Package 'domino'

May 8, 2026

Version 0.3.1

Type Package

Title R Console Bindings for the 'Domino Command-Line Client'

Date 2017-11-22

Author Jacek Glodek <jacek@theiterators.com>

Maintainer Nick Elprin <nick@dominodatalab.com>

Description A wrapper on top of the 'Domino Command-Line Client'. It lets you run 'Domino' commands (e.g., ``run'', ``upload'', ``download'') directly from your R environment. Under the hood, it uses R's system function to run the 'Domino' executable, which must be installed as a prerequisite. 'Domino' is a service that makes it easy to run your code on scalable hardware, with integrated version control and collaboration features designed for analytical workflows (see <http://www.dominodatalab.com> for more information).

License MIT + file LICENSE

SystemRequirements domino (~>1.7.1)

URL http://www.dominodatalab.com

RoxygenNote 5.0.1

Suggests testthat

Imports utils, grDevices

NeedsCompilation no

Repository CRAN

Date/Publication 2017-11-22 22:59:09 UTC

Contents

domino-package 2
domino.create 3
domino.debug 4
domino.diff 4

domino.download 4
domino.dump 5
domino.get 5
domino.init 6
domino.login 6
domino.reset 7
domino.run 7
domino.runCommand 8
domino.snapshot 9
domino.status 9
domino.sync 9
domino.upload 10

Index **11**

`domino-package` *Domino Data Lab R console bindings*

Description

The Domino R package is a wrapper on top of the Domino command-line client. It lets you run Domino commands (e.g., "run", "upload", "download") directly from your R environment. Under the hood, it uses R's system function to run the Domino executable, which must be installed as a prerequisite.

Details

Package: domino
Type: Package
Version: 0.2-7
Date: 2015-05-27
License: MIT
Author: Jacek Glodek jacek@theiterators.com
Maintainer: Nick Elprin nick@dominoup.com

References

Domino Data Lab support webpage - http://help.dominodatalab.com/

Examples

```
## Not run:
## logins as a given user to the Domino server
## and approves sending error reports to Domino.
domino.login("jglodek", "MySecretPassword", TRUE)

## creates new project.
domino.create("my-new-project")
```

```
## gets existing project from the server.
domino.get("jglodek/my-old-project")

## gets existing project from the server.
domino.get("my-old-project")

## initializes new domino project in current working directory with a given name.
domino.init("other-name")

## downloads run results from Domino server.
domino.download()

## uploads project files to Domino server.
domino.upload()

## runs main.r in the cloud with given arguments.
domino.run("main.r", "other", "console", "arguments")

## shows difference between current version and last uploaded version.
domino.diff()

## displays current run's status in the console.
domino.status()

## shows debug information
domino.debug()

## resets project defined in by current working directory
domino.reset()

## runs any of domino client command with given arguments
domino.runCommand("run my-file.r", successCallback, "failure message!")


## End(Not run)
```

`domino.create` *domino.create*

Description

Creates Domino project. Changes working directory to new project's one.

Usage

```
domino.create(projectName)
```

Arguments

`projectName`	String that will be the name of the new project.

Examples

```
## Not run:
# in directory ./
domino.create("my-new-project")
# current working directory is now switched to ./my-new-project and new
project is initialized.

## End(Not run)
```

`domino.debug` *domino.debug*

Description

Shows Domino debug information.

Usage

```
domino.debug()
```

`domino.diff` *domino.diff*

Description

Shows changes between your local version of project's data, and the version uploaded to the Domino servers.

Usage

```
domino.diff()
```

`domino.download` *domino.download*

Description

Downloads the latest Domino run results to the project's folder.

Usage

```
domino.download()
```

domino.dump *domino.dump*

Description

run dump command

Usage

```
domino.dump()
```

domino.get *domino.get*

Description

Downloads given project data from Domino. Changes working directory to the project's directory.

Usage

```
# Usage without username
domino.get("projectName")
```

Arguments

projectName	String containing project name. It can be prefixed by username and slash. Ex. "jglodek/quick-start".

Examples

```
## Not run:
# in directory ./
domino.get("my-project-in-the-cloud")
# current working directory is now switched to ./my-project-in-the-cloud
and the directory
# is filled with files from Domino server.

# The name of the project is prepended with username
    domino.get("jglodek/my-project-in-the-cloud")

## End(Not run)
```

domino.init *domino.init*

Description

Initializes new domino project in current directory. It can also set arbitrary name to the project, even if different from current directory name.

Usage

```
# inits a project inside current directory, with given name.
# ex. if run in ~/my_project, with "my_name" as a param,
# it will create a Domino project called my-param inside ~/my_project directory.
domino.init("projectName")
```

Arguments

projectName	Project name for the project that will be created in current working directory.

Examples

```
## Not run:
# in directory ./
domino.init("my-new-project")
# new project with name "my-new-project" is initialized inside current directory.

## End(Not run)
```

domino.login *domino.login*

Description

Logins to Domino server.

Usage

```
domino.login(usernameOrEmail, password,
approvalForSendingErrorReports=FALSE, host)
```

Arguments

usernameOrEmail	Login or e-mail address used when registering for Domino Data Lab account. Ex. "jglodek"
password	Secret password that was set for authenticating in Domino Data Lab server. If the password is not provided,a password prompt will be shown for interactive sessions. For non-interactive sessions, this arguments is required. Ex. "my-secret-password"
approvalForSendingErrorReports	Approval for the Domino client to send error reports to Domino in order to improve the product (these will NEVER include any of your data or source code). This defaults to FALSE. Ex. FALSE
host	The location of the domino server (this argument is optional) Ex. "https://app.dominodatalab.com"

Examples

```
## Not run:
domino.login("jglodek", TRUE)
domino.login("jglodek","my-super-secret-password", TRUE)
domino.login("jglodek","my-super-secret-password", TRUE, "https://app.dominodatalab.com")

## End(Not run)
```

---

domino.reset *domino.reset*

Description

Resets Domino project.

Usage

```
domino.reset()
```

---

domino.run *domino.run*

Description

Runs your project on Domino servers with given parameters.

Usage

```
domino.run(..., publishApiEndpoint=FALSE)
```

Arguments

...	All the run arguments will be joined together using space character. Ex. `domino.run("main.py", "-xvz", "my-file1.csv")`
publishApiEndpoint	Whether or not to republish the project's API endpoint at the end of the run.

Examples

```
## Not run:
my_data <- 4
domino.run("main.R","1","my-file1.csv", my_data)
#=> Runs "domino main.R 1 my-file1.csv 4"

## End(Not run)
```

---

domino.runCommand *domino.runCommand*

Description

Runs domino client command and runs success callback or shows failure message

Usage

```
domino.runCommand(commandAndArgs, successCallback = domino.OK,
failureMessage = "Executing the command failed", stdInput = FALSE)
```

Arguments

commandAndArgs	The commandAndArgs argument is a string that matches standard domino client's syntax,ex. "get quick-start" or "download".
successCallback	A function that will be called when domino command executes successfuly. Defaults to noop function.
failureMessage	A string representing failure message that should be printed when command fails. Has default value.
stdInput	Internal string data that is pushed to domino client's stdio streams. Default is no stdio stream input.

Examples

```
## Not run:
success <- function() {
print("Success!")
}
domino.runCommand("run main.R 1 2 3",success, "failed to succeed")
# Runs command "run main.R 1 2 3" and
```

```
# calls 'success' function if domino command ends successfuly.
# Prints error message  - "failed to succeed" if domino command fails.

## End(Not run)
```

`domino.snapshot` *domino.snapshot*

Description

take a snapshot of your command history and workspace.

Usage

```
domino.snapshot(commitMessage)
```

Arguments

`commitMessage`	A committ message to record with you snapshot

`domino.status` *domino.status*

Description

Shows status of current Domino project.

Usage

```
domino.status(...)
```

Arguments

`...`	Arguments normally passed to the domino status command.

`domino.sync` *domino.sync*

Description

Synchronizes the local project copy with the server project copy, by running 'download' followed by 'upload'.

Usage

```
domino.sync()
```

domino.upload *domino.upload*

Description

Uploads local version of Domino project files to the server.

Usage

```
domino.upload(commitMessage)
```

Arguments

commitMessage	Commit message for Domino client, explaining changes you did to the project code in recent update. Ex. "updated project files"

Index

∗ **command**
  domino.runCommand, 8
∗ **create**
  domino.create, 3
∗ **debug**
  domino.debug, 4
∗ **diff**
  domino.diff, 4
∗ **domino,**
  domino-package, 2
∗ **dominoup**
  domino-package, 2
∗ **download**
  domino.download, 4
∗ **get**
  domino.get, 5
∗ **init**
  domino.init, 6
∗ **login**
  domino.login, 6
∗ **package,**
  domino-package, 2
∗ **reset**
  domino.reset, 7
∗ **status**
  domino.status, 9
∗ **sync**
  domino.sync, 9
∗ **upload**
  domino.upload, 10

domino-package, 2
domino.create, 3
domino.debug, 4
domino.diff, 4
domino.download, 4
domino.dump, 5
domino.get, 5
domino.init, 6
domino.login, 6
domino.reset, 7
domino.run, 7
domino.runCommand, 8
domino.snapshot, 9
domino.status, 9
domino.sync, 9
domino.upload, 10