_enum.R' 'variable_integer.R' 'variable_real.R' 'variable_string.R' 'write_data.R' 'write_datamodel.R' 'write_fwf_blaise.R' 'write_fwf_blaise_with_model.R'

VignetteBuilder knitr

NeedsCompilation no

Author Sjoerd Ophof [aut, cre]

Maintainer Sjoerd Ophof <sjoerd.ophof@gmail.com>

Repository CRAN

Date/Publication 2023-08-18 14:52:32 UTC

R topics documented:

`read_fwf_blaise` *Read a fixed width datafile using a blaise datamodel*

Description

Use this function to read a fwf that is described by a blaise datamodel. If this function throws a warning, try using readr::problems() on the result, this will for instance show an error in the used locale.

Usage

```
read_fwf_blaise(
  datafile,
  modelfile,
  locale = readr::locale(),
  numbered_enum = TRUE,
  output = "data.frame"
)
```

Arguments

`datafile`	the fwf file containing the data
`modelfile`	the datamodel describing the data
`locale`	locale as specified with readr::locale(). Uses "." as default decimal separator. Can be used to change decimal separator, date_format, timezone, encoding, etc.
`numbered_enum`	use actual labels instead of numbers for enums that use non- standard numbering in the datamodel. With the default (TRUE) (Male (1), Female (2), Unknown (9)) will be read as a factor with labels (1, 2, 9). With FALSE it will be read as a factor (Male, Female, Unknown). beware that writing a dataframe read with FALSE will result in an enum with levels (1, 2, 3) unless overruled by an existing model, since R does not support custom numbering for factors.

`output` Define which output to use. Either "data.frame" (default) or "LaF". LaF does not support Datetypes, so these are converted to character vectors. Using LaF, DUMMY variables also can't be ignored, these are read as empty character vectors. Using LaF basically takes over the parsing of the datamodel from LaF, since this is more robust and accepts more types of input.

Details

Handles the following types:

- STRING
- INTEGER
- REAL
- DATETYPE
- ENUM (if numbered it will be converted to a factor with the numbers as labels)
- custom types (same as a numbered ENUM)

If you want the numbered enums to be converted to their labels, this is possible by changing the "numbered_enum" parameter

Examples

```
model = "
DATAMODEL Test
FIELDS
A     : STRING[1]
B     : INTEGER[1]
C     : REAL[3,1]
D     : REAL[3]
E     : (Male, Female)
F     : 1..20
G     : 1.00..100.00
ENDMODEL
"
data =
"A12.3.121 1  1.00
B23.41.2210 20.20
C34.512.120100.00"

blafile = tempfile('testbla', fileext = '.bla')
writeLines(model, con = blafile)
datafile = tempfile('testdata', fileext = '.asc')
writeLines(data, con = datafile)

df = read_fwf_blaise(datafile, blafile)
unlink(blafile)
unlink(datafile)
```

`write_fwf_blaise` *Write a fixed width ascii datafile and accompanying blaise datamodel*

Description

Write a datafile in the blaise format (fwf ascii without separators) will always write out a blaise datamodel describing the datafile as well

Usage

```
write_fwf_blaise(
  df,
  output_data,
  output_model = NULL,
  decimal.mark = ".",
  digits = getOption("digits"),
  justify = "right",
  write_model = TRUE,
  model_name = NULL
)
```

Arguments

`df`	dataframe to write
`output_data`	path and name to output datafile. Will add .asc if no extension
`output_model`	path and name to output datamodel. If NULL will use the same name as output_data with .bla extension.
`decimal.mark`	decimal mark to use. Default is ".".
`digits`	how many significant digits are to be used for numeric and complex x. The default uses getOption("digits"). This is a suggestion: enough decimal places will be used so that the smallest (in magnitude) number has this many significant digits.
`justify`	direction of padding for STRING type when data is smaller than the width. Defaults to right-justified (padded on the left), can be "left", "right" or "centre".
`write_model`	logical that can be used to disable the automatic writing of a datamodel
`model_name`	Custom name that can be given to the datamodel. Default is the name of the dataframe

Details

Currently supports the following dataformats:

- character => STRING,
- integer => INTEGER,
- numeric => REAL,

- Date => DATETYPE,
- factor => ENUM (will convert factor with numbers as labels to STRING)
- logical => INTEGER

Value

output as it is written to file as a character vector. Does so invisibly, will not print but can be assigned.

Examples

```
datafilename = tempfile('testdata', fileext = '.asc')
blafilename = tempfile('testbla', fileext = '.bla')
data = data.frame(1, 1:10, sample(LETTERS[1:3], 10, replace = TRUE), runif(10, 1, 10))
write_fwf_blaise(data, datafilename)
unlink(c(datafilename, blafilename))
```

---

write_fwf_blaise_with_model

Write a fixed width ascii datafile based on a given blaise datamodel

---

Description

Write a datafile in the blaise format (fwf ascii without separators) using an existing datamodel. will not write out a datamodel unless explicitly asked to. Tries to automatically match columnns by name using Levenshtein distance and will change types if required and possible.

Usage

```
write_fwf_blaise_with_model(
  df,
  output_data,
  input_model,
  output_model = NULL,
  decimal.mark = ".",
  digits = getOption("digits"),
  justify = "right",
  max.distance = 0L
)
```

Arguments

df	dataframe to write
output_data	path and name to output datafile. Will add .asc if no extension
input_model	the datamodel used to convert the dataframe and write the output

output_model	path and name to output datamodel. If NULL will not write anything. default is NULL
decimal.mark	decimal mark to use. Default is ".".
digits	how many significant digits are to be used for numeric vectors. The default uses getOption("digits"). This is a suggestion: enough decimal places will be used so that the smallest (in magnitude) number has this many significant digits.
justify	direction of padding for STRING type when data is smaller than the width. Defaults to right-justified (padded on the left), can be "left", "right" or "centre".
max.distance	maximum Levenshtein distance to match columns. ignores case changes. Set to 0 (default) to only accept exact matches ignoring case. 4 appears to be a good number in general. Will prevent double matches and will pick te best match for each variable in the datamodel.

Value

output as it is written to file as a character vector. Does so invisibly, will not print but can be assigned.

Examples

```
datafilename = tempfile('testdata', fileext = '.asc')
blafilename = tempfile('testbla', fileext = '.bla')

model = "
DATAMODEL Test
FIELDS
A     : STRING[1]
B     : INTEGER[1]
C     : REAL[3,1]
D     : REAL[3]
E     : (Male, Female)
F     : 1..20
G     : 1.00..100.00
H     : DATETYPE
ENDMODEL
"
writeLines(model, con = blafilename)

  df = data.frame(
list(
  A = rep('t',3),
  B = 1:3,
  C = 1.1:3.3,
  D = 1.0:3.0,
  E = factor(c(1,2,1), labels = c('Male', 'Female')),
  F = 1:3,
  G = c(1., 99.9, 78.5),
  H = as.Date(rep('2001-01-01', 3))
)
)
```

```
write_fwf_blaise_with_model(df, datafilename, blafilename)


model = "
DATAMODEL Test
FIELDS
A     : STRING[1]
B     : STRING[1]
C     : STRING[3]
E     : STRING[1]
H     : STRING[8]
ENDMODEL
"
writeLines(model, con = blafilename)

df = data.frame(
list(
A = rep('t',3),
E = factor(c(1,2,1), labels = c('Male', 'Female')),
B = 1:3,
C = 1.1:3.3,
H = as.Date(rep('2001-01-01', 3))
),
stringsAsFactors = FALSE
)
write_fwf_blaise_with_model(df, datafilename, blafilename)
```

Index