

# Package: readstata13 (via r-universe)

September 8, 2024

**Type** Package

**Title** Import 'Stata' Data Files

**Version** 0.10.2

**Description** Function to read and write the 'Stata' file format.

**URL** <https://github.com/sjewo/readstata13>

**BugReports** <https://github.com/sjewo/readstata13/issues>

**License** GPL-2 | file LICENSE

**Imports** Rcpp (>= 0.11.5)

**LinkingTo** Rcpp

**ByteCompile** yes

**Suggests** testthat

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Repository** <https://sjewo.r-universe.dev>

**RemoteUrl** <https://github.com/sjewo/readstata13>

**RemoteRef** HEAD

**RemoteSha** 12151bb8881de4014eac155542fd2cb193b94dd2

## Contents

as.caldays . . . . .	2
get.label . . . . .	3
get.label.name . . . . .	4
get.label.tables . . . . .	4
get.lang . . . . .	5
get.origin.codes . . . . .	6
maxchar . . . . .	7
read.dta13 . . . . .	7
readstata13 . . . . .	10
save.dta13 . . . . .	11

saveToExport . . . . .	13
set.label . . . . .	13
set.lang . . . . .	14
stbcal . . . . .	15
varlabel . . . . .	16
<b>Index</b>	<b>17</b>

---

as.caldays	<i>Convert Stata business calendar dates in readable dates.</i>
------------	---

---

## Description

Convert Stata business calendar dates in readable dates.

## Usage

```
as.caldays(buisdays, cal, format = "%Y-%m-%d")
```

## Arguments

buisdays	numeric Vector of business dates
cal	data.frame Conversion table for business calendar dates
format	character String with date format as in <a href="#">as.Date</a>

## Value

Returns a vector of readable dates.

## Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

## Examples

```
# read business calendar and data
sp500 <- stbcal(system.file("extdata/sp500.stbcal", package="readstata13"))
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"))

# convert dates and check
dat$ldatescal2 <- as.caldays(dat$ldate, sp500)
all(dat$ldatescal2==dat$ldatescal)
```

---

get.label	<i>Get Stata Label Table for a Label Set</i>
-----------	--

---

## Description

Retrieve the value labels for a specific Stata label set.

## Usage

```
get.label(dat, label.name)
```

## Arguments

dat                    *data.frame*. Data.frame created by read.dta13.

label.name            *character*. Name of the Stata label set

## Details

This function returns the table of factor levels which represent a Stata label set. The name of a label set for a variable can be obtained by [get.label.name](#).

## Value

Returns a named vector of code numbers

## Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

## Examples

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"))
labname <- get.label.name(dat,"type")
get.label(dat, labname)
```

get.label.name            *Get Names of Stata Label Set*

---

### Description

Retrieves the Stata label set in the dataset for all or an vector of variable names.

### Usage

```
get.label.name(dat, var.name = NULL, lang = NA)
```

### Arguments

dat                    *data.frame*. Data.frame created by read.dta13.  
var.name              *character vector*. Variable names. If NULL, get names of all label sets.  
lang                   *character*. Label language. Default language defined by `get.lang` is used if NA

### Details

Stata stores factor labels in variable independent labels sets. This function retrieves the name of the label set for a variable.

### Value

Returns an named vector of variable labels

### Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

---

get.label.tables            *Get all Stata Label Sets for a Data.frame*

---

### Description

Retrieve the value labels for all variables.

### Usage

```
get.label.tables(dat)
```

### Arguments

dat                    *data.frame*. Data.frame created by read.dta13.

**Details**

This function returns the factor levels which represent a Stata label set for all variables.

**Value**

Returns a named list of label tables

**Author(s)**

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

**Examples**

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"))
get.label.tables(dat)
```

---

*get.lang*                      *Show Default Label Language*

---

**Description**

Displays informations about the defined label languages.

**Usage**

```
get.lang(dat, print = T)
```

**Arguments**

**dat**                      *data.frame*. Data.frame created by `read.dta13`.  
**print**                    *logical*. If TRUE, print available languages and default language.

**Details**

Stata allows to define multiple label sets in different languages. This functions reports the available languages and the selected default language.

**Value**

Returns a list with two components:

**languages:** Vector of label languages used in the dataset

**default:** Name of the actual default label language, otherwise NA

**Author(s)**

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

---

get.origin.codes      *Get Origin Code Numbers for Factors*

---

## Description

Recreates the code numbers of a factor as stored in the Stata dataset.

## Usage

```
get.origin.codes(x, label.table)
```

## Arguments

`x`                    *factor*. Factor to obtain code for  
`label.table`        *table*. Table with factor levels obtained by [get.label](#).

## Details

While converting numeric variables into factors, the original code numbers are lost. This function reconstructs the codes from the attribute `label.table`.

## Value

Returns an integer with original codes

## Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

## Examples

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"))
labname <- get.label.name(dat,"type")
labtab <- get.label(dat, labname)

# comparsion
get.origin.codes(dat$type, labtab)
as.integer(dat$type)
```

---

`maxchar`*Check max char length of data.frame vectors*

---

**Description**

Stata requires us to provide the maximum size of a character vector as every row is stored in a bit region of this size.

**Usage**`maxchar(x)`**Arguments**

<code>x</code>	vector of data frame
----------------	----------------------

**Details**

Ex: If the max chars size is four, \_ is no character in this vector: 1. row: four 3. row: one\_ 4. row:

\_\_\_\_\_

If a character vector contains only missings or is empty, we will assign it a value of one, since Stata otherwise cannot handle what we write.

---

`read.dta13`*Read Stata Binary Files*

---

**Description**

`read.dta13` reads a Stata dta-file and imports the data into a `data.frame`.

**Usage**

```
read.dta13(  
  file,  
  convert.factors = TRUE,  
  generate.factors = FALSE,  
  encoding = "UTF-8",  
  fromEncoding = NULL,  
  convert.underscore = FALSE,  
  missing.type = FALSE,  
  convert.dates = TRUE,  
  replace.str1 = TRUE,  
  add.rownames = FALSE,  
  nonint.factors = FALSE,  
  select.rows = NULL,  
)
```

```

select.cols = NULL,
strlexport = FALSE,
strlpath = ".",
tz = "GMT"
)

```

## Arguments

`file` *character*. Path to the dta file you want to import.

`convert.factors` *logical*. If TRUE, factors from Stata value labels are created.

`generate.factors` *logical*. If TRUE and `convert.factors` is TRUE, missing factor labels are created from integers. If duplicated labels are found, unique labels will be generated according the following scheme: "label\_(integer code)".

`encoding` *character*. Strings can be converted from Windows-1252 or UTF-8 to system encoding. Options are "latin1" or "UTF-8" to specify target encoding explicitly. Stata 14, 15 and 16 files are UTF-8 encoded and may contain strings which can't be displayed in the current locale. Set `encoding=NULL` to stop reencoding.

`fromEncoding` *character*. We expect strings to be encoded as "CP1252" for Stata Versions 13 and older. For dta files saved with Stata 14 or newer "UTF-8" is used. In some situation the used encoding can differ for Stata 14 files and must be manually set.

`convert.underscore` *logical*. If TRUE, "\_" in variable names will be changed to ".".

`missing.type` *logical*. Stata knows 27 different missing types: ., .a, .b, ..., .z. If TRUE, attribute missing will be created.

`convert.dates` *logical*. If TRUE, Stata dates are converted.

`replace.strl` *logical*. If TRUE, replace the reference to a strL string in the data.frame with the actual value. The strl attribute will be removed from the data.frame (see details).

`add.rownames` *logical*. If TRUE, the first column will be used as rownames. Variable will be dropped afterwards.

`nonint.factors` *logical*. If TRUE, factors labels will be assigned to variables of type float and double.

`select.rows` *integer*. Vector of one or two numbers. If single value rows from 1:val are selected. If two values of a range are selected the rows in range will be selected.

`select.cols` *character*. Vector of variables to select.

`strlexport` *logical*. Should strl content be exported as binary files?

`strlpath` *character*. Path for strl export.

`tz` *character*. time zone specification to be used for POSIXct values. "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated).

## Details

If the filename is a url, the file will be downloaded as a temporary file and read afterwards.

Stata files are encoded in ansinew. Depending on your system's default encoding certain characters may appear wrong. Using a correct encoding may fix these.

Variable names stored in the dta-file will be used in the resulting data.frame. Stata types char, byte, and int will become integer; float and double will become numerics. R only knows a single missing type, while Stata knows 27, so all Stata missings will become NA in R. If you need to keep track of Stata's original missing types, you may use `missing.type=TRUE`.

Stata dates are converted to R's Date class the same way foreign handles dates.

Stata 13 introduced a new character type called strL. strLs are able to store strings up to 2 billion characters. While R is able to store strings of this size in a character vector, the printed representation of such vectors looks rather cluttered, so it's possible to save only a reference in the data.frame with option `replace.strl=FALSE`.

In R, you may use rownames to store characters (see for instance `data(swiss)`). In Stata, this is not possible and rownames have to be stored as a variable. If you want to use rownames, set `add.rownames` to TRUE. Then the first variable of the dta-file will hold the rownames of the resulting data.frame.

Reading dta-files of older and newer versions than 13 was introduced with version 0.8.

## Value

The function returns a data.frame with attributes. The attributes include

**datalabel:** Dataset label

**time.stamp:** Timestamp of file creation

**formats:** Stata display formats. May be used with `sprintf`

**types:** Stata data type (see Stata Corp 2014)

**val.labels:** For each variable the name of the associated value labels in "label"

**var.labels:** Variable labels

**version:** dta file format version

**label.table:** List of value labels.

**strl:** Character vector with long strings for the new strl string variable type. The name of every element is the identifier.

**expansion.fields:** list providing variable name, characteristic name and the contents of Stata characteristic field.

**missing:** List of numeric vectors with Stata missing type for each variable.

**byteorder:** Byteorder of the dta-file. LSF or MSF.

**orig.dim:** Dimension recorded inside the dta-file.

## Note

read.dta13 uses GPL 2 licensed code by Thomas Lumley and R-core members from `foreign::read.dta()`.

**Author(s)**

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

**References**

Stata Corp (2014): Description of .dta file format <https://www.stata.com/help.cgi?dta>

**See Also**

[read.dta](#) in package `foreign` and `memisc` for dta files from Stata versions < 13 and `read_dta` in package `haven` for Stata version >= 13.

**Examples**

```
## Not run:  
library(readstata13)  
r13 <- read.dta13("https://www.stata-press.com/data/r13/auto.dta")  
  
## End(Not run)
```

---

readstata13

*Import Stata Data Files*

---

**Description**

Function to read the Stata file format into a `data.frame`.

**Note**

If you catch a bug, please do not sue us, we do not have any money.

**Author(s)**

Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

**See Also**

[read.dta](#) and `memisc` for dta files from Stata Versions < 13

save.dta13

*Write Stata Binary Files***Description**

save.dta13 writes a Stata dta-file bitwise and saves the data into a dta-file.

**Usage**

```
save.dta13(
  data,
  file,
  data.label = NULL,
  time.stamp = TRUE,
  convert.factors = TRUE,
  convert.dates = TRUE,
  tz = "GMT",
  add.rownames = FALSE,
  compress = FALSE,
  version = 117,
  convert.underscore = FALSE
)
```

**Arguments**

data	<i>data.frame</i> . A data.frame Object.
file	<i>character</i> . Path to the dta file you want to export.
data.label	<i>character</i> . Name of the dta-file.
time.stamp	<i>logical</i> . If TRUE, add a time.stamp to the dta-file.
convert.factors	<i>logical</i> . If TRUE, factors will be converted to Stata variables with labels. Stata expects strings to be encoded as Windows-1252, so all levels will be recoded. Character which can not be mapped in Windows-1252 will be saved as hexcode.
convert.dates	<i>logical</i> . If TRUE, dates will be converted to Stata date time format. Code from <code>foreign::write.dta</code>
tz	<i>character</i> . time zone specification to be used for POSIXct values and dates (if <code>convert.dates</code> is TRUE). "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated).
add.rownames	<i>logical</i> . If TRUE, a new variable rownames will be added to the dta-file.
compress	<i>logical</i> . If TRUE, the resulting dta-file will use all of Statas numeric-vartypes.
version	<i>numeric</i> . Stata format for the resulting dta-file either Stata version number (6 - 16) or the internal Stata dta-format (e.g. 117 for Stata 13). Experimental support for large datasets: Use <code>version="15mp"</code> to save the dataset in the new Stata 15/16 MP file format. This feature is not thoroughly tested yet.

`convert.underscore`

*logical*. If TRUE, all non numerics or non alphabet characters will be converted to underscores.

## Value

The function writes a dta-file to disk. The following features of the dta file format are supported:

**datalabel:** Dataset label

**time.stamp:** Timestamp of file creation

**formats:** Stata display formats. May be used with `sprintf`

**type:** Stata data type (see Stata Corp 2014)

**var.labels:** Variable labels

**version:** dta file format version

**strl:** List of character vectors for the new strL string variable type. The first element is the identifier and the second element the string.

## Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

## References

Stata Corp (2014): Description of .dta file format <https://www.stata.com/help.cgi?dta>

## See Also

`read.dta` in package `foreign` and `memisc` for dta files from Stata versions < 13 and `read_dta` in package `haven` for Stata version >= 13.

## Examples

```
## Not run:  
  library(readstata13)  
  save.dta13(cars, file="cars.dta")  
  
## End(Not run)
```

---

saveToExport	<i>Check if numeric vector can be expressed as integer vector</i>
--------------	---

---

**Description**

Compression can reduce numeric vectors as integers if the vector does only contain integer type data.

**Usage**

```
saveToExport(x)
```

**Arguments**

x	vector of data frame
---	----------------------

---

set.label	<i>Assign Stata Labels to a Variable</i>
-----------	--

---

**Description**

Assign value labels from a Stata label set to a variable. If duplicated labels are found, unique labels will be generated according the following scheme: "label\_(integer code)". Levels without labels will become <NA>.

**Usage**

```
set.label(dat, var.name, lang = NA)
```

**Arguments**

dat	<i>data.frame</i> . Data.frame created by read.dta13.
var.name	<i>character</i> : Name of the variable in the data.frame
lang	<i>character</i> : Label language. Default language defined by <a href="#">get.lang</a> is used if NA

**Value**

Returns a labeled factor

### Examples

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"),
                  convert.factors=FALSE)

# compare vectors
set.label(dat, "type")
dat$type

# German label
set.label(dat, "type", "de")
```

---

set.lang

*Assign Stata Language Labels*

---

### Description

Changes default label language for a dataset. Variables with generated labels (option `generate.labels=TRUE`) are kept unchanged.

### Usage

```
set.lang(dat, lang = NA, generate.factors = FALSE)
```

### Arguments

<code>dat</code>	<i>data.frame</i> . Data.frame created by <code>read.dta13</code> .
<code>lang</code>	<i>character</i> . Label language. Default language defined by <code>get.lang</code> is used if NA
<code>generate.factors</code>	<i>logical</i> . If TRUE, missing factor levels are generated.

### Value

Returns a `data.frame` with value labels in language "lang".

### Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

### Examples

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"))
get.lang(dat)
varlabel(dat)

# set German label
datDE <- set.lang(dat, "de")
get.lang(datDE)
varlabel(datDE)
```

---

`stbcal`*Parse Stata business calendar files*

---

### Description

Create conversion table for business calendar dates.

### Usage

```
stbcal(stbcalfile)
```

### Arguments

`stbcalfile`     *stbcal-file* Stata business calendar file created by Stata.

### Details

Stata 12 introduced business calendar format. Business dates are integer numbers in a certain range of days, weeks, months or years. In this range some days are omitted (e.g. weekends or holidays). If a business calendar was created, a `stbcal` file matching this calendar was created. This file is required to read the business calendar. This parser reads the `stbcal-` file and returns a `data.frame` with dates matching business calendar dates.

A `dta`-file containing Stata business dates imported with `read.stata13()` shows in formats which `stdcal` file is required (e.g. " `sp500.stbcal` ).

Stata allows adding a short description called `purpose`. This is added as an attribute of the resulting `data.frame`.

### Value

Returns a `data.frame` with two cols:

**range:** The date matching the `businessdate`. Date format.

**buisdays:** The Stata business calendar day. Integer format.

### Author(s)

Jan Marvin Garbuszus <[jan.garbuszus@ruhr-uni-bochum.de](mailto:jan.garbuszus@ruhr-uni-bochum.de)>

Sebastian Jeworutzki <[sebastian.jeworutzki@ruhr-uni-bochum.de](mailto:sebastian.jeworutzki@ruhr-uni-bochum.de)>

### Examples

```
sp500 <- stbcal(system.file("extdata/sp500.stbcal", package="readstata13"))
```

---

varlabel *Get and assign Stata Variable Labels*

---

### Description

Retrieve or set variable labels for a dataset.

### Usage

```
varlabel(dat, var.name = NULL, lang = NA)
```

```
varlabel(dat) <- value
```

### Arguments

<code>dat</code>	<i>data.frame</i> . Data.frame created by <code>read.dta13</code> .
<code>var.name</code>	<i>character vector</i> . Variable names. If <code>NULL</code> , get label for all variables.
<code>lang</code>	<i>character</i> . Label language. Default language defined by <code>get.lang</code> is used if <code>NA</code> .
<code>value</code>	<i>character vector</i> . Character vector of size <code>ncol(data)</code> with variable names.

### Value

Returns an named vector of variable labels

### Author(s)

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>  
Sebastian Jeworutzki <sebastian.jeworutzki@ruhr-uni-bochum.de>

### Examples

```
dat <- read.dta13(system.file("extdata/statacar.dta", package="readstata13"),
  convert.factors=FALSE)

# display variable labels
varlabel(dat)

# display german variable labels
varlabel(dat, lang="de")

# display german variable label for brand
varlabel(dat, var.name = "brand", lang="de")

# define new variable labels
varlabel(dat) <- letters[1:ncol(dat)]

# display new variable labels
varlabel(dat)
```

# Index

'varlabel<-' (varlabel), 16

as.caldays, 2  
as.Date, 2

get.label, 3, 6  
get.label.name, 3, 4  
get.label.tables, 4  
get.lang, 4, 5, 13, 14, 16  
get.origin.codes, 6

maxchar, 7

read.dta, 10, 12  
read.dta13, 7  
readstata13, 10  
readstata13-package (readstata13), 10

save.dta13, 11  
saveToExport, 13  
set.label, 13  
set.lang, 14  
sprintf, 9, 12  
stbcal, 15

varlabel, 16  
varlabel<- (varlabel), 16