Package 'AnVILBase'

July 15, 2025

Title Generic functions for interacting with the AnVIL ecosystem

Version 1.3.1

Description Provides generic functions for interacting with the AnVIL ecosystem. Packages that use either GCP or Azure in AnVIL are built on top of AnVILBase. Extension packages will provide methods for interacting with other cloud providers.

License Artistic-2.0

Depends R (>= 4.5.0)

Imports httr, httr2, dplyr, jsonlite, methods, tibble

Suggests AnVIL, AnVILAz, AnVILGCP, BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), tinytest

biocViews Software, Infrastructure

URL https://github.com/Bioconductor/AnVILBase

BugReports https://github.com/Bioconductor/AnVILBase/issues

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Date 2025-04-28

Config/testthat/edition 3

Collate 'AnVILBase-generics.R' 'AnVILBase-methods.R' 'AnVILBase-package.R' 'Platform-class.R' 'Response.R' 'avnotebooks-generics.R' 'avnotebooks-methods.R' 'avtable-generics.R' 'avtable-methods.R' 'avworkflow-generics.R' 'avworkflow-methods.R' 'avworkspace-generics.R' 'avworkspace-methods.R' 'cloud_platform.R' 'has_avworkspace-generics.R' 'has_avworkspace-methods.R' 'utilities.R'

git_url https://git.bioconductor.org/packages/AnVILBase

git_branch devel

git_last_commit aa91db5

git_last_commit_date 2025-04-28

Repository Bioconductor 3.22

Date/Publication 2025-07-15

Author Marcel Ramos [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3242-0582>),
Martin Morgan [aut, ctb] (ORCID:
<https://orcid.org/0000-0002-5874-8148>),
NIH NHGRI U24HG004059 [fnd]

Maintainer Marcel Ramos <marcel.ramos@sph.cuny.edu>

Contents

`AnVILBase-package` *The AnVILBase package*

Description

The `AnVILBase` package defines S4 generics for the AnVIL package.

Author(s)

Maintainer: Marcel Ramos <marcel.ramos@sph.cuny.edu> (ORCID)

Authors:

- Martin Morgan (ORCID) [contributor]

Other contributors:

- NIH NHGRI U24HG004059 [funder]

See Also

Useful links:

- https://github.com/Bioconductor/AnVILBase
- Report bugs at https://github.com/Bioconductor/AnVILBase/issues

---

`AnVILBase-generics` *All the generics for AnVIL*

Description

These are the generics to be used by the AnVIL package.

Usage

```
avcopy(..., platform)

avlist(..., platform)

avremove(..., platform)

avbackup(..., platform)

avrestore(..., platform)

avstorage(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

called for the side effect of registering generic functions

See Also

avworkspace-generics, avtable-generics, avworkflow-generics

Examples

```
getGeneric("avcopy")
```

AnVILBase-methods *The AnVILBase methods for 'missing' and 'ANY' signatures*

Description

The AnVILBase package defines S4 methods for the AnVIL package. These methods are defined for the 'missing' and 'ANY' signatures.

Usage

```
## S4 method for signature 'missing'
avcopy(..., platform)

## S4 method for signature 'ANY'
avcopy(..., platform)

## S4 method for signature 'missing'
avlist(..., platform)

## S4 method for signature 'ANY'
avlist(..., platform)

## S4 method for signature 'missing'
avremove(..., platform)

## S4 method for signature 'ANY'
avremove(..., platform)

## S4 method for signature 'missing'
avbackup(..., platform)

## S4 method for signature 'ANY'
avbackup(..., platform)

## S4 method for signature 'missing'
avrestore(..., platform)

## S4 method for signature 'ANY'
avrestore(..., platform)

## S4 method for signature 'missing'
avstorage(..., platform)

## S4 method for signature 'ANY'
avstorage(..., platform)
```

Arguments

...	Arguments passed to the methods.
platform	A Platform derived class indicating the AnVIL environment, currently, azure and gcp classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- `avcopy(missing)`: Copy a file from the cloud
- `avcopy(ANY)`: Copy a file from the cloud
- `avlist(missing)`: List files in the cloud
- `avlist(ANY)`: List files in the cloud
- `avremove(missing)`: Remove a file from the cloud
- `avremove(ANY)`: Remove a file from the cloud
- `avbackup(missing)`: Backup a file to the cloud
- `avbackup(ANY)`: Backup a file to the cloud
- `avrestore(missing)`: Restore a file from the cloud
- `avrestore(ANY)`: Restore a file from the cloud
- `avstorage(missing)`: Get the storage location
- `avstorage(ANY)`: Get the storage location

Examples

```
findMethods("avcopy")
getMethod("avcopy", c(platform = "missing"))
```

---

avbase-utilities *Helper functions for working in AnVIL*

Description

- `avstop_for_status` - Check HTTP status code and raise error when less than 400.
- `avworkspaces_clean` - Clean workspace information

Usage

```
avstop_for_status(response, op)

avworkspaces_clean(.data)
```

Arguments

`response`	Response object from httr
`op`	Operation that was attempted
`.data`	A tibble with workspace information

Value

`avstop_for_status` - `response` if status code less than 400 otherwise throw an error

`avworkspaces_clean` - A cleaned tibble with workspace information

Examples

```
if (interactive()) {
    test <- httr::GET("http://google.com/")
    avstop_for_status(test, "google")
}
```

---

avnotebooks-generics *All the avnotebooks generics for AnVIL*

Description

These are the generics to be used by the AnVIL package.

Usage

```
avnotebooks(..., platform)

avnotebooks_localize(..., platform)

avnotebooks_delocalize(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

called for the side effect of registering generic functions

See Also

AnVILBase-generics, avworkspace-generics, avtable-generics, avworkflow-generics

Examples

```
getGeneric("avnotebooks")
```

avnotebooks-methods *All the notebook type methods for AnVIL*

Description

These are the methods to be used by the AnVIL package.

Usage

```
## S4 method for signature 'missing'
avnotebooks(..., platform)

## S4 method for signature 'ANY'
avnotebooks(..., platform)

## S4 method for signature 'missing'
avnotebooks_localize(..., platform)

## S4 method for signature 'ANY'
avnotebooks_localize(..., platform)

## S4 method for signature 'missing'
avnotebooks_delocalize(..., platform)

## S4 method for signature 'ANY'
avnotebooks_delocalize(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- `avnotebooks(missing)`: List the available notebooks in the current workspace
- `avnotebooks(ANY)`: List the available notebooks in the current workspace
- `avnotebooks_localize(missing)`: sync the contents of the workspace storage to the local file system
- `avnotebooks_localize(ANY)`: sync the contents of the workspace storage to the local file system
- `avnotebooks_delocalize(missing)`: sync the contents of the local file system to the workspace storage
- `avnotebooks_delocalize(ANY)`: sync the contents of the local file system to the workspace storage

Examples

```
findMethods("avnotebooks")
getMethod("avnotebooks", c(platform = "missing"))
```

`avtable-generics` *All the table type generics for AnVIL*

Description

These are the generics to be used by the AnVIL package.

Usage

```
avtable(..., platform)

avtables(..., platform)

avtable_import(..., platform)

avtable_import_set(..., platform)

avtable_delete(..., platform)

avtable_delete_values(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

called for the side effect of registering generic functions

See Also

AnVILBase-generics, avworkspace-generics, avworkflow-generics

Examples

```
getGeneric("avtable")
```

avtable-methods *The avtable methods for 'missing' and 'ANY' signatures*

Description

The AnVILBase package defines S4 methods for the AnVIL package. These methods are defined for the 'missing' and 'ANY' signatures.

Usage

```
## S4 method for signature 'missing'
avtable(..., platform)

## S4 method for signature 'ANY'
avtable(..., platform)

## S4 method for signature 'missing'
avtables(..., platform)

## S4 method for signature 'ANY'
avtables(..., platform)

## S4 method for signature 'missing'
avtable_import(..., platform)

## S4 method for signature 'ANY'
avtable_import(..., platform)

## S4 method for signature 'missing'
avtable_import_set(..., platform)

## S4 method for signature 'ANY'
avtable_import_set(..., platform)

## S4 method for signature 'missing'
avtable_delete(..., platform)

## S4 method for signature 'ANY'
avtable_delete(..., platform)

## S4 method for signature 'missing'
avtable_delete_values(..., platform)

## S4 method for signature 'ANY'
avtable_delete_values(..., platform)
```

Arguments

...	Arguments passed to the methods.
platform	A Platform derived class indicating the AnVIL environment, currently, azure and gcp classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- `avtable(missing)`: Show the contents of a particular table (entity / type) within a workspace.
- `avtable(ANY)`: Show the contents of a particular table (entity / type) within a workspace.
- `avtables(missing)`: List entities / types accross or within the current workspace respectively.
- `avtables(ANY)`: List entities / types accross or within the current workspace respectively.
- `avtable_import(missing)`: Upload a dataset to the DATA tab
- `avtable_import(ANY)`: Upload a dataset to the DATA tab
- `avtable_import_set(missing)`: Creates a grouping table for each distinct value in the column identified by `set`.
- `avtable_import_set(ANY)`: Creates a grouping table for each distinct value in the column identified by `set`.
- `avtable_delete(missing)`: Delete a table
- `avtable_delete(ANY)`: Delete a table
- `avtable_delete_values(missing)`: Delete a row in a table based on the key
- `avtable_delete_values(ANY)`: Delete a row in a table based on the key

Examples

```
findMethods("avtable")
getMethod("avtable", c(platform = "missing"))
```

`avworkflow-generics` *All the workflow type generics for AnVIL*

Description

These are the generics to be used by the AnVIL package.

Usage

```
avworkflow_jobs(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

called for the side effect of registering generic functions

See Also

AnVILBase-generics, avworkspace-generics, avtable-generics

Examples

```
getGeneric("avworkflow_jobs")
```

`avworkflow-methods` *All the workflow type methods for AnVIL*

Description

These are the methods to be used by the AnVIL package.

Usage

```
## S4 method for signature 'missing'
avworkflow_jobs(..., platform)

## S4 method for signature 'ANY'
avworkflow_jobs(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- `avworkflow_jobs(missing)`: Get the status of a workflow execution
- `avworkflow_jobs(ANY)`: Get the status of a workflow execution

Examples

```
findMethods("avworkflow_jobs")
getMethod("avworkflow_jobs", c(platform = "missing"))
```

avworkspace-generics *All the table type generics for AnVIL*

Description

These are the generics to be used by the AnVIL package.

Usage

```
avworkspace(..., platform)

avworkspaces(..., platform)

avworkspace_name(..., platform)

avworkspace_namespace(..., platform)

avworkspace_clone(..., platform)
```

Arguments

...	Arguments passed to the methods.
platform	A Platform derived class indicating the AnVIL environment, currently, azure and gcp classes are compatible.

Value

called for the side effect of registering generic functions

See Also

AnVILBase-generics, avtable-generics, avworkflow-generics

Examples

```
getGeneric("avworkspace")
```

avworkspace-methods *The* avworkspace *methods for 'missing' and 'ANY' signatures*

Description

The AnVILBase package defines S4 methods for the AnVIL package. These methods are defined for the 'missing' and 'ANY' signatures.

Usage

```
## S4 method for signature 'missing'
avworkspace(..., platform)

## S4 method for signature 'ANY'
avworkspace(..., platform)

## S4 method for signature 'missing'
avworkspaces(..., platform)

## S4 method for signature 'ANY'
avworkspaces(..., platform)

## S4 method for signature 'missing'
avworkspace_name(..., platform)

## S4 method for signature 'ANY'
avworkspace_name(..., platform)

## S4 method for signature 'missing'
avworkspace_namespace(..., platform)

## S4 method for signature 'ANY'
avworkspace_namespace(..., platform)

## S4 method for signature 'missing'
avworkspace_clone(..., platform)

## S4 method for signature 'ANY'
avworkspace_clone(..., platform)
```

Arguments

`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- `avworkspace(missing)`: Get the current workspace namespace/name
- `avworkspace(ANY)`: Get the current workspace namespace/name
- `avworkspaces(missing)`: List workspaces
- `avworkspaces(ANY)`: List workspaces
- `avworkspace_name(missing)`: Get the name of the current workspace
- `avworkspace_name(ANY)`: Get the name of the current workspace
- `avworkspace_namespace(missing)`: Get the namespace of the current workspace

- avworkspace_namespace(ANY): Get the namespace of the current workspace
- avworkspace_clone(missing): Clone the current workspace
- avworkspace_clone(ANY): Clone the current workspace

Examples

```
findMethods("avworkspace")
getMethod("avworkspace", c(platform = "missing"))
```

---

cloud_platform *Cloud Platform Identifier*

Description

cloud_platform calls the appropriate class constructor based on environment variables or options within the workspace. This function is used to determine the cloud platform to dispatch on for AnVIL methods. It returns an error when neither the Azure or Google Cloud environment variables are set. The avplatform_namespace function is a lower level helper to identify the platform based on environment variables or options. Generally, these functions are *not* meant to be used directly.

Usage

```
avplatform_namespace(default = "")

cloud_platform()
```

Arguments

default	character(1). The default cloud platform to use if no environment variables or options are set. The default is "".

Details

When GOOGLE_PROJECT is set, the function returns an object of class gcp. When WORKSPACE_ID is set, the function returns an object of class azure. Otherwise, the function returns an error. The user may also add options to set a default cloud platform. For AnVIL instances running on Google Cloud, the user can set options(GCLOUD_SDK_PATH = "/home/user/google-cloud-sdk") to set the default cloud platform to gcp. For AnVIL instances running on Azure, the user can set options(AnVILAz.workspace = "myworkspace") to set the default cloud platform to azure. Note that the values provided are example values and should be replaced with verifiable values.

Value

For avplatform_namespace: A character string indicating the cloud platform derived from environment variables.

For cloud_platform: An instance of class gcp or azure based on environment variables or options set within the AnVIL workspace. For avplatform_namespace: A character string indicating the cloud platform.

Examples

```
avplatform_namespace()
if (interactive()) {
    cloud_platform()
}
```

`has_avworkspace-generics`

The generic for checking if the user is within an environment

---

Description

These are generics to be used by the AnVILGCP and AnVILAz packages.

Usage

```
has_avworkspace(strict = FALSE, ..., platform)
```

Arguments

`strict`	`logical(1)` Whether to include a check for an existing `avworkspace_name()` setting. Default `FALSE`.
`...`	Arguments passed to the methods.
`platform`	A Platform derived class indicating the AnVIL environment, currently, `azure` and `gcp` classes are compatible.

Value

called for the side effect testing the workspace environment

See Also

AnVILBase-generics, avworkspace-generics, avtable-generics

Examples

```
getGeneric("has_avworkspace")
```

has_avworkspace-methods *The* has_avworkspace *methods for* missing *and* ANY *signatures.*

Description

The AnVILBase package defines S4 methods for the AnVIL package. These methods are defined for the 'missing' and 'ANY' signatures.

Usage

```
## S4 method for signature 'missing'
has_avworkspace(strict = FALSE, ..., platform)

## S4 method for signature 'ANY'
has_avworkspace(strict = FALSE, ..., platform)
```

Arguments

strict	logical(1) Whether to include a check for an existing avworkspace_name() setting. Default FALSE.
...	Arguments passed to the methods.
platform	A Platform derived class indicating the AnVIL environment, currently, azure and gcp classes are compatible.

Value

Methods for the 'missing' and 'ANY' signatures provide a way to redirect calls to the appropriate method for the current cloud platform.

Functions

- has_avworkspace(missing): Check whether the current workspace is using either Azure or Google infrastructure.
- has_avworkspace(ANY): Check whether the current workspace is using either Azure or Google infrastructure.

Examples

```
findMethods("has_avworkspace")
getMethod("has_avworkspace", c(platform = "missing"))
```

Platform-class *Virtual Class for Platform*

Description

This is a virtual class for Platform objects. It is used to define the class hierarchy for Platform objects.

Value

Used mainly for inheritance.

Response *Process service responses to tibble and other data structures.*

Description

Process service responses to tibble and other data structures.

Usage

```
flatten(x)

## S4 method for signature 'response'
flatten(x)

## S4 method for signature 'response'
str(object)

## S3 method for class 'response'
as.list(x, ..., as = c("text", "raw", "parsed"))

## S4 method for signature 'httr2_response'
flatten(x)
```

Arguments

x	A response object returned by the service.
object	A response object returned by the service.
...	not currently used
as	character(1); one of 'raw', 'text', 'parsed'

Value

flatten() returns a tibble where each row corresponds to a top-level list element of the return value, and columns are the unlisted second and more-nested elements.

str() displays a compact representation of the list-like JSON response; it returns NULL.

as.list() returns the content of the web service request as a list.

Functions

- `flatten(response)`: Reduce httr response object to a two-dimensional table
- `str(response)`: Create a compact representation of the list-like JSON response
- `flatten(httr2_response)`: Reduce httr2 request responses to two-dimensional table form

Author(s)

M. Morgan, M. Ramos

Examples

```
if (AnVILGCP::gcloud_exists() && interactive()) {
    library(AnVIL)
    leonardo <- Leonardo()
    leonardo$listRuntimes() |> flatten()

    leonardo$getSystemStatus() |> str()

    leonardo$getSystemStatus() |> as.list()
}
```

Index