Package 'AnVILAz'

November 11, 2026

Title R / Bioconductor Support for the AnVIL Azure Platform

Version 1.6.0

Date 2026-04-24

Description The AnVIL is a cloud computing resource developed in part by the National Human Genome Research Institute. The AnVILAz package supports end-users and developers using the AnVIL platform in the Azure cloud. The package provides a programmatic interface to AnVIL resources, including workspaces, notebooks, tables, and workflows. The package also provides utilities for managing resources, including copying files to and from Azure Blob Storage, and creating shared access signatures (SAS) for secure access to Azure resources.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.5.0)

Imports AnVILBase, BiocBaseUtils, curl, httr2, jsonlite, methods, rjsoncons, tibble, utils

biocViews Software, Infrastructure, ThirdPartyClient

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests BiocStyle, dplyr, knitr, readr, rmarkdown, tinytest

SystemRequirements az, azcopy

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

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

VignetteBuilder knitr

Collate 'AnVILAz-package.R' 'authentication.R' 'azure-class.R' 'avnotebooks-methods.R' 'avtable-methods.R' 'avworkflow-methods.R' 'avworkspace-methods.R' 'az-utilities.R' 'az_copy-helpers.R' 'az_sas_token.R' 'azure-methods.R' 'has_avworkspace-methods.R' 'resources.R' 'utilities.R' 'workspace-dev-ops.R' 'workspace_env.R' 'zzz.R'

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

git_branch RELEASE_3_23

git_last_commit f3d168f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

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

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

Contents

AnVILAz-package 2
avnotebooks-methods 3
avtable-methods 4
avworkflow-methods 7
avworkspace-methods 8
az-utilities 9
azure-class 10
azure-methods 10
az_copy-helpers 12
az_sas_token 14
az_token 15
has_avworkspace-methods 15
workspace-dev-ops 16
workspace-env 18

Index **20**

`AnVILAz-package` *AnVILAz: R / Bioconductor Support for the AnVIL Azure Platform*

Description

The AnVIL is a cloud computing resource developed in part by the National Human Genome Research Institute. The AnVILAz package supports end-users and developers using the AnVIL platform in the Azure cloud. The package provides a programmatic interface to AnVIL resources, including workspaces, notebooks, tables, and workflows. The package also provides utilities for managing resources, including copying files to and from Azure Blob Storage, and creating shared access signatures (SAS) for secure access to Azure resources.

Author(s)

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

Authors:

- Martin Morgan <mtmorgan.bioc@gmail.com> (ORCID) [contributor]

See Also

Useful links:

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

---

avnotebooks-methods *Azure Notebook Management*

Description

`avnotebooks()` lists the notebooks in the current workspace.

Usage

```
## S4 method for signature 'azure'
avnotebooks(local = FALSE, ..., platform = cloud_platform())

## S4 method for signature 'azure'
avnotebooks_localize(
  destination = "./analyses",
  dry = TRUE,
  ...,
  platform = cloud_platform()
)

## S4 method for signature 'azure'
avnotebooks_delocalize(
  source = "./",
  dry = TRUE,
  ...,
  platform = cloud_platform()
)
```

Arguments

`local`	`logical(1)` notebooks located on the workspace (`local = FALSE`, default) or runtime / local instance (`local = TRUE`). When `local = TRUE`, the notebook path is `<workspace_data_service_url()>/analyses`.
`...`	Additional arguments passed to lower level functions (not used)

platform	azure() The cloud platform class to dispatch on as given by AnVILBase::cloud_platform. Typically not set manually as cloud_platform() returns the "azure" class for Azure workspaces on AnVIL.
destination	character(1) or missing file path to the local file system directory for synchronization. The default location is ~/<workspace_data_service_url()>/analyses. Out-of-date local files are replaced with the workspace version.
dry	logical(1) Whether to perform a dry run i.e., add the --dry-run flag to the command.
source	character(1) or missing file path to the local file system directory for synchronization. The default location is the home folder. Out-of-date local files are replaced with the workspace version.

Value

avnotebooks() returns a character vector of files located in the workspace 'analyses/' folder path, or on the local file system.

Functions

- avnotebooks(azure): List the notebooks in the current workspace
- avnotebooks_localize(azure): Sync notebooks between the Azure Blob Storage Container and the local runtime
- avnotebooks_delocalize(azure): Sync notebooks between the local runtime and the Azure Blob Storage Container

Examples

```
if (has_avworkspace(strict = TRUE, platform = azure()))
    avnotebooks()
```

---

avtable-methods *AnVIL Azure table ("type") methods*

Description

Methods for working with AnVIL Azure tables. These are referred to as "types" in the AnVIL Workspace Data Service (WDS) API.

Usage

```
## S4 method for signature 'azure'
avtable(table, ..., platform = cloud_platform())

## S4 method for signature 'azure'
avtables(..., platform = cloud_platform())
```

```
## S4 method for signature 'azure'
avtable_import(
  .data,
  table,
  entity = names(.data)[[1L]],
  ...,
  platform = cloud_platform()
)

## S4 method for signature 'azure'
avtable_import_set(
  .data,
  origin,
  set = names(.data)[[1]],
  member = names(.data)[[2]],
  ...,
  platform = cloud_platform()
)

## S4 method for signature 'azure'
avtable_delete(table, ..., platform = cloud_platform())

## S4 method for signature 'azure'
avtable_delete_values(table, values, ..., platform = cloud_platform())
```

Arguments

table	character(1) The name of the table / type
...	Additional arguments passed to lower level functions (not used)
platform	azure() The cloud platform class to dispatch on as given by AnVILBase::cloud_platform. Typically not set manually as cloud_platform() returns the "azure" class for Azure workspaces on AnVIL.
.data	tibble() The dataset chiefly from the avtable() operation
entity	The entity name, i.e., the name of the column in the table that provides the keys for the data (a.k.a. primaryKey). By default, the first column in the table. The keys cannot contain special characters or spaces.
origin	character(1) name of the type (entity table) used to create the set e.g "sample", "participant", etc.
set	character(1) column name of .data identifying the set(s) to be created, i.e., the grouping variable.
member	character(1) column name of .data identifying the member(s) of the set(s) or groups. The values in this column may repeat if an ID is in more than one set.
values	character() vector of primaryKey values corresponding to rows to be deleted

Details

`avtable_import_set()` creates new rows in a table `<origin>_set`. One row will be created for each distinct value in the column identified by `set`. Each row entry has a corresponding column `<origin>` linking to one or more rows in the `<origin>` table, as given in the `member` column. The operation is somewhat like `split(member, set)`.

Value

`avtable`: a `tibble()` corresponding to the data with the name as given by `table`

`avtables`: a `tibble()` with columns `table`, `count`, and `colnames` corresponding to the tables / types available in the current workspace

`avtable_import()`: called for the side effect of uploading the data to the DATA tab

`avtable_import_set()`: a `character(1)` name of the imported tibble.

`avtable_delete`: a `logical(1)` indicating success or failure

`avtable_delete_values()`: a `logical(1)` vector indicating success or failure for each value in `values`

Functions

- `avtable(azure)`: List the contents of a particular table / type
- `avtables(azure)`: List the available tables / types
- `avtable_import(azure)`: Upload a dataset to the DATA tab
- `avtable_import_set(azure)`: Create a grouping table from an origin dataset
- `avtable_delete(azure)`: Delete a table / type
- `avtable_delete_values(azure)`: Delete rows from a table / type

Examples

```
if (interactive()) {
  library(dplyr)
  mtcars_tbl <-
      mtcars |>
      as_tibble(rownames = "model_id") |>
      mutate(model_id = gsub(" ", "-", model_id))

  avtable_import(
    mtcars_tbl,
    table = "testData",
    entity = "model_id"
  )

  avtable("testData")

  avtable("testData") |>                  # new 'testData_set' table
        avtable_import_set("testData", "cyl", "model_id")

  avtable_delete("testData_set")
```

```
  avtable_delete_values("testData", "Mazda-RX4")
}
```

---

avworkflow-methods *Azure Workflow methods*

Description

avworkflow_jobs() reports the status of workflow executions in the current workspace.

Usage

```
## S4 method for signature 'azure'
avworkflow_jobs(..., platform = cloud_platform())

avworkflow_jobs_inputs()
```

Arguments

...	Additional arguments passed to lower level functions (not used)
platform	azure() The cloud platform class to dispatch on as given by AnVILBase::cloud_platform. Typically not set manually as cloud_platform() returns the "azure" class for Azure workspaces on AnVIL.

Details

The avworkflow_jobs_inputs() function returns the input parameters for the workflow jobs as a tibble.

Value

avworkflow_jobs() returns a tibble with the status of the jobs in the current workspace.

Functions

- avworkflow_jobs(azure): List the status of workflow jobs

Examples

```
if (has_avworkspace(strict = TRUE, platform = azure()))
    ## from within AnVIL
    avworkflow_jobs()
```

avworkspace-methods *AnVIL Azure Workspace methods*

Description

AnVIL Azure Workspace methods

Usage

```
## S4 method for signature 'azure'
avworkspaces(..., platform = cloud_platform())

## S4 method for signature 'azure'
avworkspace_namespace(..., platform = cloud_platform())

## S4 method for signature 'azure'
avworkspace_name(..., platform = cloud_platform())

## S4 method for signature 'azure'
avworkspace(..., platform = cloud_platform())

## S4 method for signature 'azure'
avworkspace_clone(
  namespace = avworkspace_namespace(),
  name = avworkspace_name(),
  to_namespace = namespace,
  to_name,
  bucket_location = "US",
  ...,
  platform = cloud_platform()
)
```

Arguments

`...`	Additional arguments passed to lower level functions (not used)
`platform`	`azure()` The cloud platform class to dispatch on as given by AnVILBase::cloud_platform. Typically not set manually as `cloud_platform()` returns the `"azure"` class for Azure workspaces on AnVIL.
`namespace`	`character(1)` AnVIL workspace namespace as returned by, e.g., `avworkspace_namespace()`
`name`	`character(1)` AnVIL workspace name as returned by, eg., `avworkspace_name()`.
`to_namespace`	`character(1)` workspace (billing account) in which to make the clone.
`to_name`	`character(1)` name of the cloned workspace.
`bucket_location`	`character(1)` region in which bucket attached to the workspace should be created. The default is set to a single region ("US"); multi-region is available but more costly.

Value

avworkspaces(): a tibble table of available workspaces

avworkspace_namespace(): a character string of the workspace namespace

avworkspace_name(): a character string of the workspace name

avworkspace(): a character string of the workspace namespace and name combination

avworkspace_clone(): called for the side-effect of cloning a workspace to a new namespace and name.

Functions

- avworkspaces(azure): List workspaces
- avworkspace_namespace(azure): List the workspace namespace
- avworkspace_name(azure): Obtain the workspace name
- avworkspace(azure): Obtain the current workspace namespace and name combination
- avworkspace_clone(azure): Clone a workspace

---

az-utilities *az health check helpers*

Description

These functions provide checks for essential workspace tools and variables. az_exists checks for the presence of the az command line utility. az_healthcheck checks for the presence of the az command line as well as the essential environment variables.

Usage

```
az_exists()

az_health_check()
```

Value

az_exists returns a logical value indicating the presence of the az command line utility. az_healthcheck returns a logical value indicating the presence of the az command line utility and the essential environment variables.

Examples

```
if (interactive()) {
    az_exists()
    az_healthcheck()
}
```

azure-class *Azure platform class*

Description

This class represents the Azure platform.

Usage

```
azure()
```

Examples

```
az <- azure()
az
showClass(class(az))
```

azure-methods *A number of methods compatible with the Azure platform class.*

Description

A number of methods compatible with the Azure platform class.

Usage

```
## S4 method for signature 'azure'
avcopy(source, destination, dry = TRUE, ..., platform = cloud_platform())

## S4 method for signature 'azure'
avlist(..., platform = cloud_platform())

## S4 method for signature 'azure'
avremove(source, recursive = FALSE, ..., platform = cloud_platform())

## S4 method for signature 'azure'
avbackup(
  source,
  destination,
  recursive = TRUE,
  ...,
  platform = cloud_platform()
)

## S4 method for signature 'azure'
avrestore(
```

```
  source,
  destination,
  recursive = TRUE,
  ...,
  platform = cloud_platform()
)

## S4 method for signature 'azure'
avstorage(..., platform = cloud_platform())
```

Arguments

source	`character(1)` A relative file path corresponding to either the remote (`az_copy_from_storage`) or local (`az_copy_to_storage`) file location. Remote locations should be relative to the base directory in the Azure Storage Container e.g., `analyses/jupyter.log`.
destination	`character(1)` A relative file path corresponding to either the remote (`az_copy_to_storage`) or local (`az_copy_from_storage`) file location. Remote locations should be relative to the base directory in the Azure Storage Container. When not specified, it will default to the base directory of the remote location. The `to` path can be a folder path but must end in a forward slash (`/`). If the `to` path points to a non-existent directory, it will be created.
dry	`logical(1)` Whether to perform a dry run i.e., add the `--dry-run` flag to the command.
...	Additional arguments passed to lower level functions (not used)
platform	`azure()` The cloud platform class to dispatch on as given by AnVILBase::cloud_platform. Typically not set manually as `cloud_platform()` returns the `"azure"` class for Azure workspaces on AnVIL.
recursive	`logical(1)` Whether to recursively move or remove files in a directory. Only applies to `avremove`, `avbackup`, and `avrestore`. Default is `TRUE` for `backup` and `restore` operations and `FALSE` for `avremove`.

Details

The `recursive` argument for `avbackup` and `avrestore` is set to `TRUE` by default and `FALSE` for `avremove`. Note that wildcards are not supported for local or remote paths.

Value

- `avlist` - a `tibble` of files and metadata
- `avcopy` - called for the side effect of copying a file **to** or **from** the Azure Storage Container depending on the `source` and `destination` inputs
- `avremove` - called for the side effect of removing a file or folder
- `avbackup` - called for the side effect of copying a directory **to** the Azure Storage Container
- `avrestore` - called for the side effect of copying a directory **from** the Azure Storage Container
- `avstorage` - a URL string of the Azure Storage Container location
- `avworkspaces` - a tibble of workspaces on AnVIL

- avtable_import - a response list indicating successful upload
- avtable_delete_values - when successful, a NULL value

Functions

- avcopy(azure): a generalized interface for either az_copy_from_storage or az_copy_to_storage; deduced from the source and destination inputs
- avlist(azure): list all the files in the Azure Storage Container
- avremove(azure): remove a file or directory from the Azure Storage
- avbackup(azure): copy a directory from the workspace environment to the Azure Storage Container
- avrestore(azure): copy a file or directory from the Azure Storage Container to the workspace environment
- avstorage(azure): The base URI string used to move data to and from the Azure Storage Container

Examples

```
if (interactive()) {

  avlist()

  ## local -> remote
  ## using general interface avcopy
  avcopy("jupyter.log", "analyses/jupyter.log")

  ## upload a directory
  avbackup("./test/", "analyses/test/")

  ## using general interface az_copy
  avcopy("analyses/jupyter.log", "./jupyter.log")

  ## download a directory
  avrestore("analyses/test/", "./test/")

  avremove("analyses/jupyter.log")

}
```

---

az_copy-helpers *Azure Copy command line utility helpers*

Description

These functions invoke the azcopy command line utility. The utilities make use of a managed SAS token to mainly transfer files from the Azure workspace to the Azure Storage container. See az_sas_token for credential details. The results of azcopy copy commands are returned as an azcopyStatus object which has S3 methods to print and convert to logical.

Usage

```
az_copy_from_storage(from, to = "./", recursive = FALSE, dry = TRUE)

az_copy_to_storage(from, to, recursive = FALSE, dry = TRUE)

## S3 method for class 'azcopyStatus'
as.logical(x, ...)

## S3 method for class 'azcopyStatus'
print(x, ..., verbose = FALSE)

## S3 method for class 'azcopyStatus'
summary(object, ...)
```

Arguments

from	character(1) A relative file path corresponding to either the remote (az_copy_from_storage) or local (az_copy_to_storage) file location. Remote locations should be relative to the base directory in the Azure Storage Container e.g., analyses/jupyter.log.
to	character(1) A relative file path corresponding to either the remote (az_copy_to_storage) or local (az_copy_from_storage) file location. Remote locations should be relative to the base directory in the Azure Storage Container. When not specified, it will default to the base directory of the remote location. The to path can be a folder path but must end in a forward slash (/). If the to path points to a non-existent directory, it will be created.
recursive	logical(1) Whether to recursively move or remove files in a directory. Only applies to avremove, avbackup, and avrestore. Default is TRUE for backup and restore operations and FALSE for avremove.
dry	logical(1) Whether to perform a dry run i.e., add the --dry-run flag to the command.
x	azcopyStatus object to be checked; usually the output of avcopy operations
...	Additional arguments (not used).
verbose	logical(1) Print the INFO lines from the azcopy output
object	azcopyStatus object to be summarized; usually the output of avcopy operations

Details

- az_copy_from_storage - copy a file from the Azure Storage Container to the workspace environment
- az_copy_to_storage - copy a file from the workspace environment to the Azure Storage Container

Value

- az_copy_from_storage - called for the side effect of copying a file **from** the Azure Storage Container

- az_copy_to_storage - called for the side effect of copying a file **to** the Azure Storage Container

Functions

- as.logical(azcopyStatus): Convert results of azcopy operations to logical values
- print(azcopyStatus): Print the results of azcopy operations
- summary(azcopyStatus): Get a summary of the results of azcopy operations

Examples

```
if (interactive()) {

  ## local -> remote
  az_copy_to_storage("jupyter.log", "analyses/jupyter.log")
  az_copy_to_storage("jupyter.log", "analyses/test/")

  ## placed in the base storage UUID directory
  az_copy_to_storage("jupyter.log")

  ## remote -> local
  az_copy_from_storage("analyses/jupyter.log", "jupyter.log")
  ## download to the current directory
  az_copy_from_storage("analyses/jupyter.log")

}
```

---

az_sas_token *Obtain the Shared Access Signature (SAS) for the Azure Storage*

Description

The function provides a user delegation SAS token for management of resources. Mainly used in other functions to move files to and from the Azure Storage Container

Usage

```
az_sas_token(sasExpirationDuration = 28800)
```

Arguments

sasExpirationDuration
: numeric(1) The number of seconds until the SAS token expires (default: 28,800 seconds)

Value

A list of two elements named token and url

Examples

```
if (interactive()) {
  sas <- az_sas_token()
  sas[["token"]]
  sas[["url"]]
}
```

---

az_token *Generate an Azure authentication token with the* az *command line utility*

Description

This function generates an Azure authentication token with the az command line utility. The token is used to authenticate with the Azure services. This function is called internally by the az_* functions. It is not meant to be called directly.

Usage

```
az_token()
```

Value

A character string containing the authentication token with a "Bearer" prefix.

---

has_avworkspace-methods *Helper to check if the current environment is within an Azure workspace*

Description

has_avworkspace() checks that the AnVIL environment is set up to work with Azure. If strict = TRUE, it also checks that the workspace name is set.

Usage

```
## S4 method for signature 'azure'
has_avworkspace(strict = FALSE, ..., platform = cloud_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

`logical(1)` `TRUE` if the AnVIL environment is set up properly to interact with Azure, otherwise `FALSE`.

Functions

- `has_avworkspace(azure)`: Check if the AnVIL environment is set up

Examples

```
has_avworkspace(platform = azure())
```

`workspace-dev-ops` *Functions to work with workspace data for developers*

Description

These group of functions will allow you to manipulate tables in the "DATA" tab. Example operations include moving a flat Tab-Separated Values (TSV) file into the workspace, deleting records, deleting tables, adding a single row, retrieving a single row, and retrieving the data table. Note that the API used refers to tables as `types`.

Usage

```
upload_tsv(
  tsv_file,
  type = tools::file_path_sans_ext(basename(tsv_file)),
  primaryKey = NULL
)

download_tsv(type)

delete_type_id(type, id)

add_type_id(row, type, id = row[[1L]])

get_type_id(type, id)

delete_type(type)
```

Arguments

`tsv_file`	`character(1)` A path to a tab-separated values file
`type`	`character(1)` A nickname for the uploaded dataset important for retreival. By default, the file name will be used.

primaryKey	character(1) The optional column name to uniquely identify a record. By default, the first column is used as the primary key and all values in the column must be unique.
id	character(1) The value in the primaryKey column that indicates the row to be removed.
row	tibble() or data.frame() A single row to add to an existing table. The row must have the same column names as the table. The primaryKey column must be unique.

Details

These functions use the Workspace Data Services (WDS) API. Current operations that affect the "DATA" tab include:

- upload_tsv - a POST request using a TSV file that populates the data
- download_tsv - a GET request with the data name (type argument) in upload_tsv to represent the data locally as a tibble
- delete_type_id - a DELETE request to remove a record or row from type
- add_type_id - a PUT request to add a single row to an existing table (type)
- get_type_id - a GET request to retrieve a single row from an existing table (type)
- delete_type - a DELETE request to remove then entire data set (type)

Value

- upload_tsv - A response list indicating successful upload
- download_tsv - A tibble corresponding to the data labeled with type
- delete_type_id; delete_type - When successful, a NULL value

Examples

```
if (interactive()) {
  library(dplyr)
  type <- "model"
  mtcars_tbl <-
      mtcars |>
      as_tibble(rownames = "model_id") |>
      mutate(model_id = gsub(" ", "-", model_id))

  tsv_file <- tempfile()
  readr::write_tsv(mtcars_tbl, tsv_file)
  upload_tsv(
    tsv_file = tsv_file,
    type = "testData",
    primaryKey = "model_id"
  )

  download_tsv("testData")
```

```
  ## create an example single row tibble for add_type_id
  datsun <- filter(mtcars_tbl, model_id == "Datsun-710")
  ## change the model_id to be unique
  datsun[["model_id"]] <- "Datsun-512"

  add_type_id(row = datsun, type = "testData")

  get_type_id("testData", "Datsun-512")

  delete_type_id("testData", "Datsun-512")

  delete_type("testData")
}
```

`workspace-env` *Access Terra on Azure workspace session variables*

Description

A group of functions that return environment variables in the Terra Azure workspace. The Workspace Data Service URL sends out an API `GET` request to obtain the data services URL for uploading data to the workspace "DATA" tab.

Usage

```
workspace_id()

workspace_storage_cont_id()

workspace_storage_cont_url()

wds_api_version()

workspace_data_service_url()

cbas_url()
```

Value

- `workspace_id` - A UUID string referring to "workspaceId" or "workspaceid" in API calls
- `workspace_storage_cont_id` - A UUID string identifiying the resource storage container owned by the user account, a.k.a. "resourceId"
- `workspace_storage_cont_url` - The base URI string used to move data to and from the Azure Storage Container
- `wds_api_version` - The version of the Workspace Data Service API, defaults to "v0.2"
- `workspace_data_service_url` - The base URI string used to move data to to and from the workspace "DATA" tab
- `cbas_url` - The base URI string used to query the workflow submission history

Examples

```
workspace_id()
workspace_storage_cont_id()
workspace_storage_cont_url()
if (interactive()) {
  workspace_data_service_url()
  cbas_url()
}
```

Index

∗ **internal**
    AnVILAz-package, 2
    az_copy-helpers, 12
    az_token, 15
    workspace-dev-ops, 16
.azure *(azure-class)*, 10

add_type_id *(workspace-dev-ops)*, 16
AnVILAz *(AnVILAz-package)*, 2
AnVILAz-package, 2
AnVILBase::cloud_platform, *4, 5*, *7, 8*, *11*
as.logical.azcopyStatus
        (az_copy-helpers), 12
avbackup,azure-method *(azure-methods)*,
        10
avcopy,azure-method *(azure-methods)*, 10
avlist,azure-method *(azure-methods)*, 10
avnotebooks *(avnotebooks-methods)*, 3
avnotebooks,azure-method
        (avnotebooks-methods), 3
avnotebooks-methods, 3
avnotebooks_delocalize
        (avnotebooks-methods), 3
avnotebooks_delocalize,azure-method
        (avnotebooks-methods), 3
avnotebooks_localize
        (avnotebooks-methods), 3
avnotebooks_localize,azure-method
        (avnotebooks-methods), 3
avremove,azure-method *(azure-methods)*,
        10
avrestore,azure-method *(azure-methods)*,
        10
avstorage,azure-method *(azure-methods)*,
        10
avtable *(avtable-methods)*, 4
avtable,azure-method *(avtable-methods)*,
        4
avtable-methods, 4
avtable_delete *(avtable-methods)*, 4
avtable_delete,azure-method
        (avtable-methods), 4
avtable_delete_values
        (avtable-methods), 4
avtable_delete_values,azure-method
        (avtable-methods), 4
avtable_import *(avtable-methods)*, 4
avtable_import,azure-method
        (avtable-methods), 4
avtable_import_set *(avtable-methods)*, 4
avtable_import_set,azure-method
        (avtable-methods), 4
avtables *(avtable-methods)*, 4
avtables,azure-method
        (avtable-methods), 4
avworkflow-methods, 7
avworkflow_jobs *(avworkflow-methods)*, 7
avworkflow_jobs,azure-method
        (avworkflow-methods), 7
avworkflow_jobs_inputs
        (avworkflow-methods), 7
avworkspace,azure-method
        (avworkspace-methods), 8
avworkspace-methods, 8
avworkspace_clone,azure-method
        (avworkspace-methods), 8
avworkspace_name,azure-method
        (avworkspace-methods), 8
avworkspace_namespace,azure-method
        (avworkspace-methods), 8
avworkspaces,azure-method
        (avworkspace-methods), 8
az-utilities, 9
az_copy-helpers, 12
az_copy_from_storage *(az_copy-helpers)*,
        12
az_copy_to_storage *(az_copy-helpers)*, 12
az_exists *(az-utilities)*, 9
az_health_check *(az-utilities)*, 9

az_sas_token, 14
az_token, 15
azure (azure-class), 10
azure-class, 10
azure-methods, 10

cbas_url (workspace-env), 18

delete_type (workspace-dev-ops), 16
delete_type_id (workspace-dev-ops), 16
download_tsv (workspace-dev-ops), 16

get_type_id (workspace-dev-ops), 16

has_avworkspace,azure-method
        (has_avworkspace-methods), 15
has_avworkspace-methods, 15

print.azcopyStatus (az_copy-helpers), 12

summary.azcopyStatus (az_copy-helpers),
        12

upload_tsv (workspace-dev-ops), 16

wds_api_version (workspace-env), 18
workspace-dev-ops, 16
workspace-env, 18
workspace_data_service_url
        (workspace-env), 18
workspace_id (workspace-env), 18
workspace_storage_cont_id
        (workspace-env), 18
workspace_storage_cont_url
        (workspace-env), 18
workspace_storage_container_id
        (workspace-env), 18
workspace_storage_container_url
        (workspace-env), 18