Package ‘dstack’

August 12, 2020

Type Package

Title Publishing Interactive Plots

Version 0.2.1

Description A native R package that allows to publish, share and track revisions of plots using your favorite plotting package, e.g. ‘ggplot2’. It also provides a kind of interactivity for such plots by specifying certain parameters for any specific plot view. See <https://docs.dstack.ai> for more information.

License GPL-3

URL https://dstack.ai

BugReports https://github.com/dstackai/dstack-r/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Imports uuid,bit64,rjson,httr,rlist,yaml,base64enc

Suggests ggplot2,plotly,data.table

NeedsCompilation no

Author Vitaly Khudobakhshov [aut, cre]

Maintainer Vitaly Khudobakhshov <vitaly@dstack.ai>

Repository CRAN

Date/Publication 2020-08-12 15:20:02 UTC

R topics documented:

auto_handler ..................................................... 2
commit .................................................................. 2
configure .................................................................. 3
create_frame .......................................................... 4
dataframe_handler .................................................. 6
datatable_handler .................................................. 6
ggplot_handler ....................................................... 7
auto_handler

*Handle Object Types Automatically*

**Description**

Handle Object Types Automatically

**Usage**

```r
auto_handler(use_plotly_instead_of_ggplot = TRUE)
```

**Arguments**

- `use_plotly_instead_of_ggplot`
  
  Use plot.ly to render ggplot2 charts. It is TRUE by default.

**commit**

*Commit Data to Stack Frame*

**Description**

Function adds a new view to the stack frame. Multiple views can be added to one frame, but in this case every plot must be supplied with certain parameters to distinguish one view from another. In the case of single plot parameters are not necessary. For multiple views parameters will be automatically converted to UI controls like sliders and drop down lists.
configure

Usage

commit(frame, obj, description = NULL, params = NULL, handler = auto_handler(), ...)

Arguments

frame A frame you want to commit.
obj A data to commit. Data will be preprocessed by the handler but dependently on auto_push mode will be sent to server or not. If auto_push is False then the data won’t be sent. Explicit push call need anyway to process committed data. auto_push is useful only in the case of multiple data objects in the stack frame, e.g. set of plots with settings.
description Description of the data.
params Parameters associated with this data, e.g. plot settings.
handler A handler which can be specified in the case of custom content, but by default it is auto_handler.
... Optional parameters is an alternative to params. If both are present this one will be merged into params.

Value

Changed frame.

Examples

library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame)) # print actual stack URL

configure Configure ‘dstack’

Description

Function allows to add or replace profile.
create_frame

Usage

configure(
  profile = "default",
  user,
  token,
  persist = c("global", "local", "in_place"),
  server = NULL,
  dstack_dir = .dstack_env$dstack_dir
)

Arguments

profile A name of profile. It will be "default" if not specified.
user Username in 'dstack'.
token A token. It can be obtained from https://dstack.ai web site.
persist Persistence level. It can be 'local' - this means that config will be stored in working directory, 'global' - config will be stored in user's home,cor 'in-place' - in this case config will be store in the memory and exists only while R session exists.
server Server to connect. By default it's NULL, so default API endpoint will be used.
dstack_dir Directory where dstack files are stored. By default it is '.dstack', so in the case of global persistence path to config file will be ~/.dstack/config.yaml.

create_frame Create a New Frame in Stack

Description

Frame is kind of revision of data user is going to publish. It consists of one or more views. Views are usually plots with some parameters to distinguish one plot from another. This function creates a frame and by default it checks permission to publish to this stack.

Usage

create_frame(
  stack,
  profile = "default",
  auto_push = FALSE,
  protocol = NULL,
  encryption = NULL,
  check_access = TRUE
)
create_frame

Arguments

  stack    A name of stack to use.
  profile  A profile refers to credentials, i.e. username and token. Default profile is named 'default'. The system is looking for specified profile as follows: it looks into working directory to find a configuration file (local configuration), if the file doesn’t exist it looks into user directory to find it (global configuration). The best way to manage profiles is to have dstack CLI tools installed. These tools are written in Python 3, so you have to install dstack support. In the case of PyPI you should type

      $ pip install dstack

  or

      $ conda install -c dstack.ai dstack

  We recommend to use local (virtual) environment to install the package. You can use this command in console:

      $ dstack config list

  to list existing profiles or add or replace token by following command

      $ dstack config add <PROFILE>

  or simply

      $ dstack config add

  if profile is not specified 'default' profile will be created. The system asks you about token from command line, make sure that you have already obtained token from the site.

  auto_push Tells the system to push frame just after commit. It may be useful if you want to see result immediately. Default is FALSE.

  protocol Protocol to use, usually it is NULL it means that json_protocol will be used.

  encryption This is a encryption method. By default is NULL and no encryption will be used.

  check_access Check access to specified stack, default is TRUE.

Value

New frame.

Examples

```
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame))  # print actual stack URL
```
**datatable_handler**

*Handle 'data.frame' Objects*

**Description**

Handle 'data.frame' Objects

**Usage**

```r
dataframe_handler(col_names = TRUE, row_names = FALSE)
```

**Arguments**

- `col_names`: Save column names. TRUE by default.
- `row_names`: Save row names. FALSE by default.

---

**datatable_handler**

*Handle 'data.table' Objects*

**Description**

Handle 'data.table' Objects

**Usage**

```r
datatable_handler(col_names = TRUE, row_names = FALSE)
```

**Arguments**

- `col_names`: Save column names. TRUE by default.
- `row_names`: Save row names. FALSE by default.
**ggplot_handler**

*Handles `ggplot2` Objects*

**Description**

Returns a function that converts `ggplot2` plots to appropriate format and format itself. PNG and SVG are supported.

**Usage**

```
ggplot_handler(dpi = 300, width = NA, height = NA, format = "png")
```

**Arguments**

- **dpi**: DPI, default is 300.
- **width**: Image width.
- **height**: Image height.
- **format**: Image format to use, can be "png" or "svg", by default PNG will be used.

**Value**

A list which contains conversion function and format itself.

---

**in_place_config**

*In-Place Configuration*

**Description**

It is used to configure dstack in R session without creating any configuration file on disk.

**Usage**

```
in_place_config()
```
is.datatable  

**Check That Specified Object is ‘data.table’ Object**

**Description**
Check That Specified Object is ‘data.table’ Object

**Usage**
is.datatable(x)

**Arguments**
x  An object to check.

is.ggplot2  

**Check That Specified Object is ‘ggplot2’ Chart**

**Description**
Check That Specified Object is ‘ggplot2’ Chart

**Usage**
is.ggplot2(x)

**Arguments**
x  An object to check.

is.plotly  

**Check That Specified Object is ‘plot.ly’ Chart**

**Description**
Check That Specified Object is ‘plot.ly’ Chart

**Usage**
is.plotly(x)

**Arguments**
x  An object to check.
**json_protocol**

**JSON Protocol Implementation to Connect API Server**

**Description**
Protocol is an abstraction which allows to send data to server. This function implements JSON-based protocol. It provides token in 'Authorization' header.

**Usage**

```python
call function
```

**Arguments**

- **server**: A server to connect.
- **error**: An error handling function.

**Value**

A function that implements JSON protocol.

---

**list_profiles**

**List Profiles**

**Description**
List Profiles

**Usage**

```python
list_profiles()
```

---

**plotly_handler**

**Handle 'plot.ly' Charts**

**Description**
Handle 'plot.ly' Charts

**Usage**

```python
plotly_handler()
```
pull

Pull data object from stack frame (head) which matches specified parameters.

Description

Pull data object from stack frame (head) which matches specified parameters.

Usage

```r
pull(
  stack,
  profile = "default",
  filename = NULL,
  error = .error,
  params = NULL,
  ...
)
```

Arguments

- **stack**: Stack you want to pull from.
- **profile**: Profile to use. ‘default’ will be used if profile is not specified.
- **filename**: Filename if you want to save downloaded file to disk. Lifespan of URL is limited by minutes, so filename is highly recommended for large files (> 5Mb), especially in the case of interactive computations. Specify the parameter in the case of non-text data.
- **error**: HTTP error handling function.
- **params**: Optional parameters to match.
- **...**: Parameters to match. Can be used as alternative to `params`. In the case of both are present this one will be merged into `params`.

Value

If filename is not NULL then it will be filename itself, otherwise it can be URL in the case of large files.

Examples

```r
df <- read.csv(pull("/public_datasets/fusionbase/covid19-germany", "Bundesland name"="All"))
summary(df)
```
**push**

### Push All Commits to Server

**Description**

This function is used to send a branch of committed plots to server or say server that operation with this frame is done. In the case of `auto_push` mode it sends only a total number of views in the frame. So call this method is obligatory to close the frame anyway.

**Usage**

```
push(frame, message = NULL)
```

**Arguments**

- `frame` A frame to push.
- `message` Push message. NULL by default.

**Value**

Stack URL.

**Examples**

```
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame)) # print actual stack URL
```

---

**push_frame**

### Creates a Frame, Commits and Pushes Data in a Single Operation

**Description**

In the case of one plot per push you can use do all operations in one call. This function creates a frame, commits view and pushes all data to server. The main difference in behaviour in this case is the function creates frame without permission check, so be sure that you have certain permission to push in the stack.

**Usage**

```
push_frame(stack = "diamonds")
```

**Arguments**

- `stack` The name of the stack to push.

**Value**

The URL of the stack.
Usage

```r
push_frame(
  stack,  
  obj,  
  description = NULL,  
  params = NULL,  
  message = NULL,  
  profile = "default",  
  handler = auto_handler(),  
  protocol = NULL,  
  encryption = .no_encryption,
  ...
)
```

Arguments

- **stack**: A name of stack to use.
- **obj**: Object to commit and push. e.g. plot.
- **description**: Optional description of the object.
- **params**: Optional parameters.
- **message**: Push message. NULL by default.
- **profile**: Profile you want to use, i.e. username and token. Default profile is 'default'.
- **handler**: Specify handler to handle the object, if it's None then `auto_handler` will be used.
- **protocol**: Protocol to use, usually it is NULL it means that `json_protocol` will be used.
- **encryption**: Encryption method by default `no_encryption` will be used.
- **...**: Optional parameters is an alternative to `params`. If both are present this one will be merged into `params`.

Value

Stack URL.

Examples

```r
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
push_frame("diamonds", image, "Diamonds bar chart")
```
use_config  Use Specified Configuration

Description
Use Specified Configuration

Usage
use_config(config_func)

Arguments
config_func A function to be used for configuration. It can be yaml_config or in_place_config.

Examples
use_config(yaml_config) # to use standard YAML configuration file
use_config(in_place_config) # to use "in-place" configuration which is stored in memory

yaml_config  YAML-based Configuration

Description
It tries to find YAML file in working directory looking for .dstack/config.yaml by default. If it’s failed it tries to use global setting in home directory in the same relative path.

Usage
yaml_config()

Value
A function that returns a list that contains user, token and server for specified profile.
Index

auto_handler, 2
commit, 2
configure, 3
create_frame, 4
dataframe_handler, 6
datatable_handler, 6
ggplot_handler, 7
in_place_config, 7
is.datatable, 8
is.ggplot2, 8
is.plotly, 8
json_protocol, 9
list_profiles, 9
plotly_handler, 9
pull, 10
push, 11
push_frame, 11
use_config, 13
yaml_config, 13