Package: sasquatch (via r-universe)

August 18, 2025

Title Use 'SAS', R, and 'quarto' Together

Version 0.0.0.9045

Description Use R and 'SAS' within reproducible multilingual 'quarto' documents. Run 'SAS' code blocks interactively, send data back and forth between 'SAS' and R, and render 'SAS' output within quarto documents. 'SAS' connections are established through a combination of 'SASPy' and 'reticulate'.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements python (>= 3.4.0), SASPy, java (>= 7) required for IOM access method

Depends R (>= 4.1.0)

Imports cli, htmlwidgets, knitr, reticulate, rlang (>= 1.1.0), rstudioapi

Suggests curl, rmarkdown, testthat (>= 3.0.0), withr

URL https://docs.ropensci.org/sasquatch, https://github.com/ropensci/sasquatch

BugReports https://github.com/ropensci/sasquatch/issues

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs make default-jdk libpng-dev python3

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/sasquatch

RemoteRef main

RemoteSha 5dbca94ed0ddb9892cd5df0ffc7aa0390e8cf256

Contents

`configure_saspy` *Configure SASPy package*

Description

Adds `sascfg_personal.py` and `authinfo` files and prefills relevant info according to a specified template.

Usage

```
configure_saspy(template = c("none", "oda"), overwrite = FALSE)
```

Arguments

`template`	Default template to base configuration files off of.
`overwrite`	Can new configuration files overwrite existing config files (if they exist)?

Details

Configuration for SAS can vary greatly based on your computer's operating system and the SAS platform you wish to connect to (see `vignette("configuration")` for more information).

Regardless of your desired configuration, configuration always starts with the creation of a `sascfg_personal.py` file within the `SASPy` package installation. This will look like:

```
SAS_config_names = ['config_name']

config_name = {

}
```

SAS_config_names should contain a string list of the variable names of all configurations. Configurations are specified as dictionaries, and configuration parameters depend on the access method.

Additionally, some access methods will require an additional authentication file (.authinfo for Linux and Mac, _authinfo for Windows) stored in the user's home directory, which are constructed as follows:

```
config_name user {your username} password {your password}
```

Templates:

The "none" template simply creates a sascfg_personal.py file within the SASPy package installation.

The "oda" template will set up a configuration for SAS On Demand for Academics. The sascfg_personal.py and authinfo files will be automatically configured using the information you provide through prompts.

Value

No return value.

See Also

install_saspy()

Examples

```
## Not run:
config_saspy()

## End(Not run)
```

---

install_saspy *Install SASPy package*

Description

Installs the SASPy package and its dependencies within a virtual Python environment.

Behavior was derived from tensorflow::install_tensorflow().

Usage

```
install_saspy(
  method = c("auto", "virtualenv", "conda"),
  conda = "auto",
  envname = "r-saspy",
  extra_packages = NULL,
  restart_session = TRUE,
  conda_python_version = NULL,
  ...,
  pip_ignore_installed = FALSE,
  new_env = identical(envname, "r-saspy"),
  python_version = NULL
)
```

Arguments

method	By default, "auto" automatically finds a method that will work in the local environment. Change the default to force a specific installation method.
conda	The path to a conda executable. Use "auto" to allow reticulate to automatically find an appropriate conda binary.
envname	The name, or full path, of the environment in which Python packages are to be installed.
extra_packages	Additional packages to install.
restart_session	Restart session?
conda_python_version	Passed to conda (only applicable if method = "conda")
...	other arguments passed to reticulate::conda_install() or reticulate::virtualenv_install(), depending on the method used.
pip_ignore_installed	Should pip ignore installed python packages and reinstall all already installed python packages?
new_env	If TRUE, any existing Python virtual environment and/or conda environment specified by envname is deleted first.
python_version	Select the Python that will be used to create the virtualenv. Pass a string with version constraints like "3.8", or ">=3.9,<=3.11" or a file path to a python executable like "/path/to/bin/python3". The supplied value is passed on to reticulate::virtualenv_starter(). Note that SASPy requires a Python version of at least >=3.4.

Value

No return value.

See Also

configure_saspy()

Examples

```
## Not run:
install_saspy()

## End(Not run)
```

---

sas_connect *Establish SAS session*

Description

Starts a SAS session. This is required before doing anything!

Usage

```
sas_connect(cfgname, reconnect = FALSE)
```

Arguments

cfgname	string; Name of configuration to use from the SAS_config_names list within in sascfg_personal.py.
reconnect	logical; Establish a new connection if a connection already exists?

Details

All configurations are specified within the sascfg_personal.py file inside the SASPy package. For more information about SASPy configuration, check out the configuration documentation or vignette("configuration").

Value

No return value.

See Also

Other session management functions: sas_disconnect(), sas_get_session()

Examples

```
## Not run:
sas_connect(cfgname = "oda")

## End(Not run)
```

sas_disconnect *Disconnect SAS session*

Description

Disconnects from the current SAS session.

Usage

```
sas_disconnect()
```

Value

No return value.

See Also

Other session management functions: `sas_connect()`, `sas_get_session()`

Examples

```
## Not run:
sas_disconnect()

## End(Not run)
```

sas_engine *A SAS engine for* `knitr`

Description

Produces HTML or latex output for rending within quarto or rmarkdown documents.

Usage

```
sas_engine(options)
```

Arguments

`options`	Options from `knitr`.

Details

Will be activated by running `library(sasquatch)`

Supported `knitr` chunk options:

`sasquatch`'s engine implements may of the same options as the R engine in `knitr`, but not all.

- `eval` (Default: `TRUE`): Evaluate the code chunk (if false, just echos the code into the output)
- `echo` (Default: `TRUE`): Include the source code in output
- `output` (Default: `TRUE`): Include the results of executing the code in the output (`TRUE` or `FALSE`).
- `include` (Default: `TRUE`): Include any output (code or results).
- `capture` (Only within HTML; Default: `"both"`): If `"both"`, tabpanel with output and log included. If `"listing"`, only output is included. If `"log"` only log is included.

Value

`knitr` engine output.

Examples

```
# The below function is run internally within `sasquatch` on startup
knitr::knit_engines$set(sas = sas_engine)
```

---

sas_file_copy *Copy a file on SAS*

Description

Copies a file on the remote SAS server. Is analogous to `file.copy()`, but for the remote SAS server.

Usage

```
sas_file_copy(from_path, to_path)
```

Arguments

`from_path`	string; Path of file on remote SAS server to be copied.
`to_path`	string; Path of file on remote SAS server to copy to.

Value

`logical`; value indicating if the operation succeeded.

See Also

Other file management functions: `sas_file_download()`, `sas_file_exists()`, `sas_file_remove()`, `sas_file_upload()`, `sas_list()`

Examples

```
## Not run:
# connect to SAS
sas_connect()

# create a file and upload it to SAS
cat("PROC MEANS DATA = sashelp.cars;RUN;", file = "script.sas")
sas_file_upload(local_path = "script.sas", sas_path = "~/script.sas")

# copy file on SAS
sas_file_copy("~/script.sas", "~/script_copy.sas")

## End(Not run)
```

`sas_file_download` *Download a file from SAS*

Description

Downloads a file to the remote SAS server.

Usage

```
sas_file_download(sas_path, local_path)
```

Arguments

`sas_path`	string; Path of file on remote SAS server to be download
`local_path`	string; Path to upload SAS file to on local machine.

Value

`logical`; value indicating if the operation succeeded.

See Also

Other file management functions: `sas_file_copy()`, `sas_file_exists()`, `sas_file_remove()`, `sas_file_upload()`, `sas_list()`

Examples

```
## Not run:
# connect to SAS
sas_connect()

# create a file and upload it to SAS
cat("PROC MEANS DATA = sashelp.cars;RUN;", file = "script.sas")
sas_file_upload(local_path = "script.sas", sas_path = "~/script.sas")
```

```
# download file from SAS
sas_file_download(sas_path = "~/script.sas", local_path = "script.sas")

## End(Not run)
```

`sas_file_exists` *Check if file on SAS exists*

Description

Checks if a file exists on the remote SAS server. Is analogous to `file.exists()`, but for the remote SAS server.

Usage

```
sas_file_exists(path)
```

Arguments

`path`	string; Path of file on remote SAS server.

Value

`logical`; value indicating if the operation succeeded.

See Also

Other file management functions: `sas_file_copy()`, `sas_file_download()`, `sas_file_remove()`, `sas_file_upload()`, `sas_list()`

Examples

```
## Not run:
# connect to SAS
sas_connect()

# create a file and upload it to SAS
cat("PROC MEANS DATA = sashelp.cars;RUN;", file = "script.sas")
sas_file_upload(local_path = "script.sas", sas_path = "~/script.sas")

# check if file exists on SAS
sas_file_exists("~/script.sas")

## End(Not run)
```

sas_file_remove *Delete a file or directory from SAS*

Description

Deletes a file or directory from the remote SAS server. Is analogous to `file.remove()`, but for the remote SAS server.

Usage

```
sas_file_remove(path)
```

Arguments

`path`	string; Path of file on remote SAS server to be deleted.

Value

`logical`; value indicating if the operation succeeded.

See Also

Other file management functions: `sas_file_copy()`, `sas_file_download()`, `sas_file_exists()`, `sas_file_upload()`, `sas_list()`

Examples

```
## Not run:
# connect to SAS
sas_connect()

# create a file and upload it to SAS
cat("PROC MEANS DATA = sashelp.cars;RUN;", file = "script.sas")
sas_file_upload(local_path = "script.sas", sas_path = "~/script.sas")

# remove file from SAS
sas_file_remove(sas_path = "~/script.sas")

## End(Not run)
```

`sas_file_upload` *Upload a file to SAS*

Description

Uploads a file to the remote SAS server.

Usage

```
sas_file_upload(local_path, sas_path)
```

Arguments

`local_path`	string; Path of file on local machine to be uploaded.
`sas_path`	string; Path to upload local file to on the remote SAS server.

Value

`logical`; value indicating if the operation succeeded.

See Also

Other file management functions: `sas_file_copy()`, `sas_file_download()`, `sas_file_exists()`, `sas_file_remove()`, `sas_list()`

Examples

```
## Not run:
# connect to SAS
sas_connect()

# create a file to upload
cat("PROC MEANS DATA = sashelp.cars;RUN;", file = "script.sas")

# upload file
sas_file_upload(local_path = "script.sas", sas_path = "~/script.sas")

## End(Not run)
```

sas_from_r *Convert R table to SAS*

Description

Converts R table into a table in the current SAS session. R tables must only have logical, integer, double, factor, character, POSIXct, or Date class columns.

Usage

```
sas_from_r(x, table_name, libref = "WORK")
```

Arguments

x	data.frame; R table.
table_name	string; Name of table to be created in SAS.
libref	string; Name of libref to store SAS table within.

Details

SAS only has two data types (numeric and character). Data types are converted as follows:

- logical -> numeric
- integer -> numeric
- double -> numeric
- factor -> character
- character -> character
- POSIXct -> numeric (datetime; timezones are lost)
- Date -> numeric (date)

Value

data.frame; x.

See Also

sas_to_r()

Examples

```
## Not run:
sas_connect()
sas_from_r(mtcars, "mtcars")

## End(Not run)
```

sas_get_session *Get current SAS session*

Description

Returns the current SAS session, which can be used to extend sasquatch functionality or access the current session within Python.

Usage

```
sas_get_session()
```

Details

Extending sasquatch functionality:

SASPy has a wealth of functionality, some of which have not all been implemented within sasquatch. sas_get_session() offers a gateway to unimplemented functionality within the SASsession class.

Using Python:

When utilizing Python, R, and SAS, start the session within R using sas_connect() and utilize reticulate to pass the saspy.sasbase.SASsession object to Python.

Value

saspy.sasbase.SASsession; Current SAS session.

See Also

Other session management functions: sas_connect(), sas_disconnect()

Examples

```
## Not run:
sas_connect()
sas_get_session()

## End(Not run)
```

sas_list *List contents of a SAS directory*

Description

Lists the files or directories of a directory within the remote SAS server.

Usage

```
sas_list(path)
```

Arguments

path	string; Path of directory on remote SAS server to list the contents of.

Value

`character` vector; File or directory names.

See Also

Other file management functions: `sas_file_copy()`, `sas_file_download()`, `sas_file_exists()`, `sas_file_remove()`, `sas_file_upload()`

Examples

```
## Not run:
sas_connect()
sas_list(".")

## End(Not run)
```

sas_run_file *Execute SAS file*

Description

Execute a SAS file and render html output or save output as html and log.

Usage

```
sas_run_file(input_path, output_path, overwrite = FALSE)
```

Arguments

input_path	string; Path of SAS file to run.
output_path	optional string; Path to save html output to (log file will be named the same).
overwrite	logical; Can output overwrite prior output?

Value

If output_path specified, htmlwidget. Else, no return value.

See Also

Other code execution functions: sas_run_selected(), sas_run_string()

Examples

```
## Not run:
cat("PROC MEANS DATA = sashelp.cars;\n RUN;", file = "test.sas")

sas_connect()
sas_run_file("test.sas", "test.html")

## End(Not run)
```

---

sas_run_selected *Execute selected SAS code*

Description

Execute selected SAS code in current session and render html output as SAS widget. See vignette("overview") for more information on how to utilize the addin within RStudio or Positron.

Usage

```
sas_run_selected()
```

Value

htmlwidget; HTML5 output.

See Also

Other code execution functions: sas_run_file(), sas_run_string()

Examples

```
## Not run:
sas_connect()

# highlight something in the active editor of RStudio or Positron

sas_run_selected()

## End(Not run)
```

`sas_run_string` *Execute SAS code string*

Description

Execute SAS code in current session and render html output.

Usage

```
sas_run_string(input)
```

Arguments

`input`	string; SAS code to run.

Value

`htmlwidget`; HTML5 output.

See Also

Other code execution functions: `sas_run_file()`, `sas_run_selected()`

Examples

```
## Not run:
sas_connect()
sas_run_string("PROC MEANS DATA = sashelp.cars;RUN;")

## End(Not run)
```

sas_to_r *Convert SAS table to R*

Description

Converts table from current SAS session into a R `data.frame`.

Usage

```
sas_to_r(table_name, libref = "WORK")
```

Arguments

`table_name`	string; Name of table in SAS.
`libref`	string; Name of libref SAS table is stored within.

Details

SAS only has two data types (numeric and character). Data types are converted as follows:

- numeric -> double
- character -> character
- numeric (datetime, timezones are lost) -> POSIXct
- numeric (date) -> POSIXct

In the conversion process dates and datetimes are converted to local time. If utilizing another timezone, use `attr(date, "tzone") <-` or `lubridate::with_tz()` to convert back to the desired time zone.

Value

`data.frame` of the specified SAS table.

See Also

`sas_from_r()`

Examples

```
## Not run:
sas_connect()
cars <- sas_to_r("cars", "sashelp")

## End(Not run)
```

Index