

# Package ‘SigOptR’

March 10, 2017

**Title** R API Wrapper for SigOpt

**Version** 0.0.1

**Author** Alexandra Johnson [aut, cre]

**Maintainer** Alexandra Johnson <alexandra@sigopt.com>

**Description** Interfaces with the 'SigOpt' API. More info at <<https://sigopt.com>>.

**Imports** http, jsonlite

**Depends** R (>= 3.2.0)

**License** MIT + file LICENSE

**LazyData** true

**RoxygenNote** 6.0.1.9000

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-03-10 08:50:09

## R topics documented:

create_experiment . . . . .	2
create_observation . . . . .	2
create_suggestion . . . . .	3
fetch_experiment . . . . .	4
franke . . . . .	5
sigopt_api_token . . . . .	5
sigopt_api_url . . . . .	6
sigopt_api_user_agent . . . . .	6
sigopt_auth . . . . .	7
sigopt_check . . . . .	7
sigopt_GET . . . . .	8
sigopt_parse . . . . .	8
sigopt_POST . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

create\_experiment      *Create an experiment*

---

**Description**

Create an experiment

**Usage**

```
create_experiment(body)
```

**Arguments**

body                    POST body of create request

**Value**

experiment created by SigOpt

**Examples**

```
env <- Sys.getenv("NOT_CRAN")
if (!identical(env, "true")) {
  0
} else {
  create_experiment(list(
    name="R test experiment",
    parameters=list(
      list(name="x1", type="double", bounds=list(min=0, max=100)),
      list(name="x2", type="double", bounds=list(min=0, max=100))
    )
  ))
}
```

---

create\_observation      *Create an observation for an experiment*

---

**Description**

Create an observation for an experiment

**Usage**

```
create_observation(experiment_id, body)
```

**Arguments**

experiment\_id    the id of an experiment to create an observation for  
body                    POST body of create request

**Value**

observation created by SigOpt

**Examples**

```
env <- Sys.getenv("NOT_CRAN")
if (!identical(env, "true")) {
  0
} else {
  experiment <- create_experiment(list(
    name="R test experiment",
    parameters=list(
      list(name="x1", type="double", bounds=list(min=0, max=100)),
      list(name="x2", type="double", bounds=list(min=0, max=100))
    )
  ))
  suggestion <- create_suggestion(experiment$id)
  create_observation(experiment$id, list(suggestion=suggestion$id, value=99.08))
  create_observation(experiment$id, list(suggestion=suggestion$id, value=99.58, value_stddev=0.1))}

```

---

<code>create_suggestion</code>	<i>Create a suggestion for an experiment</i>
--------------------------------	--

---

**Description**

Create a suggestion for an experiment

**Usage**

```
create_suggestion(experiment_id, body = NULL)
```

**Arguments**

`experiment_id` the id of an experiment to create an suggestion for  
`body` POST body of create request

**Value**

suggestion created by SigOpt

**Examples**

```
env <- Sys.getenv("NOT_CRAN")
if (!identical(env, "true")) {
  0
} else {
  experiment <- create_experiment(list(
    name="R test experiment",
    parameters=list(

```

```
      list(name="x1", type="double", bounds=list(min=0, max=100)),
      list(name="x2", type="double", bounds=list(min=0, max=100))
    )
  ))
  create_suggestion(experiment$id)}
```

---

fetch_experiment	<i>Fetch an experiment</i>
------------------	----------------------------

---

## Description

Fetch an experiment

## Usage

```
fetch_experiment(experiment_id, body = NULL)
```

## Arguments

experiment\_id the id of an experiment to fetch  
body Url params of GET request

## Value

SigOpt experiment with id experiment\_id

## Examples

```
env <- Sys.getenv("NOT_CRAN")
if (!identical(env, "true")) {
  0
} else {
  experiment <- create_experiment(list(
    name="R test experiment",
    parameters=list(
      list(name="x1", type="double", bounds=list(min=0, max=100)),
      list(name="x2", type="double", bounds=list(min=0, max=100))
    )
  ))
  fetch_experiment(experiment$id)}
```

---

franke	<i>Franke function - <a href="http://www.sfu.ca/~ssurjano/franke2d.html">http://www.sfu.ca/~ssurjano/franke2d.html</a></i>
--------	--

---

**Description**

Franke function - <http://www.sfu.ca/~ssurjano/franke2d.html>

**Usage**

```
franke(x, y)
```

**Arguments**

x	First dimension
y	Second dimension

**Value**

The franke function evaluated at x,y

**Examples**

```
franke(0,1)
```

---

sigopt_api_token	<i>Get the SigOpt API token from the SIGOPT_API_TOKEN environment variable or user input</i>
------------------	--

---

**Description**

Get the SigOpt API token from the SIGOPT\_API\_TOKEN environment variable or user input

**Usage**

```
sigopt_api_token(force = FALSE)
```

**Arguments**

force	force entry of SigOpt API token, even if present
-------	--

**Value**

SigOpt API token

sigopt\_api\_url      *Get the SigOpt API url from the SIGOPT\_API\_URL environment variable or use default Most users will be ok with the default value*

---

**Description**

Get the SigOpt API url from the SIGOPT\_API\_URL environment variable or use default Most users will be ok with the default value

**Usage**

```
sigopt_api_url()
```

**Value**

Base url for SigOpt API requests

**See Also**

[sigopt\\_GET](#) and [sigopt\\_POST](#), which perform the HTTP requests

---

sigopt\_api\_user\_agent      *User agent for current version of SigOpt R API Client*

---

**Description**

User agent for current version of SigOpt R API Client

**Usage**

```
sigopt_api_user_agent()
```

**Value**

User agent

**See Also**

[sigopt\\_GET](#) and [sigopt\\_POST](#), which perform the HTTP requests

---

sigopt_auth	<i>Create authentication for SigOpt API, using HTTP Basic Auth</i>
-------------	--

---

**Description**

Create authentication for SigOpt API, using HTTP Basic Auth

**Usage**

```
sigopt_auth(api_token = sigopt_api_token())
```

**Arguments**

api\_token      SigOpt API token

**Value**

http basic authentication with api\_token as username and no password

---

sigopt_check	<i>Check content returned by the SigOpt API</i>
--------------	---

---

**Description**

Check content returned by the SigOpt API

**Usage**

```
sigopt_check(req)
```

**Arguments**

req              result of request to the SigOpt API

**Value**

invisible(), stops if there is an error status code

**See Also**

[sigopt\\_GET](#) and [sigopt\\_POST](#), which call this function interally

---

sigopt_GET	<i>GET request to SigOpt API path, with json encoded body</i>
------------	---

---

**Description**

GET request to SigOpt API path, with json encoded body

**Usage**

```
sigopt_GET(path, query = NULL, api_token = sigopt_api_token())
```

**Arguments**

path	path of SigOpt API url to POST to
query	list of query parameters to be url-encoded
api_token	SigOpt api token

**Value**

result of request to SigOpt API, needs to be JSON decoded

**See Also**

[sigopt\\_parse](#), which parses the result of this function

---

---

sigopt_parse	<i>Parse content returned by the SigOpt API</i>
--------------	---

---

**Description**

Parse content returned by the SigOpt API

**Usage**

```
sigopt_parse(req)
```

**Arguments**

req	result of request to the SigOpt API
-----	-------------------------------------

**Value**

json decoding of request object

**See Also**

[sigopt\\_GET](#) and [sigopt\\_POST](#), which call this function interally



---

sigopt_POST	<i>POST request to SigOpt API path, with json encoded body</i>
-------------	--

---

**Description**

POST request to SigOpt API path, with json encoded body

**Usage**

```
sigopt_POST(path, body, api_token = sigopt_api_token())
```

**Arguments**

path	path of SigOpt API url to POST to
body	POST body, will be json-encoded
api_token	SigOpt api token

**Value**

result of request to SigOpt API, needs to be JSON decoded

**See Also**

[sigopt\\_parse](#), which parses the result of this function

# Index

`create_experiment`, [2](#)  
`create_observation`, [2](#)  
`create_suggestion`, [3](#)

`fetch_experiment`, [4](#)  
`franke`, [5](#)

`sigopt_api_token`, [5](#)  
`sigopt_api_url`, [6](#)  
`sigopt_api_user_agent`, [6](#)  
`sigopt_auth`, [7](#)  
`sigopt_check`, [7](#)  
`sigopt_GET`, [6–8](#), [8](#)  
`sigopt_parse`, [8](#), [8](#), [9](#)  
`sigopt_POST`, [6–8](#), [9](#)