Package 'StataDCTutils'

April 23, 2013

Type Package

Title Converts Stata dictionary files to formats more meaningful for R users

Version 1.1.1

Date 2013-01-16

Author Ananda Mahto

Maintainer Ananda Mahto <mrdwab@gmail.com>

Description Some datasets are distributed as a combination of a fixed width data file, a Stata '.dct' dictionary file, and a Stata '.do' file. The dictionary usually includes details like the variable name, variable label, variable start and end position in the fixed width data file, and the storage type of the variable. The functions and utilities in this package parse such dictionary files and attempt to convert the data into more usable formats for R users.

Depends R (>= 2.10)

License GPL-2

R topics documented:

StataDCTutils-package *Parses Stata dictionary files for further use with R*

Description

Utilities to make reading fixed-width-format datasets distributed as a `.dat/.dct/.do` set of Stata files more convenient with R.

Details

Package:	StataDCTutils
Type:	Package
Version:	1.0
Date:	2013-01-16
License:	CC-SA

Author(s)

Ananda Mahto

Maintainer: Ananda Mahto <mrdwab@gmail.com>

References

Initial versions of this function can be found at http://stackoverflow.com/questions/14224321/reading-dat-and-dct-directly-from-r

csvkit.fwf2csv *Convenience function to create a csv file from a fixed-width file*

Description

This is purely a convenience function to use the start and width definitions from a dictionary file to convert a fixed-width file to a csv file using `in2csv` from `csvkit` using a `system` call.

Usage

```
csvkit.fwf2csv(datafile, schema, output)
```

Arguments

datafile	The name of the flat data file (optionally including the path if the file is not in the working directory).
schema	The name of the schema file (perhaps generated using dct.parser and csvkit.schema) that defines the variable names, start positions, and column widths (can optionally include the file path if the file is not in the working directory).
output	The desired name of the output file.

Details

This function essentially makes a system call to in2csv from csvkit and instantly returns to the R prompt while the processing continues in the background. For small files, the conversion happens very quickly. For larger files, you can expect to wait a while.

The csv file might be considerably larger than the flat-file, particularly if the dictionary file defines overlapping columns, as some files do. You can verify the entire file was written by checking the number of lines in the file (perhaps using another system call to wc, for example system("wc -l path/to/flat-file"); system("wc -l path/to/csv")). The csv file should have one file more than the data file since it would include a line of headers.

Author(s)

Ananda Mahto

References

csvkit's in2csv documentation: https://csvkit.readthedocs.org/en/latest/scripts/in2csv.html

See Also

csvkit.schema

Examples

```
## Read an example dictionary file
data(sampleDctData)
## Write the dictionary to a dictionary file
## Write the data to a data file
currentdir <- getwd()
setwd(tempdir())
list.files(pattern=".dat|.dct|.csv")
writeLines(data66_dct, "data66.dct")
writeLines(data66_dat, "data66.dat")
## Is everything there in the dictionary file?
dct.parser("data66.dct", preview = TRUE)
# Missing the storage type, so remove that from includes
data66_dict <- dct.parser("data66.dct",
                          includes = c("StartPos", "ColName",
                                       "ColWidth", "VarLabel"))
```

```
list.files(pattern=".dat|.dct|.csv")
csvkit.schema(data66_dict)
list.files(pattern=".dat|.dct|.csv")
csvkit.fwf2csv(datafile = "data66.dat",
               schema   = "data66.dct.csv",
               output   = "data66-FINAL.csv")
Sys.sleep(10)
list.files(pattern=".dat|.dct|.csv")
read.csv("data66-FINAL.csv", nrows = 5)
setwd(currentdir)
```

`csvkit.schema` *Convert a parsed dictionary file to a csvkit schema file*

Description

After parsing a `.dct` dictionary file with the `dct.parser` function, it may be useful to convert that file to a schema that can be used by *csvkit*, a useufl Pyton tool for working with csv files. In particular, this creates a schema that allows you to convert a fixed width format file to a csv file.

Usage

```
csvkit.schema(x, columns.to.match = NULL)
```

Arguments

`x`	Your input `data.frame`. Must include at least the following information in separate columns: the variable names, the starting position of the variable, and the length of the variable in the fixed width file.
`columns.to.match`	By default, if the input file is the output of `dct.parser`, the values for this argument do not need to be specified. If you are using your own `data.frame`, specify which columns contain the (1) variable name, (2) the starting position, and (3) the width of the variable.

Details

This function will write a csv file to your current working directory. It takes the name of the original parsed dictionary file appended with .csv by default (which is stored as an attribute of the `data.frame` created during the dictionary parsing step). If that attribute is not present, it prompts the user for a file name, which should be provided *not quoted*.

Author(s)

Ananda Mahto

References

csvkit's in2csv documentation: https://csvkit.readthedocs.org/en/latest/scripts/in2csv.html

Examples

```
## Read an example dictionary file
data(sampleDctData)
## Write the data to a dictionary file
currentdir <- getwd()
setwd(tempdir())
writeLines(sipp84fp_dct, "sipp84fp.dct")
sipp84_R_dict <- dct.parser("sipp84fp.dct")
list.files(pattern=".dat|.dct|.csv")
csvkit.schema(sipp84_R_dict)
list.files(pattern=".dat|.dct|.csv")
setwd(currentdir)
```

---

data66_dat *Example dataset to be converted to a csv file*

Description

This is a sample dataset useful to demonstrate conversion to csv using csvkit.fwf2csv after parsing the dictionary file (using dct.parser) and creating a csvkit schema file (using csvkit.schema).

Source

Obtained from http://faculty-staff.ou.edu/L/Carlos.E.Lamarche-1/ec5243/data.html

---

data66_dct *Example dictionary file accompanying* data66_dat

Description

Example dictionary file accompanying data66_dat to be parsed using dct.parser and then possibly using other tools from the StataDCTutils package.

Source

Obtained from http://faculty-staff.ou.edu/L/Carlos.E.Lamarche-1/ec5243/data.html

`dct.parser` *Parse a Stata dictionary file for use in R*

Description

R cannot read Stata's dictionary files directly. This function parses the dictionary file to a `data.frame` that can be used to further process the data files and make them usable with R.

Usage

```
dct.parser(dct,
  includes = c("StartPos", "StorageType", "ColName", "ColWidth", "VarLabel"),
  preview = FALSE)
```

Arguments

`dct`	Stata dictionary file, most often with a `.dct` extension.
`includes`	A complete dictionary file includes (usually in this order), the column starting position, the storage type of the variable, the variable name, the width of the column, and the variable label. Delete any which are not relevant to your dictionary file.
`preview`	If you are not sure what values to select for `includes`, use the `preview = TRUE` argument to see the first few lines of the relevant portion of the dictionary file to decide what the dictionary file structure is.

Details

Many datasets are distributed as a combination of Stata `.dat` (data, usually fixed-width-format), `.dct` (dictionary), and `.do` (other commands for Stata, for example recoding the data and so on) files. The dictionary files are used to tell Stata details like which column in the data file represents the starting position of the data for a given variable, how many columns should be read for that given variable, what the storage type of that variable is, and what that variable's name and label shoud be.

The expected workflow might include (1) parsing the dictionary file using `dct.parser`, (2) converting the fixed width data file to a csv file using `csvkit` after generating a csvkit *schema* file using `csvkit.schema`, (3) reading in the file using your preferred method (for example, `fread`, `sqldf`, `read.csv`, or another appropriate method), (4) re-assigning some of the metadata extracted from the dictionary file to your newly imported dataset.

Author(s)

Ananda Mahto

References

- Stata data types: http://www.stata.com/help.cgi?datatypes
- Stata help for fixed-format data: http://www.stata.com/support/faqs/data-management/reading-fixed-format-data/
- Initial version of function on Stack Overflow: http://stackoverflow.com/questions/14224321/reading-dat-and-dct-directly-from-r

See Also

read.dta

Examples

```
## Read an example dictionary file
data(sampleDctData)
## Write the data to a dictionary file
currentdir <- getwd()
setwd(tempdir())
writeLines(sipp84fp_dct, "sipp84fp.dct")
dct.parser("sipp84fp.dct", preview = TRUE)
sipp84_R_dict <- dct.parser("sipp84fp.dct")
head(sipp84_R_dict)
setwd(currentdir)
```

MESSAGES *Print messages from parsed dictionary files*

Description

This is a simple convenience/utility function to print a nicely formatted message that might be stored in the output of a dictionary file parsed using dct.parser.

Usage

```
MESSAGES(x)
```

Arguments

x	The object that contains the message.

Author(s)

Ananda Mahto

Examples

```
## Read an example dictionary file
data(sampleDctData)
## Write the data to a dictionary file
currentdir <- getwd()
setwd(tempdir())
writeLines(sipp84fp_dct, "sipp84fp.dct")
sipp84_R_dict <- dct.parser("sipp84fp.dct")
MESSAGES(sipp84_R_dict)
#'setwd(currentdir)
```

`sipp84fp_dct` *Survey of Income and Program Participation (SIPP) - 1984 Panel*

Description

Dictionary file for the full 1984 panel for NBER's Survey of Income and Program Participation (SIPP).

Source

http://thedataweb.rm.census.gov/ftp/sipp_ftp.html#sipp84

Index