

# Package 'SASxport'

February 14, 2012

**Type** Package

**Title** Read and Write SAS XPORT Files

**Version** 1.2.4

**Date** 2010-11-11

**Description** This package provides functions for reading, listing the contents of, and writing SAS xport format files. The functions support reading and writing of either individual data frames or sets of data frames. Further, a mechanism has been provided for customizing how variables of different data types are stored.

**Author** Unless otherwise noted, the contents of this package were written by Gregory R. Warnes <greg@warnes.net> and are provided under the terms of the GNU General Public License, version 2.0 or later. -- The files 'ieee2ibm.c' and 'ibm2ieee.c' were extracted from BRL-CAD, file /brlcad/src/libbu/htond.c written by Michael John Muuss, Copyright (c) 2004-2007 United States Government as represented by the U.S. Army Research Laboratory, and is utilized and redistributed under the terms of the GNU Lesser General Public License, version 2.1. -- The files 'AFirst.lib.s', 'all.is.numeric.R', 'importConvertDateTime.R', 'in.operator.R', 'makeNames.R', 'read.xport.R', and 'testDateTime.R' are copied or adapted from the 'Hmisc' package created by Frank E. Harrell, Jr. <f.harrell@vanderbilt.edu>, and are utilized and redistributed under the terms of the GNU General Public License, version 2.0 or later. -- The creation of this package was partially funded by Metrum Institute <<http://metruminstitute.org>>.

**Maintainer** Gregory R. Warnes <greg@warnes.net>

**License** GPL-2

**Depends** R (>= 2.4.0)

**Imports** chron

**URL** <http://www.warnes.net>, <http://metruminstitute.org>

**Repository** CRAN

**Date/Publication** 2010-11-12 07:31:55

## R topics documented:

SASxport-package . . . . .	2
Alfa . . . . .	3
label . . . . .	4
lookup.xport . . . . .	6
read.xport . . . . .	8
toSAS.default . . . . .	11
write.xport . . . . .	13

<b>Index</b>	<b>16</b>
--------------	-----------

---

SASxport-package	<i>Read and Write SAS Export Files</i>
------------------	--

---

### Description

This package provides functions to read, list contents of, and write SAS export files.

### Details

The `read.xport` function reads SAS xport formatted files, augmenting the functionality of the `read.xport` function provided in the 'foreign' package with additional features borrowed from `sasxport.get` in Frank Harrell's 'Hmisc' package. Namely, variables are properly coerced into the types specified by the format field. All standard numeric and string SAS formats are supported automatically, while user-defined formats are supported when the user has included the appropriate format data in the xport file via:

```
PROC FORMAT CNTLOUT=format;
```

The `write.xport` function writes one or more data sets into a SAS xport formatted file. Standard R data types, including date and time objects (e.g. `Date`, and `POSIX.t`) are stored with proper SAS format types. Handling of object formatting is customizable by providing methods for the function `toSAS`. This is accomplished by writing a new method for `toSAS` for the object class of interest. The `toSAS` method is responsible for converting its argument to either a simple floating point or character variable (the only basic types permitted by the xport format) and adding the appropriate SAS format code in the "SASformat" attribute.

The `write.xport` function also allows the user to override the operating system type and SAS version information, as well as object creation and modification times.

### Index

<code>lookup.xport</code>	Lookup Information on a SAS XPORT Format Library
<code>read.xport</code>	Import SAS XPORT files
<code>toSAS.default</code>	Convert R data object for storage in SAS xport file
<code>units</code>	Set or Retrieve the label, SASformat, SASifformat, or units Attribute of a Vector
<code>write.xport</code>	Write data to a SAS XPORT file

**Funding**

This creation of this package was partially funded by Metrum Institute <http://metruminstitute.org>.

**Maintainer**

Gregory R. Warnes <[greg@warnes.net](mailto:greg@warnes.net)>

**Support**

Technical support contracts and other services for R, this package, and other packages are available from Random Technologies LLC <http://random-technologies-llc.com>.

**Author(s)**

Unless otherwise noted, the contents of this package were written by Gregory R. Warnes <[greg@warnes.net](mailto:greg@warnes.net)>, are Copyright (c) 2007 by Random Technologies LLC <http://random-technologies-llc.com>, and are provided under the terms of the GNU General Public License, version 2.0 or later.

The file 'htond.c' is extracted from BRL-CAD <http://www.brlcad.org/>, written by Michael John Muuss, and is Copyright (c) 2004-2007 United States Government as represented by the U.S. Army Research Laboratory, and is utilized and redistributed under the terms of the GNU Lesser General Public License, version 2.1.

The files 'AFirst.lib.s', 'all.is.numeric.R', 'importConvertDateTime.R', 'in.operator.R', 'makeNames.R', 'read.xport.R', and 'testDateTime.R' are copied or adapted from the 'Hmisc' package created by Frank E. Harrell, Jr. <[f.harrell@vanderbilt.edu](mailto:f.harrell@vanderbilt.edu)>, and are utilized and redistributed under the terms of the GNU General Public License, version 2.0 or later.

---

Alfalfa

*Example SAS data set*

---

**Description**

This data set exists to provide an example file for `lookup.xport()` and `read.xport()`

**Usage**

```
data(Alfalfa)
```

**Format**

A data frame with 40 observations on the following 6 variables.

POP Population, a factor with levels MAX and min

SAMPLE Sample ID (0:5)

REP Replicate (always 1)

SEEDWT Seed weight  
 HARV1 Harvest 1 volume  
 HARV2 Harvest 2 volume

### Details

Population "MAX" has slightly higher harvest volumes (HARV1 and HARV2) than population "min".  
 (Surprise! Shock! Awe!)

### Source

The 'Alfalfa.xpt' file was obtained from the R 'foreign' package.

### Examples

```
data(Alfalfa)

# go were the data is...
here <- getwd()
setwd(file.path(.path.package("SASxport"), "data"))

# Description of the file contents
lookup.xport("Alfalfa.xpt")

# Load the file contents
Alfalfa <- read.xport("Alfalfa.xpt")
head(Alfalfa)

# return home
setwd(here)

# Just for fun, plot the data
par(mfrow=c(1,2))
plot( HARV1 ~ POP, data=Alfalfa)
plot( HARV2 ~ POP, data=Alfalfa)
```

---

label	<i>Set or Retrieve the 'label', 'SASformat', or 'SASiformat' Attribute of a Vector</i>
-------	--

---

### Description

Sets or retrieves the "label", "SASformat", or "SASiformat" attribute of an object.

More comprehensive support for object labels, and SASformat, are available in Frank Harrell's Hmisc package.

**Usage**

```
label(x, default)
label(x) <- value

SASformat(x, default)
SASformat(x) <- value

SASiformat(x, default)
SASiformat(x) <- value
```

**Arguments**

x	any object
value	new value for the "label", "SASformat", or "SASiformat" attribute of an object.
default	value to return when no appropriate attribute is found. The usual return value is NULL.

**Value**

the contents of the "label", "SASformat", or "SASiformat" attribute of x, if any; otherwise, the value provided by default.

**Author(s)**

Gregory R. Warnes <greg@random-techologies-llc.com> based on code from the Hmisc library by Frank E. Harrell, Jr.

**Examples**

```
fail.time <- c(10,20)

# set attributes
label(fail.time) <- 'Failure Time'
SASformat(fail.time) <- 'Numeric2'
SASiformat(fail.time) <- 'Numeric2'

# display individual attributes
label(fail.time)
SASformat(fail.time)
SASiformat(fail.time)

# display all attributes
attributes(fail.time)

# Example showing specification of default return value
a <- 70
label(a, default="no label")
```

```
## Not run:
# for a nice display
library(Hmisc)
describe(fail.time)

f <- cph(Surv(fail.time, event) ~ xx)
plot(xx,xx2,xlab=label(xx),"s",sep="")

## End(Not run)
```

---

lookup.xport

*Describe the Contents of an SAS XPORT File*


---

## Description

Describe the contents of an SAS XPORT file.

## Usage

```
lookup.xport(file)
## S3 method for class 'lookup.xport'
print(x, ...)
## S3 method for class 'lookup.xport'
summary(object, ...)
## S3 method for class 'summary.lookup.xport'
print(x, ...)
```

## Arguments

file	Character string specifying the name or URL of a SAS XPORT file.
x, object	Object to be printed or summarized
...	Optional arguments

## Details

The `lookup.xport` function is a simple wrapper for the `lookup.xport` function provided by the `foreign` library. The wrapper adds the ability to handle URL's, and returns an object of class `lookup.xport` for which appropriate `print`, and `summary` functions are provided.

## Value

`lookup.xport` returns a list with one component for each dataset in the XPORT format library.

`summary.lookup.xport` returns a single data frame containing:

dataset	Dataset name,
---------	---------------

name	Variable name,
type	Type of variable (one of 'character' or 'numeric'),
format	SAS format,
width	SAS format width,
label	Variable label,
nobs	Number of observations.

### See Also

For complete documentation of `lookup.xport` see the manual page for [lookup.xport](#).

### Examples

```
## Get information on a local file
lookup.xport("Alfalfa.xpt")

## Or read a copy of test2.xpt available on the web:
## Not run:
host <- 'http://biostat.mc.vanderbilt.edu'
path <- '/cgi-bin/viewvc.cgi/*checkout*/Hmisc/trunk/tests/test2.xpt'
url <- paste(host,path,sep="")

w <- lookup.xport(url)

# display the information (calls 'print.lookup.xport')
w

# names of data sets
names(w)

# names of variables within data sets
w$Z$name

# use summary
wS <- summary(w)
wS # same display

# variable names within all data sets
wS$name

## End(Not run)
```

read.xport

*Import a SAS XPORT File***Description**

Read a SAS XPORT format file and return the contained dataset(s).

**Usage**

```
read.xport(file,
           force.integer=TRUE,
           formats=NULL,
           name.chars=NULL,
           names.lower=FALSE,
           keep=NULL,
           drop=NULL,
           as.is=0.95,
           verbose=FALSE,
           as.list=FALSE,
           include.formats=FALSE
          )
```

**Arguments**

<code>file</code>	Character string specifying the name or URL of a SAS XPORT file.
<code>force.integer</code>	Logical flag indicating whether integer-valued variables should be returned as integers (TRUE) or doubles (FALSE). Variables outside the supported integer range ( <code>.Machine\$integer.max</code> ) will always be converted to doubles.
<code>formats</code>	a data frame or list (like that created by <code>foreign::read.xport</code> ) containing PROC FORMAT output, if such output is not stored in the main transport file.
<code>name.chars</code>	Vector of additional characters permissible in variable names. By default, only the alpha and numeric characters ( <code>[A-Za-z0-9]</code> ) and periods ( <code>'.'</code> ) are permitted. All other characters are converted into periods ( <code>'.'</code> ).
<code>names.lower</code>	Logical indicating whether variable and dataset names should be converted to lowercase (TRUE) or left uppercase (FALSE)
<code>keep</code>	a vector of names of SAS datasets to process. This list must include PROC FORMAT dataset if it is present for datasets to use any of its value label formats.
<code>drop</code>	a vector of names of SAS datasets to ignore (original SAS upper case names)
<code>as.is</code>	Either a logical flag indicating whether SAS character variables should be preserved as character objects (TRUE) or factor objects (FALSE), or a fractional cutoff between 0 and 1.  When a fractional cutoff is provided, character variables containing a more than this fraction of unique values will be stored as a character variables. This is done in order to preserve space, since factors must store both the integer factor codes and the character factor labels.

<code>verbose</code>	Logical indicating whether progress should be printed during the data loading and conversion process.
<code>as.list</code>	Logical indicating whether to return a list even if the SAS xport file contains only one dataset.
<code>include.formats</code>	Logical indicating whether to include SAS format information (if present) in the returned list

### Details

- SAS date, time, and date/time variables are converted respectively to Date, POSIX, or chron objects
- SAS labels are stored in "label" attributes on each variable
- SAS formats are stored in "format" attributes on each variable
- SAS integer variables are stored as integers unless `force.integer` is FALSE

If the file includes the output of PROC FORMAT CNTLOUT=, variables having customized label formats will be converted to factor objects with appropriate labels.

### Value

If only a single dataset is present (after removing PROC FORMAT data when `include.formats=FALSE`), the return value is a single dataframe object. Otherwise the return is a list of dataframe objects.

Note that if `include.formats=TRUE`, the returned list will contain a dataframe named "FORMATS" containing any available 'PROC FORMAT' information.

### Note

This code provides a subset of the functionality of the `sasxport.get` function in the Hmisc library.

### Author(s)

Gregory R. Warnes <greg@warnes.net> based on `Hmisc:::sasxport.get` by Frank E. Harrell, Jr.

### See Also

[read.xport](#), [label](#), [sas.get](#), [sasxport.get](#), [Dates](#), [DateTimeClasses](#), [chron](#), [lookup.xport](#), [contents](#), [describe](#)

### Examples

```
# -----
# SAS code to generate test dataset:
# -----
# libname y SASV5XPT "test2.xpt";
#
# PROC FORMAT; VALUE race 1=green 2=blue 3=purple; RUN;
```

```

# PROC FORMAT CNTLOUT=format;RUN; * Name, e.g. 'format', unimportant;
# data test;
# LENGTH race 3 age 4;
# age=30; label age="Age at Beginning of Study";
# race=2;
# d1='3mar2002'd ;
# dt1='3mar2002 9:31:02'dt;
# t1='11:13:45't;
# output;
#
# age=31;
# race=4;
# d1='3jun2002'd ;
# dt1='3jun2002 9:42:07'dt;
# t1='11:14:13't;
# output;
# format d1 mmdyy10. dt1 datetime. t1 time. race race.;
# run;
# data z; LENGTH x3 3 x4 4 x5 5 x6 6 x7 7 x8 8;
#   DO i=1 TO 100;
#     x3=ranuni(3);
#     x4=ranuni(5);
#     x5=ranuni(7);
#     x6=ranuni(9);
#     x7=ranuni(11);
#     x8=ranuni(13);
#     output;
#     END;
#   DROP i;
#   RUN;
# PROC MEANS; RUN;
# PROC COPY IN=work OUT=y;SELECT test format z;RUN; *Creates test2.xpt;
# -----

# Read this dataset from a local file:
## Not run:
w <- read.xport('test2.xpt')

## End(Not run)

# Or read a copy of test2.xpt available on the web:
host <- 'http://biostat.mc.vanderbilt.edu'
path <- '/cgi-bin/viewvc.cgi/*checkout*/Hmisc/trunk/tests/test2.xpt'
url <- paste(host,path,sep="")

w <- read.xport(url)

# We can also get the dataset wrapped in a list
w <- read.xport(url, as.list=TRUE)

# And we can ask for the format information to be included as well.
w <- read.xport(url, as.list=TRUE, include.formats=TRUE)

```

```
## Not run:
## The Hmisc library provides many useful functions for interacting with
## data imported from SAS via read.xport()
library(Hmisc)

describe(w$test) # see labels, format names for dataset test
lapply(w, describe)# see descriptive stats in more detail for each variable

contents(w$test) # another way to see variable attributes
lapply(w, contents)# show contents of individual items in more detail

options(digits=7) # compare the following matrix with PROC MEANS output
t(sapply(w$z, function(x)
  c(Mean=mean(x),SD=sqrt(var(x)),Min=min(x),Max=max(x))))))

## End(Not run)
```

---

toSAS.default

---

*Convert R Data Object for Storage in a SAS XPORT File*


---

## Description

The toSAS methods control how R objects and data types are represented when stored into a SAS xport format file using write.xport.

## Usage

```
toSAS(x, format, format.info=NULL)
## Default S3 method:
toSAS(x, format=SASformat(x), format.info=NULL)
## S3 method for class 'numeric'
toSAS(x, format=SASformat(x), format.info=NULL)
## S3 method for class 'logical'
toSAS(x, format=SASformat(x), format.info=NULL)
## S3 method for class 'character'
toSAS(x, format=SASformat(x), format.info=NULL)
## S3 method for class 'factor'
toSAS(x, format=SASformat(x), format.info=NULL)
## S3 method for class 'POSIXt'
toSAS(x, format="DATETIME16.", format.info=NULL)
## S3 method for class 'Date'
toSAS(x, format="DATE9.", format.info=NULL)
```

## Arguments

x	Object to be converted
format	SAS format name
format.info	Table of SAS format information

## Details

To add support for a new object type, create an appropriate `toSAS` method. This method must convert the object data to either an object of type "numeric" (double-precision floating point) or type "character", the only basic types permitted by the `xport` format, and should add an attribute named "SASformat" to the object providing an appropriate SAS format string or "" (indicating the default SAS format).

## Value

A vector of type "character" or of type "numeric", with an attribute named "label" containing the SAS format specification.

## Author(s)

Gregory R. Warnes <greg@warnes.net>

## See Also

[write.xport](#), [read.xport](#), [lookup.xport](#)

## Examples

```
####
## See how an R date/time object will be stored in a SAS xport file:
####

# Date and time
dateTimeObj <- ISOdate(2007,08,01,10,14,37)
class(dateTimeObj)
dateTimeObj

sasDateTimeObj <- toSAS(dateTimeObj)
sasDateTimeObj

# Now just the date portion
dateObj <- as.Date(dateTimeObj)
dateObj

sasDateObj <- toSAS(dateObj)
sasDateObj

####
## Create a new R object class based on factor to hold color names
```

```
#####
colorFactor <- function(x) # constructor
{
  retval <- factor(x, levels=c("Red","Green","Blue") )
  class(retval) <- c("colorFactor","factor")
  retval
}

## create one and look at it
cf <- colorFactor( c("Red","Red","Blue",NA) )
cf

## See how it will be represented in a SAS xport file
toSAS(cf)

## Create a new conversion function to store as a RGB hex value
toSAS.colorFactor <- function(x, format="")
{
  retval <- ifelse(x=="Red", "#FF0000",
                  ifelse(x=="Green", "#00FF00", "#0000FF") )
  attr(retval, "SASformat") <- format
  retval
}

## see it in action
toSAS(cf)
```

---

write.xport

---

*Write Data to a SAS XPORT File*


---

## Description

This function writes one or more data frames into a SAS XPORT format library file.

## Usage

```
write.xport(...,
            list=base::list(),
            file = stop("'file' must be specified"),
            verbose=FALSE,
            sasVer="7.00",
            osType,
            cDate=Sys.time(),
            formats=NULL,
            autogen.formats=TRUE
          )
```

**Arguments**

<code>...</code>	One or more data frames to be stored
<code>list</code>	A list containing data frames to be stored.
<code>file</code>	File name or connection object. Use "" to view the raw data
<code>verbose</code>	Logical flag controlling whether status is reported during processing
<code>sasVer</code>	SAS version string
<code>osType</code>	Opererating system, defaults to "R X.Y.Z" for appropriate values of X, Y, and Z
<code>cDate</code>	Date object specifying dataset creation date
<code>formats</code>	Optional data frame containing SAS format information.
<code>autogen.formats</code>	Logical indiciating whether SAS formats should be auto-generated for factor variables.

**Details**

The function creates a SAS XPORT data file (see reference) from one or more data frames. This file format imposes a number of constraints:

- Data set and variable names are truncated to 8 characters and converted to upper case. All characters outside of the set A-Z, 0-9, and '\\_' are converted to '\\_'.
- Character variables are stored as characters.
- If `autogen.formats=TRUE` (the default), factor variables are stored as numeric with an appropriate SAS format specification. If `autogen.formats=FALSE`, factor variables are stored as characters.
- All numeric variables are stored as double-precision floating point values utilizing the IBM mainframe double precision floating point format (see the reference).
- Date and time variables are either converted to number of days since 1960-01-01 (date only), or number of seconds since 1960-01-01:00:00:00 GMT (date-time variables).
- Missing values are converted to the standard SAS missing value '.'

In addition, the SAS XPORT format allows each variable to have a corresponding label, display format, and input format. To set these values, add the attribute 'label', 'SASformat', or 'SASifformat' to individual data frame variables. (See the example section.)

The actual translation of R objects to objects appropriate for SAS is handled by the [toSAS](#) generic and associated methods.

**Value**

No return value

**Note**

This package was created by Random Technologies LLC <http://random-technologies-llc.com> with partial funding by Metrum Institute <http://metruminstitute.org>.

Technical support contracts for this and other R packages are available from Random Technologies LLC <http://random-technologies-llc.com>.

**Author(s)**

Gregory R. Warnes <greg@warnes.net>

**References**

SAS Technical Support document TS-140: “The Record Layout of a Data Set in SAS Transport (XPORT) Format” available at <http://ftp.sas.com/techsup/download/technote/ts140.html>.

**See Also**

[toSAS](#), [lookup.xport](#), [read.xport](#)

**Examples**

```
#####
## R version of the example given in TS-140
#####

## manually create a data set
temp <- data.frame( x=c(1, 2, NA, NA ), y=c('a', 'B', NA, '*' ) )

## look at it
temp

## add a format specifier (not used by R)
attr(temp$x, 'SASformat') <- 'date7.'

## add a variable label (not used by R)
attr(temp$y, 'label') <- 'character variable'

## verify the additions
str(temp)

## rename the data set
abc <- temp

# create a SAS XPORT file
write.xport( abc, file="xxx.dat" )

# list the contents of the file
lookup.xport("xxx.dat")

## reload the data
xxx.abc <- read.xport("xxx.dat")

## and look at it
xxx.abc

## Note that the variable names have been converted to uppercase
```

# Index

- \*Topic **IO**
    - toSAS.default, 11
  - \*Topic **datasets**
    - Alfalfa, 3
  - \*Topic **file**
    - lookup.xport, 6
    - write.xport, 13
  - \*Topic **interface**
    - label, 4
    - read.xport, 8
  - \*Topic **manip**
    - lookup.xport, 6
    - read.xport, 8
    - toSAS.default, 11
  - \*Topic **package**
    - SASxport-package, 2
  - \*Topic **utilities**
    - label, 4
- Alfalfa, 3
- chron, 9
- contents, 9
- Dates, 9
- DateTimeClasses, 9
- describe, 9
- label, 4, 9
- label<- (label), 4
- lookup.xport, 6, 6, 7, 9, 12, 15
- print.lookup.xport (lookup.xport), 6
- print.summary.lookup.xport  
(lookup.xport), 6
- read.xport, 2, 8, 9, 12, 15
- sas.get, 9
- SASformat (label), 4
- SASformat<- (label), 4
- SASiformat (label), 4
- SASiformat<- (label), 4
- SASxport (SASxport-package), 2
- SASxport-package, 2
- sasxport.get, 2, 9
- summary.lookup.xport (lookup.xport), 6
- toSAS, 14, 15
- toSAS (toSAS.default), 11
- toSAS.default, 11
- write.xport, 12, 13