|  |  |
| --- | --- |
| **Problem** | **Solutions** |
| 1. How to Import SAS XPORT Files into R With The foreign package | read.xport: Import a SAS XPORT File library(foreign)  > read.xport("test.xpt")  Error in file(con, "rb") : cannot open the connection  In addition: Warning message:  In file(con, "rb") : cannot open file 'test.xpt': No such file or directory  >  > read.xport(file,  + force.integer=TRUE,  + formats=NULL,  + name.chars=NULL,  + names.tolower=FALSE,  + keep=NULL,  + drop=NULL,  + as.is=0.95,  + verbose=FALSE,  + as.list=FALSE,  + include.formats=FALSE  + )  Error in as.character(x) :  cannot coerce type 'closure' to vector of type 'character'  > lookup.xport("test.xpt")  Error in lookup.xport.inner(file) :  unable to open file: 'No such file or directory'Read a SAS XPORT format file and return the contained dataset(s). Usage  |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12 | [read.xport](https://rdrr.io/cran/SASxport/man/read.xport.html)([file](https://rdrr.io/r/base/connections.html),  force.integer=[**TRUE**](https://rdrr.io/r/base/logical.html),  formats=[**NULL**](https://rdrr.io/r/base/NULL.html),  name.chars=[**NULL**](https://rdrr.io/r/base/NULL.html),  names.tolower=[**FALSE**](https://rdrr.io/r/base/logical.html),  keep=[**NULL**](https://rdrr.io/r/base/NULL.html),  [drop](https://rdrr.io/r/base/drop.html)=[**NULL**](https://rdrr.io/r/base/NULL.html),  as.is=0.95,  verbose=[**FALSE**](https://rdrr.io/r/base/logical.html),  [as.list](https://rdrr.io/r/base/list.html)=[**FALSE**](https://rdrr.io/r/base/logical.html),  include.formats=[**FALSE**](https://rdrr.io/r/base/logical.html)  ) |  Arguments  |  |  |  | | --- | --- | --- | | **file** | Character string specifying the name or URL of a SAS XPORT file. | | | **force.integer** | Logical flag indicating whether integer-valued variables should be returned as integers (TRUE) or doubles (FALSE). Variables outside the supported integer range (.Machine$integer.max) will always be converted to doubles. | | | **formats** | a data frame or list (like that created by foreign:::read.xport) containing PROC FORMAT output, if such output is not stored in the main transport file. | | | **name.chars** | Vector of additional characters permissible in variable names. By default, only the alpha and numeric characters ([A-Za-z0-9]) and periods ('.') are permitted. All other characters are converted into periods ('.'). | | | **names.tolower** | Logical indicating whether variable and dataset names should be converted to lowercase (TRUE) or left uppercase (FALSE) | | | **keep** | a vector of names of SAS datasets to process. This list must include PROC FORMAT dataset if it is present for datasets to use use any of its value label formats. | | | **drop** | a vector of names of SAS datasets to ignore (original SAS upper case names) | | | **as.is** | 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. | | | **verbose** | Logical indicating whether progress should be printed during the data loading and conversion process. | | | **as.list** | Logical indicating whether to return a list even if the SAS xport file contains only only one dataset. | | | **include.formats** | Logical indicating whether to include SAS format information (if present) in the returned list | | | read.xport {foreign} | | R Documentation |  Read a SAS XPORT Format LibraryDescription Reads a file as a SAS XPORT format library and returns a list of data.frames. Usage read.xport(file) Arguments |
| 2. How To Import SAS Files into R With The haven Package? | read\_sas: Read and write SAS files Description read\_sas() supports both sas7bdat files and the accompanying sas7bcat files that SAS uses to record value labels. write\_sas() is currently experimental and only works for limited datasets. Usage  |  |  | | --- | --- | | 1  2  3  4 | [read\_sas](https://rdrr.io/cran/haven/man/read_sas.html)(data\_file, catalog\_file = [**NULL**](https://rdrr.io/r/base/NULL.html), encoding = [**NULL**](https://rdrr.io/r/base/NULL.html),  catalog\_encoding = encoding, cols\_only = [**NULL**](https://rdrr.io/r/base/NULL.html))  [write\_sas](https://rdrr.io/cran/haven/man/read_sas.html)([data](https://rdrr.io/r/utils/data.html), path) |  Arguments  |  |  | | --- | --- | | **data\_file, catalog\_file** | Path to data and catalog files. The files are processed with readr::datasource(). | | **encoding, catalog\_encoding** | The character encoding used for the data\_file and catalog\_encoding respectively. A value of NULL uses the encoding specified in the file; use this argument to override it if it is incorrect. | | **cols\_only** | A character vector giving an experimental way to read in only specified columns. | | **data** | Data frame to write. | | **path** | Path to file where the data will be written. |  Value A tibble, data frame variant with nice defaults.  Variable labels are stored in the "label" attribute of each variable. It is not printed on the console, but the RStudio viewer will show it.  write\_sas() returns the input data invisibly. Examples  |  |  | | --- | --- | | 1  2 | path <- [system.file](https://rdrr.io/r/base/system.file.html)("examples", "iris.sas7bdat", package = "haven")  [read\_sas](https://rdrr.io/cran/haven/man/read_sas.html)(path) | |  | Run this example Create a free Jupyter Notebook Embed on your website |  Example output *# A tibble: 150 x 5*  Sepal\_Length Sepal\_Width Petal\_Length Petal\_Width Species  <dbl> <dbl> <dbl> <dbl> <chr>  1 5.1 3.5 1.4 0.2 setosa  2 4.9 3.0 1.4 0.2 setosa  3 4.7 3.2 1.3 0.2 setosa  4 4.6 3.1 1.5 0.2 setosa  5 5.0 3.6 1.4 0.2 setosa  6 5.4 3.9 1.7 0.4 setosa  7 4.6 3.4 1.4 0.3 setosa  8 5.0 3.4 1.5 0.2 setosa  9 4.4 2.9 1.4 0.2 setosa  10 4.9 3.1 1.5 0.1 setosa  *# ... with 140 more rows* |
| 3. How to read Weka Attribute-Relation File Format (ARFF) files in R?  4. How to read a heavy csv/tsv file using readr package? | read.arff: Read Data from ARFF FilesIn [RWeka: R/Weka Interface](https://rdrr.io/cran/RWeka/)Description Reads data from Weka Attribute-Relation File Format (ARFF) files. Usage  |  |  | | --- | --- | | 1 | [read.arff](https://rdrr.io/cran/RWeka/man/read.arff.html)([file](https://rdrr.io/r/base/connections.html)) |  Arguments  |  |  | | --- | --- | | **file** | a character string with the name of the ARFF file to read from, or a connection which will be opened if necessary, and if so closed at the end of the function call. |  Value A data frame containing the data from the ARFF file. Examples  |  |  | | --- | --- | | 1  2 | [read.arff](https://rdrr.io/cran/RWeka/man/read.arff.html)([system.file](https://rdrr.io/r/base/system.file.html)("arff", "contact-lenses.arff",  package = "RWeka")) |  Example output OpenJDK 64-Bit Server VM [warning](https://rdrr.io/r/base/warning.html): Can't detect initial thread stack location - find\_vma failed  age spectacle-prescrip astigmatism tear-[prod](https://rdrr.io/r/base/prod.html)-rate contact-lenses  1 young myope no reduced none  2 young myope no normal soft  3 young myope yes reduced none  4 young myope yes normal hard  5 young hypermetrope no reduced none  6 young hypermetrope no normal soft  7 young hypermetrope yes reduced none  8 young hypermetrope yes normal hard  9 pre-presbyopic myope no reduced none  10 pre-presbyopic myope no normal soft  11 pre-presbyopic myope yes reduced none  12 pre-presbyopic myope yes normal hard  13 pre-presbyopic hypermetrope no reduced none  14 pre-presbyopic hypermetrope no normal soft  15 pre-presbyopic hypermetrope yes reduced none  16 pre-presbyopic hypermetrope yes normal none  17 presbyopic myope no reduced none  18 presbyopic myope no normal none  19 presbyopic myope yes reduced none  20 presbyopic myope yes normal hard  21 presbyopic hypermetrope no reduced none  22 presbyopic hypermetrope no normal soft  23 presbyopic hypermetrope yes reduced none  24 presbyopic hypermetrope yes normal none  Warning [message](https://rdrr.io/r/base/message.html):  [system](https://rdrr.io/r/base/system.html) [call](https://rdrr.io/r/base/call.html) failed: Cannot allocate memory   |  |  | | --- | --- | | read.arff {foreign} | R Documentation |  Read Data from ARFF FilesDescription Reads data from Weka Attribute-Relation File Format (ARFF) files. Usage read.arff(file) Arguments  |  |  | | --- | --- | | file | a character string with the name of the ARFF file to read from, or a [connection](https://stat.ethz.ch/R-manual/R-devel/library/base/html/connections.html) which will be opened if necessary, and if so closed at the end of the function call. |  Value A data frame containing the data from the ARFF file.   * library(readr) * read\_csv(): comma separated (CSV) files * read\_tsv(): tab separated files * mtcars <- read\_csv(readr\_example("mtcars.csv")) * #> Parsed with column specification: * #> cols( * #> mpg = col\_double(), * #> cyl = col\_double(), * #> disp = col\_double(), * #> hp = col\_double(), * #> drat = col\_double(), * #> wt = col\_double(), * #> qsec = col\_double(), * #> vs = col\_double(), * #> am = col\_double(), * #> gear = col\_double(), * #> carb = col\_double() * #> ) * Note that readr prints the column specification. This is useful because it allows you to check that the columns have been read in as you expect, and if they haven’t, you can easily copy and paste into a new call: * mtcars <- read\_csv(readr\_example("mtcars.csv"), col\_types = * cols( * mpg = col\_double(), * cyl = col\_integer(), * disp = col\_double(), * hp = col\_integer(), * drat = col\_double(), * vs = col\_integer(), * wt = col\_double(), * qsec = col\_double(), * am = col\_integer(), * gear = col\_integer(), * carb = col\_integer() * ) * ) |