Package 'ProjectTemplate'

August 9, 2017

Type Package

Title Automates the Creation of New Statistical Analysis Projects

Version 0.8

Date 2017-08-02

Description Provides functions to
automatically build a directory structure for a new R
project. Using this structure, 'ProjectTemplate'
automates data loading, preprocessing, library
importing and unit testing.

License Artistic-2.0

LazyLoad yes

Depends R (>= 2.7)

Suggests foreign, reshape, plyr, digest, formatR, stringr, ggplot2,
lubridate, log4r (>= 0.1-5), DBI, RMySQL, RSQLite, gdata,
RODBC, RJDBC, xlsx, tuneR, pixmap, data.table, RPostgreSQL,
GetoptLong, whisker, testthat (>= 0.8)

URL http://projecttemplate.net

BugReports https://github.com/johnmyleswhite/ProjectTemplate/issues

Collate 'add.config.R' 'xport.reader.R' 'xlsx.reader.R' 'xls.reader.R'
'wsv.reader.R' 'url.reader.R' 'tsv.reader.R' 'systat.reader.R'
'stata.reader.R' 'require.package.R' 'sql.reader.R'
'spss.reader.R' 'rdata.reader.R' 'r.reader.R' 'ppm.reader.R'
'octave.reader.R' 'mtp.reader.R' 'mp3.reader.R' 'file.reader.R'
'epiinfo.reader.R' 'dbf.reader.R' 'db.reader.R' 'csv2.reader.R'
'csv.reader.R' 'arff.reader.R' 'preinstalled.readers.R'
'add.extension.R' 'cache.R' 'cache.name.R' 'cache.project.R'
'clean.variable.name.R' 'clear.R' 'clear.cache.R'
'translate.dcf.R' 'config.R' 'create.project.R' 'get.project.R'
'help.R' 'list.data.R' 'load.project.R' 'migrate.project.R'
'project.config.R' 'reload.project.R' 'run.project.R'
'show.project.R' 'stopifnotproject.R' 'stub.tests.R'
'test.project.R'

RoxygenNote 6.0.1

NeedsCompilation no

Author Aleksandar Blagotic [ctb],
Diego Valle-Jones [ctb],
Jeffrey Breen [ctb],
Joakim Lundborg [ctb],
John Myles White [aut, cph],
Josh Bode [ctb],
Kenton White [ctb, cre],
Kirill Mueller [ctb],
Matteo Redaelli [ctb],
Noah Lorang [ctb],
Patrick Schalk [ctb],
Dominik Schneider [ctb]

Maintainer Kenton White <jkentonwhite@gmail.com>

Repository CRAN

Date/Publication 2017-08-09 04:11:27 UTC

R topics documented:

`.add.extension` *Associate a reader function with an extension.*

Description

This function will associate an extension with a custom reader function.

Usage

```
.add.extension(extension, reader)
```

Arguments

`extension`	The extension of the new data file.
`reader`	The function to use when reading the data file. It should accept three arguments: `data.file`, `filename` and `variable.name` (in that order). The function should read the contents of the file `filename`, and save it into the workspace under the name `variable.name`. The `data.file` argument is just a relative file name and can be ignored.

Value

No value is returned; this function is called for its side effects.

Warning

This interface should not be considered as stable and is likely to be replaced by a different mechanism in a forthcoming version of this package.

Examples

```
## Not run: .add.extension('foo', foo.reader)
```

`add.config` *Add project specific config to the global config*

Description

Enables project specific configuation to be added to the global config object. The allowable format is key value pairs which are appended to the end of the `config` object, which is accessible from the global environment.

Usage

```
add.config(...)
```

Arguments

`...`	A series of key-value pairs containing the configuration. The key is the name that gets added to the config object.

Details

Once defined, the value can be accessed from any `ProjectTemplate` script by referencing `config$my_project_var`.

Examples

```
library('ProjectTemplate')
## Not run:
add.config(
    keep_bigdata=TRUE,     # Whether to keep the big data file in memory
    parse=7                # number of fields to parse
)

if (config$keep_bigdata) ...

## End(Not run)
```

`arff.reader` *Read the Weka file format.*

Description

This function will load a data set stored in the Weka file format into the specified global variable binding.

Usage

```
arff.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
## Not run: arff.reader('example.arff', 'data/example.arff', 'example')
```

`cache` *Cache a data set for faster loading.*

Description

This function will store a copy of the named data set in the `cache` directory. This cached copy of the data set will then be given precedence at load time when calling `load.project`. Cached data sets are stored as `.RData` files.

Usage

```
cache(variable = NULL, CODE = NULL, depends = NULL, ...)
```

Arguments

variable	A character string containing the name of the variable to be saved. If the CODE parameter is defined, it is evaluated and saved, otherwise the variable with that name in the global environment is used.
CODE	A sequence of R statements enclosed in {..} which produce the object to be cached.
depends	A character vector of other global environment objects that the CODE depends upon. Caching will be forced if those objects have changed since last caching
...	additional arguments passed to save

Details

Usually you will want to cache datasets during munging. This can be the raw data just loaded, or it can be the result of further processing during munge. Either way, it can take a while to cache large variables, so cache will only cache when it needs to. The clear.cache("variable") command can be run to flush individual items from the cache.

Calling cache() with no arguments returns the current status of the cache.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')
## Not run: create.project('tmp-project')

setwd('tmp-project')

dataset1 <- 1:5
cache('dataset1')

setwd('..')
unlink('tmp-project')
## End(Not run)
```

---

cache.name *Translate a variable name into a file name for caching.*

Description

This function will translate a variable name into a form that is suitable as a filename on most OS's.

Usage

```
cache.name(data.filename)
```

Arguments

`data.filename` The variable name to be translated into a filename.

Value

A translated variable name.

Examples

```
library('ProjectTemplate')

## Not run: cache.name('example.1')
```

---

cache.project *Cache a project's data sets in binary format.*

Description

This function will cache all of the data sets that were loaded by the `load.project` function in a binary format that is easier to load quickly. This is particularly useful for data sets that you've modified during a slow munging process that does not need to be repeated.

Usage

```
cache.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

`create.project`, `load.project`, `get.project`, `show.project`

Examples

```
library('ProjectTemplate')
## Not run: load.project()

cache.project()
## End(Not run)
```

clean.variable.name *Translate a file name into a valid R variable name.*

Description

This function will translate a file name into a name that is a valid variable name in R. Non-alphabetic characters on the boundaries of the file name will be stripped; non-alphabetic characters inside of the file name will be replaced with dots.

Usage

```
clean.variable.name(variable.name)
```

Arguments

variable.name	A character vector containing a variable's proposed name that should be standardized.

Value

A translated variable name.

Examples

```
library('ProjectTemplate')

## Not run: clean.variable.name('example_1')
```

clear *Clear objects from the global environment*

Description

This function removes specific (or all by default) named objects from the global environment. If used within a ProjectTemplate project, then any variables defined in the config$sticky_variables will remain.

Usage

```
clear(..., keep = c(), force = FALSE)
```

Arguments

...	A sequence of character strings of the objects to be removed from the global environment. If none given, then all items except those in keep will be deleted. This includes items beginning with .
keep	A character vector of variables that should remain in the global environment
force	If TRUE, then variables will be deleted even if specifed in keep or config$sticky_variables

Value

The variables kept and removed are reported

Examples

```
library('ProjectTemplate')
## Not run:
clear("x", "y", "z")
clear(keep="a")
clear()

## End(Not run)
```

`clear.cache` *Clear data sets from the cache*

Description

This function remove specific (or all by default) named data sets from the `cache` directory. This will force that data to be read in from the `data` directory next time `load.project` is called.

Usage

```
clear.cache(...)
```

Arguments

`...`	A sequence of character strings of the variables to be removed from the cache. If none given, then all items in the cache will be removed.

Value

Success or failure is reported

Examples

```
library('ProjectTemplate')
## Not run:
clear.cache("x", "y", "z")

## End(Not run)
```

`create.project` *Create a new project.*

Description

This function will create all of the scaffolding for a new project. It will set up all of the relevant directories and their initial contents. For those who only want the minimal functionality, the `minimal` argument can be set to `TRUE` to create a subset of ProjectTemplate's default directories. For those who want to dump all of ProjectTemplate's functionality into a directory for extensive customization, the `dump` argument can be set to `TRUE`.

Usage

```
create.project(project.name = "new-project", minimal = FALSE,
  dump = FALSE, merge.strategy = c("require.empty", "allow.non.conflict"))
```

Arguments

`project.name`	A character vector containing the name for this new project. Must be a valid directory name for your file system.
`minimal`	A boolean value indicating whether to create a minimal project or a full project. A minimal project contains only the directories strictly necessary to use ProjectTemplate and does not provide template code for profiling, unit testing or documenting your project.
`dump`	A boolean value indicating whether the entire functionality of ProjectTemplate should be written out to flat files in the current project.
`merge.strategy`	What should happen if the target directory exists and is not empty? If `"force.empty"`, the target directory must be empty; if `"allow.non.conflict"`, the method succeeds if no files or directories with the same name exist in the target directory.

Details

If the target directory does not exist, it is created. Otherwise, it can only contain files and directories allowed by the merge strategy.

Value

No value is returned; this function is called for its side effects.

See Also

`load.project`, `get.project`, `cache.project`, `show.project`

Examples

```
library('ProjectTemplate')

## Not run: create.project('MyProject')
```

csv.reader *Read a comma separated values (.csv) file.*

Description

This function will load a data set stored in the CSV file format into the specified global variable binding.

Usage

```
csv.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: csv.reader('example.csv', 'data/example.csv', 'example')
```

csv2.reader *Read a semicolon separated values (.csv2) file.*

Description

This function will load a data set stored in the CSV2 file format into the specified global variable binding. The default behaviour will soon change to actually use R's read.csv2() function, assuming dec=".", sep=";".

Usage

```
csv2.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: csv2.reader('example.csv2', 'data/example.csv2', 'example')
```

`db.reader` *Read a SQlite3 database with a (.db) file extension.*

Description

This function will load all of the data sets stored in the SQlite3 database into the global environment. If you want to specify a single table or query to execute against the database, move it elsewhere and use a .sql file interpreted by `sql.reader`.

Usage

```
db.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: db.reader('example.db', 'data/example.db', 'example')
```

dbf.reader *Read an XBASE file with a .dbf file extension.*

Description

This function will load all of the data sets stored in the specified XBASE file into the global environment.

Usage

```
dbf.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: dbf.reader('example.dbf', 'data/example.dbf', 'example')
```

epiinfo.reader *Read an Epi Info file with a .rec file extension.*

Description

This function will load all of the data sets stored in the specified Epi Info file into the global environment.

Usage

```
epiinfo.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: epiinfo.reader('example.rec', 'data/example.rec', 'example')
```

`file.reader` *Read an arbitrary file described in a .file file.*

Description

This function will load all of the data sets described in the specified .file file into the global environment. A .file file must contain DCF that specifies the path to the data set and which extension should be used from the dispatch table to load the data set.

Usage

```
file.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Details

Examples of the DCF format and settings used in a .file file are shown below:

path: http://www.johnmyleswhite.com/ProjectTemplate/sample_data.csv extension: csv

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: file.reader('example.file', 'data/example.file', 'example')
```

get.project *Show information about the current project.*

Description

This function will return all of the information that ProjectTemplate has about the current project. This information is gathered when `load.project` is called. At present, ProjectTemplate keeps a record of the project's configuration settings, all packages that were loaded automatically and all of the data sets that were loaded automatically. The information about autoloaded data sets is used by the `cache.project` function.

Usage

```
get.project()
```

Details

In previous releases this information has been available through the global variable `project.info`. Using this variable is now deprecated and will result in a warning.

Value

A named list.

See Also

`create.project`, `load.project`, `cache.project`, `show.project`

Examples

```
library('ProjectTemplate')

## Not run: load.project()

get.project()
## End(Not run)
```

list.data *Listing the data for the current project*

Description

This function produces a data.frame of all data files in the project, with meta data on if and how the file will be loaded by `load.project`.

Usage

```
list.data(override.config = NULL)
```

Arguments

override.config	Named list, allows overriding individual configuration items.

Details

The returned data.frame contains the following variables, with one observation per file in data/:

filename	Character variable containing the filename relative to data/ directory.
varname	Character variable containing the name of the variable into which the file will be imported. *
is_ignored	Logical variable that indicates whether the file. is ignored through the data_ignore option in the configurat
is_directory	Logical variable that indicates whether the file is a directory.
is_cached	Logical variable that indicates whether the file is already available in the cache/ directory.
cached_only	Logical variable that indicates whether the variable is only available in the cache/ directory. This occurs wh
reader	Character variable containing the name of the reader function that will be used to load the data. Contains a c

* Note that some readers return more than one variable, usually with the listed variablename as prefix. This is true for for example the xls.reader and xlsx.reader.

Value

A data.frame listing the available data, with relevant meta data

See Also

load.project, show.project, project.config

Examples

```
library('ProjectTemplate')

## Not run: list.data()
```

---

load.project *Automatically load data and packages for a project.*

Description

This function automatically load all of the data and packages used by the project from which it is called. The behaviour can be controlled by adjusting the project.config configuration.

Usage

```
load.project(override.config = NULL)
```

Arguments

override.config
: Named list, allows overriding individual configuration items.

Value

No value is returned; this function is called for its side effects.

See Also

create.project, get.project, cache.project, show.project, project.config

Examples

```
library('ProjectTemplate')

## Not run: load.project()
```

---

migrate.project *Migrates a project from a previous version of ProjectTemplate*

Description

This function automatically performs all necessary steps to migrate an existing project so that it is compatible with this version of ProjectTemplate

Usage

```
migrate.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

create.project

Examples

```
library('ProjectTemplate')

## Not run: migrate.project()
```

mp3.reader — *Read an MP3 file with a .mp3 file extension.*

Description

This function will load the specified MP3 file into memory using the tuneR package. This is useful for working with music files as a data set.

Usage

```
mp3.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: mp3.reader('example.mp3', 'data/example.mp3', 'example')
```

mtp.reader — *Read a Minitab Portable Worksheet with an .mtp3 file extension.*

Description

This function will load the specified Minitab Portable Worksheet into memory.

Usage

```
mtp.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: mtp.reader('example.mtp', 'data/example.mtp', 'example')
```

---

octave.reader *Read an Octave file with a .m file extension.*

Description

This function will load the specified Octave file into memory.

Usage

```
octave.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: octave.reader('example.m', 'data/example.m', 'example')
```

ppm.reader *Read a PPM file with a .ppm file extension.*

Description

This function will load the specified PPM file into memory using the pixamp package. This is useful for working with image files as a data set.

Usage

```
ppm.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: ppm.reader('example.ppm', 'data/example.ppm', 'example')
```

preinstalled.readers *Maps file types to the reader functions used to autoload them.*

Description

This list stores a mapping from regular expressions that match file extensions for the file types supported by ProjectTemplate to the reader functions that implement autoloading for those formats. Any new file type must be appended to this dispatch table.

Usage

```
preinstalled.readers
```

Format

An object of class list of length 53.

project.config *ProjectTemplate Configuration file*

Description

Every ProjectTemplate project has a configuration file found at config/global.dcf that contains various options that can be tweaked to control runtime behaviour. The valid options are shown below, and must be encoded using the DCF format.

Usage

```
project.config()
```

Details

Calling the project.config() function will display the current project configuration.

The options that can be configured in the config/global.dcf are shown below

data_loading	This can be set to 'on' or 'off'. If data_loading is on, the system will load data from both the
data_loading_header	This can be set to 'on' or 'off'. If data_loading_header is on, the system will load text data f
data_ignore	A comma separated list of files to be ignored when importing from the data/ directory. Reg
cache_loading	This can be set to 'on' or 'off'. If cache_loading is on, the system will load data from the ca
recursive_loading	This can be set to 'on' or 'off'. If recursive_loading is on, the system will load data from the
munging	This can be set to 'on' or 'off'. If munging is on, the system will execute the files in the mun
logging	This can be set to 'on' or 'off'. If logging is on, a logger object using the log4r package is au
logging_level	The value of logging_level is passed to a logger object using the log4r package during loggi
load_libraries	This can be set to 'on' or 'off'. If load_libraries is on, the system will load all of the R packa
libraries	This is a comma separated list of all the R packages that the user wants to automatically load
as_factors	This can be set to 'on' or 'off'. If as_factors is on, the system will convert every character ve
data_tables	This can be set to 'on' or 'off'. If data_tables is on, the system will convert every data set lo
attach_internal_libraries	This can be set to 'on' or 'off'. If attach_internal_libraries is on, then every time a new pack
cache_loaded_data	This can be set to 'on' or 'off'. If cache_loaded_data is on, then data loaded from the data d
sticky_variables	This is a comma separated list of any project-specific variables that should remain in the glo

If the config/globals.dcf is missing some items (for example because it was created under an old version of ProjectTemplate, then the following configuration is used for any missing items during load.project():

```
data_loading               TRUE
data_loading_header        TRUE
data_ignore
cache_loading              TRUE
recursive_loading          FALSE
munging                    TRUE
logging                    FALSE
logging_level              INFO
```

```
load_libraries              FALSE
libraries                   reshape, plyr, ggplot2, stringr, lubridate
as_factors                  TRUE
data_tables                 FALSE
attach_internal_libraries   TRUE
cache_loaded_data           FALSE
sticky_variables            NONE
```

When a new project is created using `create.project()`, the following values are pre-populated:

```
version                     0.8
data_loading                TRUE
data_loading_header         TRUE
data_ignore
cache_loading               TRUE
recursive_loading           FALSE
munging                     TRUE
logging                     FALSE
logging_level               INFO
load_libraries              FALSE
libraries                   reshape, plyr, dplyr, ggplot2, stringr, lubridate
as_factors                  TRUE
data_tables                 FALSE
attach_internal_libraries   FALSE
cache_loaded_data           TRUE
sticky_variables            NONE
```

Value

The current project configuration is displayed.

See Also

load.project

`ProjectTemplate` *Automates the creation of new statistical analysis projects.*

Description

ProjectTemplate provides functions to automatically build a directory structure for a new R project. Using this structure, ProjectTemplate automates data loading, preprocessing, library importing and unit testing.

References

This code is inspired by the skeleton structure used by Ruby on Rails.

Examples

```
library('ProjectTemplate')

## Not run: create.project('project_name')

setwd('project_name')
load.project()
## End(Not run)
```

`r.reader` *Read an R source file with a .R file extension.*

Description

This function will call source on the specified R file, executing the code inside of it as a way of generating data sets dynamically, as in many Monte Carlo applications.

Usage

```
r.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: r.reader('example.R', 'data/example.R', 'example')
```

rdata.reader *Read an RData file with a .rdata or .rda file extension.*

Description

This function will load the specified RData file into memory using the load function. This may generate many data sets simultaneously.

Usage

```
rdata.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: rdata.reader('example.RData', 'data/example.RData', 'example')
```

reload.project *Reload or reset a project*

Description

This function will clear the global environment and reload a project. This is useful when you've updated your data sets or changed your preprocessing scripts. Any sticky_variables configuration parameter in project.config will remain both in memory and (if present) in the cache by default. If the reset parameter is TRUE, then all variables are cleared from both the global environment and the cache.

Usage

```
reload.project(..., reset = FALSE)
```

Arguments

...	Optional parameters passed to load.project
reset	A boolean value, which if set TRUE clears the cache and everything in the global environment, including any sticky_variables

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: load.project()

reload.project()
## End(Not run)
```

---

require.package *Require a package for use in the project*

Description

This functions will require the given package. If the package is not installed it will stop execution and print a message to the user instructing them which package to install and which function caused the error.

Usage

```
require.package(package.name, attach = TRUE)

.require.package(package.name)
```

Arguments

package.name	A character vector containing the package name. Must be a valid package name installed on the system.
attach	Should the package be attached to the search path (as with library) or not (as with loadNamespace)? Defaults to TRUE. (Internal code will use FALSE by default unless a compatibility switch is set, see below.)

Details

The function .require.package is called by internal code. It will attach the package to the search path (with a warning) only if the compatibility configuration attach_internal_libraries is set to TRUE. Normally, packages used for loading data are not needed on the search path, but not loading them might break existing code. In a forthcoming version this compatibility setting will be removed, and no packages will be attached to the search path by internal code.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: require.package('PackageName')
```

---

run.project *Run all of the analyses in the* src *directory.*

Description

This function will run each of the analyses in the `src` directory in separate processes. At present, this is done serially, but future versions of this function will provide a means of running the analyses in parallel.

Usage

```
run.project()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: run.project()
```

---

show.project *Show information about the current project.*

Description

This function will show the user all of the information that ProjectTemplate has about the current project. This information is gathered when `load.project` is called. At present, ProjectTemplate keeps a record of the project's configuration settings, all packages that were loaded automatically and all of the data sets that were loaded automatically. The information about autoloaded data sets is used by the `cache.project` function.

Usage

```
show.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

create.project, load.project, get.project, cache.project

Examples

```
library('ProjectTemplate')

## Not run: load.project()

show.project()
## End(Not run)
```

---

spss.reader *Read an SPSS file with a .sav file extension.*

Description

This function will load the specified SPSS file into memory. It will convert the resulting list object into a data frame before inserting the data set into the global environment.

Usage

```
spss.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: spss.reader('example.sav', 'data/example.sav', 'example')
```

`sql.reader` *Read a database described in a .sql file.*

Description

This function will load data from a SQL database based on configuration information found in the specified .sql file. The .sql file must specify a database to be accessed. All tables from the database, one specific tables or one specific query against any set of tables may be executed to generate a data set.

Usage

```
sql.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Details

queries can support string interpolation to execute code snippets using mustache syntax (http://mustache.github.io). This is used to create queries that depend on data from other sources. Code delimited is {{...}}

Example: query: SELECT * FROM my_table WHERE id IN ({{ids}}). Here ids is a vector previously loaded into the Global Environment through ProjectTemplate

Examples of the DCF format and settings used in a .sql file are shown below:

Example 1 type: mysql user: sample_user password: sample_password host: localhost dbname: sample_database table: sample_table

Example 2 type: mysql user: sample_user password: sample_password host: localhost port: 3306 socket: /Applications/MAMP/tmp/mysql/mysql.sock dbname: sample_database table: sample_table

Example 3 type: sqlite dbname: /path/to/sample_database table: sample_table

Example 4 type: sqlite dbname: /path/to/sample_database query: SELECT * FROM users WHERE user_active == 1

Example 5 type: sqlite dbname: /path/to/sample_database table: *

Example 6 type: postgres user: sample_user password: sample_password host: localhost dbname: sample_database table: sample_table

Example 7 type: odbc dsn: sample_dsn user: sample_user password: sample_password dbname: sample_database query: SELECT * FROM sample_table

Example 8 type: oracle user: sample_user password: sample_password dbname: sample_database table: sample_table

Example 9 type: jdbc class: oracle.jdbc.OracleDriver classpath: /path/to/ojdbc5.jar (or set in CLASSPATH) user: scott password: tiger url: jdbc:oracle:thin:@myhost:1521:orcl query: select * from emp

Example 10 type: heroku classpath: /path/to/jdbc4.jar (or set in CLASSPATH) user: scott password: tiger host: heroku.postgres.url port: 1234 dbname: herokudb query: select * from emp

Example 11 In this example RSQLite::initExtension() is automatically called on the established connection.

Liam Healy has written extension-functions.c, which is available on http://www.sqlite.org/contrib. It provides mathematical and string extension functions for SQL queries using the loadable extensions mechanism.

type: sqlite dbname: /path/to/sample_database plugin: extension query: SELECT *,STDEV(value1) FROM example_table

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: sql.reader('example.sql', 'data/example.sql', 'example')
```

`stata.reader` *Read a Stata file with a .stata file extension.*

Description

This function will load the specified Stata file into memory.

Usage

```
stata.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: stata.reader('example.stata', 'data/example.stata', 'example')
```

stub.tests *Generate unit tests for your helper functions.*

Description

This function will parse all of the functions defined in files inside of the `lib` directory and will generate a trivial unit test for each function. The resulting tests are stored in the file `tests/autogenerated.R`. Every test is excepted to fail by default, so you should edit them before calling `test.project`.

Usage

```
stub.tests()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: stub.tests()
```

systat.reader *Read a Systat file with a .sys or .syd file extension.*

Description

This function will load the specified Systat file into memory.

Usage

```
systat.reader(data.file, filename, variable.name)
```

Arguments

`data.file`	The name of the data file to be read.
`filename`	The path to the data set to be loaded.
`variable.name`	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: systat.reader('example.sys', 'data/example.sys', 'example')
```

---

test.project *Run all unit tests for this project.*

Description

This function will run all of the `testthat` style unit tests for the current project that are defined inside of the `tests` directory. The tests will be run in the order defined by the filenames for the tests: it is recommend that each test begin with a number specifying its position in the sequence.

Usage

```
test.project()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: load.project()

test.project()
## End(Not run)
```

---

translate.dcf *Read a DCF file into an R list.*

Description

This function will read a DCF file and translate the resulting data frame into a list. The DCF format is used throughout ProjectTemplate for configuration settings and ad hoc file format specifications.

Usage

```
translate.dcf(filename)
```

Arguments

`filename`	A character vector specifying the DCF file to be translated.

Details

The content of the DCF file are stored as character strings. If the content is placed between the back tick character , then the content is evaluated as R code and the result returned in a string

Value

Returns a list containing the entries from the DCF file.

Examples

```
library('ProjectTemplate')

## Not run: translate.dcf(file.path('config', 'global.dcf'))
```

---

tsv.reader *Read a tab separated values (.tsv or .tab) file.*

Description

This function will load a data set stored in the TSV file format into the specified global variable binding.

Usage

```
tsv.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: tsv.reader('example.tsv', 'data/example.tsv', 'example')
```

url.reader *Read a remote file described in a .url file.*

Description

This function will load data from a remote source accessible through HTTP or FTP based on configuration information found in the specified .url file. The .url file must specify the URL of the remote data source and the type of data that is available remotely. Only one data source per .url file is supported currently.

Usage

```
url.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Details

Examples of the DCF format and settings used in a .url file are shown below:

Example 1 url: http://www.johnmyleswhite.com/ProjectTemplate/sample_data.csv separator: ,

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: url.reader('example.url', 'data/example.url', 'example')
```

wsv.reader *Read a whitespace separated values (.wsv or .txt) file.*

Description

This function will load a data set stored in the WSV file format into the specified global variable binding.

Usage

```
wsv.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: wsv.reader('example.wsv', 'data/example.wsv', 'example')
```

---

xls.reader *Read an Excel 2004 file with a .xls file extension.*

Description

This function will load the specified Excel file into memory using the gdata package. Each sheet of the Excel workbook will be read into a separate variable in the global environment.

Usage

```
xls.reader(data.file, filename, workbook.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
workbook.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: xls.reader('example.xls', 'data/example.xls', 'example')
```

xlsx.reader *Read an Excel 2007 file with a .xlsx file extension.*

Description

This function will load the specified Excel file into memory using the xlsx package. Each sheet of the Excel workbook will be read into a separate variable in the global environment.

Usage

```
xlsx.reader(data.file, filename, workbook.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
workbook.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: xlsx.reader('example.xlsx', 'data/example.xlsx', 'example')
```

xport.reader *Read an XPort file with a .xport file extension.*

Description

This function will load the specified XPort file into memory.

Usage

```
xport.reader(data.file, filename, variable.name)
```

Arguments

data.file	The name of the data file to be read.
filename	The path to the data set to be loaded.
variable.name	The name to be assigned to in the global environment.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: xport.reader('example.xport', 'data/example.xport', 'example')
```

Index