Package ‘pins’

April 7, 2020

Type Package
Title Pin, Discover and Share Resources
Version 0.4.0
Maintainer Javier Luraschi <javier@rstudio.com>
Description Pin remote resources into a local cache to work offline, improve speed and avoid recomputing; discover and share resources in local folders, 'GitHub', 'Kaggle' or 'RStudio Connect'. Resources can be anything from 'CSV', 'JSON', or image files to arbitrary R objects.
License Apache License 2.0
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Depends R (>= 3.2.0)
Imports backports, base64enc, crayon, digest, filelock, httr, jsonlite, magrittr, mime, openssl, rappdirs, withr, yaml, zip
Suggests callr, covr, data.table, knitr, rmarkdown, R6, tibble, testthat, xml2
VignetteBuilder knitr
URL https://github.com/rstudio/pins
BugReports https://github.com/rstudio/pins/issues
NeedsCompilation no
Author Javier Luraschi [aut, cre], RStudio [cph]
Repository CRAN
Date/Publication 2020-04-07 18:50:02 UTC
R topics documented:

- board_cache_path ........................................... 2
- board_default .............................................. 3
- board_deregister ........................................... 3
- board_get ..................................................... 4
- board_list .................................................... 4
- board_pin_create .......................................... 5
- board_register .............................................. 5
- board_register_azure ....................................... 6
- board_register_datatxt .................................... 7
- board_register_dospace ..................................... 8
- board_register_gcloud .................................... 9
- board_register_github .................................... 10
- board_register_kaggle ................................... 11
- board_register_local ...................................... 12
- board_register_rsconnect ................................ 13
- board_register_s3 ......................................... 14
- pin ......................................................... 15
- pin_find .................................................... 16
- pin_get ...................................................... 17
- pin_info ...................................................... 18
- pin_reactive ............................................... 19
- pin_remove .................................................. 20
- pin_versions ............................................... 20

Index 22

---

**board_cache_path**

*Retrieve Default Cache Path*

**Description**

Retrieves the default path used to cache boards and pins. Makes use of the `rappdirs` package to use cache folders defined by each OS.

**Usage**

```r
board_cache_path()
```

**Examples**

```r
# retrieve default cache path
board_cache_path()
```
**board_default**  

**Default Board**

### Description

Retrieves the default board, which defaults to "local" but can also be configured with the `pins.board` option.

### Usage

```r
board_default()
```

### Examples

```r
library(pins)

# create temp board
board_register_local("temp", cache = tempfile())

# configure default board
options(pins.board = "temp")

# retrieve default board
board_default()

# revert default board
options(pins.board = NULL)
```

---

**board_deregister**  

**Deregister Board**

### Description

Deregisters a board, useful to disable boards no longer in use.

### Usage

```r
board_deregister(name, ...)
```

### Arguments

- `name`  
  An optional name to identify this board, defaults to the board name.

- `...`  
  Additional parameters required to deregister a particular board.
Examples

# create a new local board
board_register("local", "other_board", cache = tempfile())

# pin iris to new board
pin(iris, board = "other_board")

# deregister new board
board_deregister("other_board")

---

board_get

*Get Board*

**Description**

Retrieves information about a particular board.

**Usage**

```bash
board_get(name)
```

**Arguments**

- **name**
  - The name of the board to use

---

board_list

*List Boards*

**Description**

Retrieves all available boards.

**Usage**

```bash
board_list()
```
board_pin_create  Custom Boards

Description

Family of functions meant to be used to implement custom boards extensions, not to be used by users.

Usage

board_pin_create(board, path, name, metadata, ...)
board_initialize(board, ...)
board_pin_get(board, name, ...)
board_pin_remove(board, name, ...)
board_pin_find(board, text, ...)
board_browse(board, ...)

Arguments

board The board to extend, retrieved with board_get().
path The path to store as a pin.
name The name of the pin.
metadata A list of metadata associated with this pin.
... Additional parameters.
text The text pattern to find a pin.

board_register  Register Board

Description

Registras a board, useful to find resources with pin_find() or pin to additional boards with pin().

Usage

board_register(board, name = board, cache = board_cache_path(),
versions = NULL, ...)
Arguments

- board: The name of the board to register.
- name: An optional name to identify this board, defaults to the board name.
- cache: The local folder to use as a cache, defaults to `board_cache_path()`.
- versions: Should this board be registered with support for versions?
- ...: Additional parameters required to initialize a particular board.

Details

A board requires a local cache to avoid downloading files multiple times. It is recommended to not specify the cache parameter since it defaults to a well known `rappdirs`. However, you are welcome to specify any other location for this cache or even a temp folder with `tempfile()`. Notice that, when using a temp folder, pins will be cleared when your R session restarts. The cache parameter can be also set with the `pins.path` option.

The versions parameter default to different values depending on which board is choosed. For instance, local boards do not use versions by default, but GitHub boards do.

See Also

- `board_register_local`, `board_register_github`, `board_register_kaggle`, `board_register_rsconnect` and `board_register_datatxt`.

Examples

```r
# create a new local board
board_register("local", "other_board", cache = tempfile())

# create a Website board
board_register("datatxt",
              name = "txtexample",
              url = "https://datatxt.org/data.txt",
              cache = tempfile())
```

Description

Wrapper with explicit parameters over `board_register()` to register a Microsoft Azure Storage Blob as a board.
board_register_datatxt

Usage

```r
board_register_azure(name = "azure",
    container = Sys.getenv("AZURE_STORAGE_CONTAINER"),
    account = Sys.getenv("AZURE_STORAGE_ACCOUNT"),
    key = Sys.getenv("AZURE_STORAGE_KEY"),
    cache = board_cache_path(),
    ...
)
```

Arguments

- **name**: Optional name for this board, defaults to 'azure'.
- **container**: The name of the Azure Storage container. Defaults to the AZURE_STORAGE_CONTAINER environment variable.
- **account**: The account of the Azure Storage container. Defaults to the AZURE_STORAGE_ACCOUNT environment variable.
- **key**: The key of the Azure Storage container. Defaults to the AZURE_STORAGE_KEY environment variable.
- **cache**: The local folder to use as a cache, defaults to board_cache_path().
- **...**: Additional parameters required to initialize a particular board.

Details

This function requires an Azure Storage container to be manually created; otherwise, registering an Azure board will fail.

See Also

board_register

Examples

```r
## Not run:
# the following example requires an Azure Storage key
board_register_azure(container = "pinscontainer",
    account = "pinsstorage",
    key = "abcabcabcabcabcabcabcabcabcab==")
```

---

**board_register_datatxt**

*Register Data TXT Board*

Description

Wrapper with explicit parameters over `board_register()` to register as a board a website describing resources with a data.txt file.
Usage

`board_register_datatxt(url, name = NULL, headers = NULL, cache = board_cache_path(), ...)`

Arguments

- `url`: Path to the `data.txt` file or path containing it.
- `name`: The name for this board, usually the domain name of the website.
- `headers`: Optional list of headers to include or a function to generate them.
- `cache`: The local folder to use as a cache, defaults to `board_cache_path()`.
- `...`: Additional parameters required to initialize a particular board.

See Also

`board_register`

Examples

```r
# register website board using datatxt file
board_register_datatxt(url = "https://datatxt.org/data.txt", name = "txtexample", cache = tempfile())

# find pins
pin_find(board = "txtexample")
```

---

`board_register_dospace`

Register DigitalOcean Board

Description

Wrapper with explicit parameters over `board_register()` to register a DigitalOcean Spaces board.

Usage

```r
board_register_dospace(name = "dospace", space = Sys.getenv("DO_SPACE"), key = Sys.getenv("DO_ACCESS_KEY_ID"), secret = Sys.getenv("DO_SECRET_ACCESS_KEY"), datacenter = Sys.getenv("DO_DATACENTER"), cache = board_cache_path(), host = "digitaloceanspaces.com", ...)
```
Arguments

- **name**: Optional name for this board, defaults to 's3'.

- **space**: The name of the DigitalOcean space. Defaults to the DO_SPACE environment variable.

- **key**: The key of the DigitalOcean space. Defaults to the DO_ACCESS_KEY_ID environment variable.

- **secret**: The secret of the DigitalOcean space. Defaults to the DO_SECRET_ACCESS_KEY environment variable.

- **datacenter**: The datacenter of the DigitalOcean space. Defaults to the DO_DATACENTER environment variable.

- **cache**: The local folder to use as a cache, defaults to board_cache_path().

- **host**: The host to use for storage, defaults to "digitaloceanspaces.com".

- **...**: Additional parameters required to initialize a particular board.

Details

This function requires a DigitalOcean space to be manually created; otherwise, registering a DigitalOcean space will fail.

See Also

- board_register

Examples

```r
## Not run:
# the following example requires a DigitalOcean Spaces API key
board_register_s3(bucket = "s3bucket")

## End(Not run)
```

---

**board_register_gcloud**  
*Register Google Cloud Board*

Description

Wrapper with explicit parameters over board_register() to register a Google Cloud Storage container as a board.

Usage

```r
board_register_gcloud(name = "gcloud",
bucket = Sys.getenv("GCloud_STORAGE_BUCKET"),
token = NULL,
cache = board_cache_path(), ...)
```
Arguments

name  
Optional name for this board, defaults to ‘gcloud’.

bucket  
The name of the Google Cloud Storage bucket. Defaults to the GCLOUD_STORAGE_BUCKET environment variable.

token  
The access token of the Google Cloud Storage container. Defaults to use the Google Cloud SDK if configured.

cache  
The local folder to use as a cache, defaults to board_cache_path().

...  
Additional parameters required to initialize a particular board.

Details

This function requires a Google Cloud Storage container to be manually created; otherwise, registering a Google Cloud board will fail.

See Also

board_register

Examples

## Not run:

# the following example requires the Google Cloud SDK to be configured
board_register_gcloud(container = "gcloudcontainer")

## End(Not run)

board_register_github  
Register GitHub Board

Description

Wrapper with explicit parameters over board_register() to register a GitHub repo as a board.

Usage

board_register_github(name = "github", repo = NULL,
branch = "master", token = NULL, path = "",
host = "https://api.github.com", cache = board_cache_path(), ...)

Arguments

name  
Optional name for this board, defaults to ‘github’.

repo  
The GitHub repository formatted as ‘owner/repo’, can be NULL if the GITHUB_PAT environment variable is set.

branch  
The branch to use when committing pins.
board_register_kaggle

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Token to use when GITHUB_PAT is not specified.</td>
</tr>
<tr>
<td>path</td>
<td>The subdirectory in the repo where the pins will be stored.</td>
</tr>
<tr>
<td>host</td>
<td>The URL hosting the GitHub API, defaults to &quot;<a href="https://api.github.com">https://api.github.com</a>&quot;.</td>
</tr>
<tr>
<td>cache</td>
<td>The local folder to use as a cache, defaults to board_cache_path().</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters required to initialize a particular board.</td>
</tr>
</tbody>
</table>

details

This function requires a GitHub repo to be manually created; otherwise, registering a GitHub board will fail.

When a file upload exceeds 25MB, a GitHub release file will be used since they support up to 2GB file uploads. This threshold can be configured through the pins.github.release option which is specified in megabytes and defaults to 25.

When using GitHub Enterprise, consider customizing the host parameter to "https://yourhostname/api/v3".

see also

board_register

examples

## Not run:
# the following example requires a GitHub API key
board_register_github(repo = "owner/repo")

## End(Not run)

---

**board_register_kaggle**  
*Register Kaggle Board*

description

Wrapper with explicit parameters over board_register() to register Kaggle as a board.

usage

```r
board_register_kaggle(name = "kaggle", token = NULL, overwrite = FALSE, cache = board_cache_path(), ...)
```

arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Optional name for this board, defaults to 'kaggle'.</td>
</tr>
<tr>
<td>token</td>
<td>The Kaggle token as a path to the kaggle.json file, can be NULL if the ~/.kaggle/kaggle.json file already exists.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Should ~/.kaggle/kaggle.json be overridden?</td>
</tr>
<tr>
<td>cache</td>
<td>The local folder to use as a cache, defaults to board_cache_path().</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters required to initialize a particular board.</td>
</tr>
</tbody>
</table>
See Also

board_register

Examples

```
## Not run:
# the following example requires a Kaggle API token
board_register_kaggle(token = "path/to/kaggle.json")

## End(Not run)
```

```
board_register_local  Register Local Board

Description

Wrapper with explicit parameters over board_register() to register a local folder as a board.

Usage

board_register_local(name = "local", cache = board_cache_path(), ...)

Arguments

name  Optional name for this board, defaults to 'local'.
cache  The local folder to use as a cache, defaults to board_cache_path().
...  Additional parameters required to initialize a particular board.

See Also

board_register

Examples

```
# register local board using a temp folder
board_register_local(cache = tempfile())
```
function {board_register_rsconnect}

Description

Wrapper with explicit parameters over `board_register()` to register RStudio Connect as a board.

Usage

```r
board_register_rsconnect(name = "rsconnect", server = NULL,
                         account = NULL, key = NULL, output_files = FALSE,
                         cache = board_cache_path(), ...)
```

Arguments

- `name`: Optional name for this board, defaults to 'rsconnect'.
- `server`: Optional address to RStudio Connect server.
- `account`: Optional account name to use with RStudio Connect.
- `key`: The RStudio Connect API key.
- `output_files`: Should the output in an automated report create output files?
- `cache`: The local folder to use as a cache, defaults to `board_cache_path()`.
- `...`: Additional parameters required to initialize a particular board.

See Also

`board_register`

Examples

```r
## Not run:
# the following examples require an RStudio Connect API key

# register from rstudio
board_register_rsconnect()

# register from rstudio with multiple servers
board_register_rsconnect(server = "https://rstudio-connect-server")

# register from rstudio with multiple account
board_register_rsconnect(account = "account-name")

# register automated report for rstudio connect
board_register_rsconnect(key = Sys.getenv("CONNECT_API_KEY"),
                         server = Sys.getenv("CONNECT_SERVER"))
```

## End(Not run)


**board_register_s3**  
*Register S3 Board*

**Description**

Wrapper with explicit parameters over board_register() to register an Amazon S3 bucket as a board.

**Usage**

```r
board_register_s3(name = "s3", bucket = Sys.getenv("AWS_BUCKET"),  
key = Sys.getenv("AWS_ACCESS_KEY_ID"),  
secret = Sys.getenv("AWS_SECRET_ACCESS_KEY"),  
cache = board_cache_path(), host = "s3.amazonaws.com", ...)
```

**Arguments**

- **name**: Optional name for this board, defaults to 's3'.
- **bucket**: The name of the Amazon S3 bucket. Defaults to the AWS_BUCKET environment variable.
- **key**: The key of the Amazon S3 bucket. Defaults to the AWS_ACCESS_KEY_ID environment variable.
- **secret**: The secret of the Amazon S3 bucket. Defaults to the AWS_SECRET_ACCESS_KEY environment variable.
- **cache**: The local folder to use as a cache, defaults to board_cache_path().
- **host**: The host to use for storage, defaults to "s3.amazonaws.com".
- **...**: Additional parameters required to initialize a particular board.

**Details**

This function requires an Amazon S3 bucket to be manually created; otherwise, registering an S3 board will fail.

**See Also**

board_register

**Examples**

```r
## Not run:
# the following example requires an Amazon S3 API key
board_register_s3(bucket = "s3bucket")
```

## End(Not run)
Description

Pins the given resource locally or to the given board.

Usage

```r
pin(x, name = NULL, description = NULL, board = NULL, ...)  
```

Arguments

- `x`: An object, local file or remote URL to pin.
- `name`: The name for the dataset or object.
- `description`: Optional description for this pin.
- `board`: The board where this pin will be placed.
- `...`: Additional parameters.

Details

`pin()` allows you to cache remote resources and intermediate results with ease. When caching remote resources, usually URLs, it will check for HTTP caching headers to avoid re-downloading when the remote result has not changed.

This makes it ideal to support reproducible research by requiring manual instruction to download resources before running your R script.

In addition, `pin()` still works when working offline or when the remote resource becomes unavailable; when this happens, a warning will be triggered but your code will continue to work.

Examples

```r
library(pins)

# define local board
board_register_local(cache = tempfile())

# cache the mtcars dataset
pin(mtcars)

# cache computation over mtcars
mtcars[mtcars$mpg > 30,] %>%
  pin(name = "mtefficient")

# retrieve cached pin
pin_get("mtefficient")

# url to remote resource
```
Find Pin

Find a pin in any board registered using board_register().

Usage

pin_find(text = NULL, board = NULL, name = NULL, extended = FALSE, ...)

Arguments

text The text to find in the pin description or name.
board The board name used to find the pin.
name The exact name of the pin to match when searching.
extended Should additional board-specific columns be shown?
... Additional parameters.

Details

pin_find() allows you to discover new resources or retrieve pins you’ve previously created with pin().

The pins package comes with a CRAN packages board which allows searching all CRAN packages; however, you can add additional boards to search from like Kaggle, Github and RStudio Connect.

For 'local' and 'packages' boards, the 'text' parameter searches the title and description of a pin using a regular expression. Other boards search in different ways, most of them are just partial matches, please refer to their documentation to understand how other boards search for pins.

Once you find a pin, you can retrieve with pin_get("pin-name").
Examples

```r
library(pins)

# retrieve pins
pin_find()

# search pins related to 'cars'
pin_find("cars")

# search pins related to 'seattle' in the 'packages' board
pin_find("seattle", board = "packages")

# search pins related to 'london' in the 'packages' board
pin_find("london", board = "packages")

# retrieve 'hpiR/seattle_sales' pin
pin_get("hpiR/seattle_sales")

# retrieve 'bsamGP/London.Mortality' pin
pin_get("bsamGP/London.Mortality")
```

---

**pin_get**

*Retrieve Pin*

**Description**

Retrieves a pin by name from the local or given board.

**Usage**

```r
pin_get(name, board = NULL, cache = TRUE, extract = NULL,
version = NULL, files = FALSE, signature = NULL, ...)
```

**Arguments**

- **name**: The name of the pin.
- **board**: The board where this pin will be retrieved from.
- **cache**: Should the pin cache be used? Defaults to TRUE.
- **extract**: Should compressed files be extracted? Each board defines the default behavior.
- **version**: The version of the dataset to retrieve, defaults to latest one.
- **files**: Should only the file names be returned?
- **signature**: Optional signature to validate this pin, use `pin_info()` to compute signature.
- **...**: Additional parameters.
Details

`pin_get()` retrieves a pin by name and, by default, from the local board. You can use the `board` parameter to specify which board to retrieve a pin from. If a board is not specified, it will use `pin_find()` to find the pin across all boards and retrieve the one that matches by name.

Examples

```r
library(pins)

# define local board
board_register_local(cache = tempfile())

# cache the mtcars dataset
pin(mtcars)

# retrieve the mtcars pin
pin_get("mtcars")

# retrieve mtcars pin from packages board
pin_get("easyalluvial/mtcars2", board = "packages")
```

---

**pin_info**

*Pin Info*

**Description**

Retrieve information for a given pin.

**Usage**

```r
pin_info(name, board = NULL, extended = TRUE, metadata = TRUE, signature = FALSE, ...)
```

**Arguments**

- `name`: The exact name of the pin to match when searching.
- `board`: The board name used to find the pin.
- `extended`: Should additional board-specific information be shown?
- `metadata`: Should additional pin-specific information be shown?
- `signature`: Should a signature to identify this pin be shown?
- `...`: Additional parameters.
Examples

```r
library(pins)

# define local board
grep_register_local(cache = tempfile())

# cache the mtcars dataset
pin(mtcars)

# print pin information
pin_info("mtcars")
```

---

**pin_reactive**  
*Reactive Pin*

**Description**

Creates a pin that reacts to changes in the given board by polling `pin_get()`, useful when used from the `shiny` package.

**Usage**

```r
pin_reactive(name, board, interval = 5000, session = NULL, extract = NULL)
```

**Arguments**

- **name**: The name of the pin.
- **board**: The board where this pin will be retrieved from.
- **interval**: Approximate number of milliseconds to wait to retrieve updated pin. This can be a numeric value, or a function that returns a numeric value.
- **session**: The user session to associate this file reader with, or NULL if none. If non-null, the reader will automatically stop when the session ends.
- **extract**: Should compressed files be extracted? Each board defines the default behavior.
**Description**

Unpins the given named pin from the given board.

**Usage**

```
pin_remove(name, board)
```

**Arguments**

- **name**: The name for the pin.
- **board**: The board from where this pin will be removed.

**Details**

Notice that some boards do not support deleting pins, this is the case for the Kaggle board. For these boards, you would manually have to remove resources using the tools the board provides.

**Examples**

```r
library(pins)

# define local board
board_register_local(cache = tempfile())

# create mtcars pin
pin(mtcars)

# remove mtcars pin
pin_remove("mtcars", board = "local")
```

---

**Description**

Retrieve versions available for a given pin.

**Usage**

```
pin_versions(name, board = NULL, full = FALSE, ...)
```
Arguments

name  The exact name of the pin to match when searching.
board  The board name used to find the pin.
full  Should the full versioned paths be shown? Defaults to FALSE.
...  Additional parameters.

Examples

library(pins)

# define local board with versioning enabled
board_register_local(cache = tempfile(), versions = TRUE)

# cache the mtcars dataset
pin(mtcars, name = "mtcars")

# cache variation of the mtcars dataset
pin(mtcars * 10, name = "mtcars")

# print the mtcars versions
versions <- pin_versions("mtcars") %>% print()

# retrieve the original version
pin_get("mtcars", version = versions$version[1])

# retrieve the variation version
pin_get("mtcars", version = versions$version[2])
Index

board_browse (board_pin_create), 5
board_cache_path, 2
board_default, 3
board_deregister, 3
board_get, 4
board_initialize (board_pin_create), 5
board_list, 4
board_pin_create, 5
board_pin_find (board_pin_create), 5
board_pin_get (board_pin_create), 5
board_pin_remove (board_pin_create), 5
board_register, 5
board_register_azure, 6
board_register_datatxt, 6, 7
board_register_dospace, 8
board_register_gcloud, 9
board_register_github, 6, 10
board_register_kaggle, 6, 11
board_register_local, 6, 12
board_register_rsconnect, 6, 13
board_register_s3, 14
pin, 15
pin_find, 16
pin_get, 17
pin_info, 18
pin_reactive, 19
pin_remove, 20
pin_versions, 20