

Package ‘AnVILWorkflow’

September 22, 2026

Title Run workflows implemented in Terra/AnVIL workspace

Version 1.12.1

Date 2023-4-23

Description The AnVIL is a cloud computing resource developed in part by the National Human Genome Research Institute. The main cloud-based genomics platform deployed by the AnVIL project is Terra. The AnVILWorkflow package allows remote access to Terra implemented workflows, enabling end-user to utilize Terra/ AnVIL provided resources - such as data, workflows, and flexible/scalable computing resources - through the conventional R functions.

Depends R (>= 4.4.0), AnVILGCP, AnVILBase, httr

Imports AnVIL, GCPtools, dplyr, jsonlite, rlang, tibble, tidyr, utils, methods, plyr, stringr

Suggests knitr, rmarkdown, BiocStyle

License Artistic-2.0

biocViews Infrastructure, Software

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL <https://github.com/shbrief/AnVILWorkflow>

BugReports <https://github.com/shbrief/AnVILWorkflow/issues>

git_url <https://git.bioconductor.org/packages/AnVILWorkflow>

git_branch RELEASE_3_23

git_last_commit 9bff89f

git_last_commit_date 2026-08-31

Repository Bioconductor 3.23

Date/Publication 2026-09-22

Author Sehyun Oh [aut, cre] (ORCID: <<https://orcid.org/0000-0002-9490-3061>>),
Marcel Ramos [ctb] (ORCID: <<https://orcid.org/0000-0002-3242-0582>>),
Kai Gravel-Pucillo [aut]
Maintainer Sehyun Oh <shbrief@gmail.com>

Contents

.biobakery_currentInput	2
.getAgeMinMax	3
.get_workflow_fullname	4
.get_workspace_fullname	4
.nonMetadataOutputs	5
.search_keyword	5
.stop_quietly	6
.tableToString	6
AnVILBrowse	7
availableAnalysis	8
cloneWorkspace	9
currentInput	10
findInputName	11
getAllDataTables	11
getAllWorkflows	12
getAllWorkspaces	13
getDashboard	13
getData	14
getOutput	14
getWorkflowConfig	15
getWorkflows	16
getWorkspaces	16
monitorWorkflow	17
runWorkflow	17
setCloudEnv	18
stopWorkflow	19
updateInput	20
Index	22

.biobakery_currentInput
<i>Check the current input arguments</i>

Description

Check the current input arguments

Usage

```
.biobakery_currentInput(config)
```

Arguments

config Workflow configuration. Output from the [getWorkflowConfig](#) function.

Value

A list length of two, including inputListPath and inputFilePath.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  config <- avworkflow_configuration_get(
    workflow_namespace = "mtx_workflow_biobakery_version3",
    workflow_name = "mtx_workflow_biobakery_version3",
    workspace_namespace = "waldronlab-terra-rstudio",
    workspace_name = "mtx_workflow_biobakery_version3_template")
  biobakery_inputs <- .biobakery_currentInput(config)
}
```

.getAgeMinMax

Apply to the returned value from .avSampleData function

Description

Apply to the returned value from .avSampleData function

Usage

```
.getAgeMinMax(sampleData)
```

Arguments

sampleData A data frame.

`.get_workflow_fullname`*Get the workflow namespace and name*

Description

Use this internally when `setCloudEnv` is already run.

Usage

```
.get_workflow_fullname(workspaceName, workflowName = NULL)
```

Arguments

<code>workspaceName</code>	A character. Name of the workspace to use.
<code>workflowName</code>	A character. Name of the workflow to run. If a single workflow is available under the selected workspace, this function will check the input of that workflow under the default (NULL). If there are multiple workflows available, you should specify the workflow.

Value

A character of `workflow_namespace/workflow_name`

`.get_workspace_fullname`*Get the fullname of the workspace*

Description

Get the fullname of the workspace

Usage

```
.get_workspace_fullname(workspaceName)
```

Arguments

<code>workspaceName</code>	Character(1). Name of the template workspace name you want to clone. You can provide name or namespace/name.
----------------------------	--

Value

Character(1) of `workspaceNamespace/workspaceName`

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  .get_workspace_fullname(workspaceName = "Bioconductor-Workflow-DESeq2")
}
```

<code>.nonMetadataOutputs</code>	<i>Subset to non-metadata output files</i>
----------------------------------	--

Description

Subset to non-metadata output files

Usage

```
.nonMetadataOutputs(workflowOutputs)
```

Arguments

`workflowOutputs`

A data frame of workflow outputs with four columns: file, workflow, task, and path. Returned value from [avworkflow_files](#).

Value

A character vector containing the names of non-metadata output files

<code>.search_keyword</code>	<i>Search keywords in a given metadata table</i>
------------------------------	--

Description

Search keywords in a given metadata table

Usage

```
.search_keyword(keyword, metadata)
```

Arguments

<code>keyword</code>	A character(1). Regular expression is accepted. For example, you can search multiple keywords separated by the vertical bar (" ").
<code>metadata</code>	A data frame. Metadata table of workspace, workflow, or AnVIL data.

Value

A data frame. A subset of input metadata table with the rows containing the keyword.

<code>.stop_quietly</code>	<i>Stop the execution without error messages</i>
----------------------------	--

Description

Stop the execution without error messages

Usage

`.stop_quietly()`

Value

Stop the function call without warning/error messages.

<code>.tableToString</code>	<i>Summary table into a single string where the column name and the value is separated by sep and columns are separated by delim</i>
-----------------------------	--

Description

Summary table into a single string where the column name and the value is separated by sep and columns are separated by delim

Usage

`.tableToString(tb, sep = ":", delim = ";")`

Arguments

- tb A table. Output from the table function.
- sep A delimiter to separate the column name and value
- delim A delimiter to separate columns

AnVILBrowse

*Search AnVIL workspaces using keywords***Description**

Search AnVIL workspaces using keywords

Usage

```
AnVILBrowse(
  keyword,
  searchFrom = "all",
  returnFrom = NULL,
  metaTables = "default",
  minAge = 0,
  maxAge = 130,
  minCount = 0,
  workspaceTable = NULL,
  workflowTable = NULL,
  dataTable = NULL
)
```

Arguments

keyword	A character(1). Regular expression is accepted. For example, you can search multiple keywords separated by the vertical bar (" ").
searchFrom	Under the default (all), all the workspaces containing keywords in their workspace/workflow/data will be returned. The other available options are workspace, workflow, and data.
returnFrom	Under the default (NULL), the same data type as for searchFrom will be used, while searchFrom = "all" returns workspaces.
metaTables	Under the default (default), all the publicly accessible AnVIL workspaces will be subjected for search. If you want to search in all the workspaces you have access to, set this argument as custom, and provide the inputs for workspaceTable, workflowTable, and dataTable arguments.
minAge	A numeric (1). Any data with a maximum participant age lower than this parameter will be excluded from the output. Under the default (0), no filtering will be applied. Data entries with no maximum participant age listed will not be removed by this argument.
maxAge	A numeric (1). Any data with a minimum participant age higher than this parameter will be excluded from the output. Under the default (130), no data entries will be removed due to the minimum participant age. Data entries with no minimum participant age listed will not be removed by this argument.

minCount	A numeric (1). Any data with the number of subjects fewer than this parameter will be excluded from the output. Under the default (0), no data entries will be removed due to participant count. Data entries with no participant count listed will not be removed by this argument.
workspaceTable	A data frame. This argument is counted only when metaTables = "custom". Provide the output from the getWorkspaces function, to search in all the workspaces you have access to.
workflowTable	A data frame. This argument is counted only when metaTables = "custom". Provide the output from the getWorkflows function, to search in all the workflows you have access to.
dataTable	A data frame. This argument is counted only when metaTables = "custom". Provide the output from the getData function, to search in all the AnVIL data you have access to.

Value

A data frame of AnVIL resources containing keywords. Depending on the returnFrom argument, it can be workspaces, workflows, or data. Under the default returnFrom = NULL, it returns the same data type as specified in searchFrom or workspace for searchFrom = "all".

Examples

```
AnVILBrowse("malaria")
AnVILBrowse("resistance")
AnVILBrowse("resistance", searchFrom = "workflow")
```

availableAnalysis	<i>Find the available analysis</i>
-------------------	------------------------------------

Description

This function shows the available analyses and the brief descriptions of them.

Usage

```
availableAnalysis(curatedOnly = TRUE, keyword = NULL)
```

Arguments

curatedOnly	Default is TRUE, returning only workspaces that offer simplified input configuration by this package. If it is set to FALSE, all the workspaces
keyword	Default is NULL. When this argument is provided as a character(1), it will return only the workspaces containing the keyword and the user has an access to.

Value

A data frame. The analysis columns shows the name of the available analyses, which is the required input (analysis argument) for the functions implemented in AnVILWorkflow package.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  availableAnalysis()
}
```

cloneWorkspace	<i>Clone template workspace</i>
----------------	---------------------------------

Description

This function makes your own copy of the existing workspace, selected through templateName or analysis. Your copied/cloned workspace name will be workspaceName and any computing cost will be charged to the billing linked to your billingProjectName. You should provide at least one argument templateName or analysis.

Usage

```
cloneWorkspace(
  workspaceName,
  templateName = "",
  analysis = NULL,
  bucketLocation = "us-central1",
  accountEmail = GCPTools::gcloud_account(),
  billingProjectName = GCPTools::gcloud_project()
)
```

Arguments

workspaceName	Name of the workspace you are creating
templateName	Character(1). Name of the template workspace name you want to clone. You can provide name or namespace/name.
analysis	Character(1). Name of the analysis you want to clone it's workspace. The list of available analyses can be found using availableAnalysis .
bucketLocation	Character(1). Region in which bucket attached to the workspace should be created. Default is us-central1. For multi-region, enter US.
accountEmail	Character(1). Email linked to Terra account
billingProjectName	Character(1). Name of the billing project

Value

Name of the cloned workspace

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  cloneWorkspace(workspaceName = "salmon",
                 templateName = "Bioconductor-Workflow-DESeq2")
}
```

currentInput	<i>Check the current input arguments</i>
--------------	--

Description

Check the current input arguments

Usage

```
currentInput(workspaceName, config, requiredInputOnly = TRUE, analysis = NULL)
```

Arguments

- workspaceName Name of the workspace
- config Workflow configuration. Output from the [getWorkflowConfig](#) function.
- requiredInputOnly Under the default (TRUE), only the required inputs are returned.
- analysis If specified, only the minimally required inputs for a given workflow will be returned.

Value

A data.frame for the inputs defined in a workflow configuration.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  workspaceName <- "Bioconductor-Workflow-DESeq2"
  config <- getWorkflowConfig(workspaceName)
  currentInput(workspaceName = workspaceName, config = config)
}
```

findInputName	<i>Find the root entity name</i>
---------------	----------------------------------

Description

Find the root entity name

Usage

```
findInputName(workspaceName, rootEntity = "", nameOnly = TRUE)
```

Arguments

workspaceName	Name of the workspace
rootEntity	A character. Type of root entity for Terra's data model. For example, participant, participant_set, sample, etc.
nameOnly	Under the default (TRUE), only the names of a given root entity type will be returned.

Value

A character vector of input names under the given root entity.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  .findInputName(
    workspaceName = "Bioconductor-Workflow-DESeq2",
    rootEntity = "participant_set")
}
```

getAllDataTables	<i>Get all the data tables</i>
------------------	--------------------------------

Description

Get all the data tables

Usage

```
getAllDataTables(workspaces = NULL)
```

Arguments

workspaces A character vector. Under the default (NULL), all the data tables from all the workspaces user has access to will be returned. If you specify this, the data tables only from the specified workspace(s) will be returned.

Value

A Data Frame of all the data tables

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  allDataTables <- getAllDataTables()
}
```

getAllWorkflows	<i>Collect workflows from all workspaces a user has access to</i>
-----------------	---

Description

Collect workflows from all workspaces a user has access to

Usage

```
getAllWorkflows(workspaces = NULL)
```

Arguments

workspaces Under the default (NULL), workflows from all the workspaces a user has access to will be collected.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  allWorkflows <- getAllWorkflows()
}
```

getAllWorkspaces	<i>Get AnVIL workspaces</i>
------------------	-----------------------------

Description

Different from [avworkspaces](#)

Usage

```
getAllWorkspaces()
```

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  allWorkspaces <- getAllWorkspaces()
}
```

getDashboard	<i>Print out Dashboard contents</i>
--------------	-------------------------------------

Description

This function prints out the Dashboard contents of the target workspace. You can provide either workspaceName or analysis. If both values are provided, this function will use workspaceName argument over analysis argument.

Usage

```
getDashboard(workspaceName = "", analysis = NULL)
```

Arguments

workspaceName	The name of the workspace you want to get the overview provided through the Dashboard.
analysis	The name of the analysis use want to check the Dashboard of. The list of available analyses can be found with availableAnalysis().

Value

The last modified date as a message, followed by the Dashboard contents from the target workspace.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  getDashboard(analysis = "salmon")
  getDashboard(workspaceName = "Bioconductor-Workflow-DESeq2")
}
```

getData

Creates a metadata table of data from all workspaces provided

Description

This function usually takes a long time to run due to the large volume of AnVIL data.

Usage

```
getData(allWorkspaces)
```

Arguments

`allWorkspaces` A data frame of all the workspaces you have access to. An output from the `getWorkspaces` function.

getOutput

Download output files from Terra

Description

Download output files from Terra

Usage

```
getOutput(
  workspaceName,
  submissionId = NULL,
  keyword = NULL,
  dest_dir = ".",
  dry = TRUE
)
```

Arguments

workspaceName	Name of the workspace
submissionId	Submission Id. If it's not provided, the most recent submission id with the 'succeeded' status will be used.
keyword	A character string containing a regular expression to be matched in the output file name. Under the default NULL, all the outputs from the workflow, including log files, will be returned.
dest_dir	Path to the directory where downloaded files are saved
dry	To download the output data, set dry = FALSE.

Value

If "dry=TRUE", this function will return a data frame with two columns named 'filename' and 'name'.

- filename: Name of the actual output files.
- name: Name of the output defined in your workflow script. This is how configuration refers the outputs.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  getOutput(workspaceName = "Bioconductor-Workflow-DESeq2")
}
```

getWorkflowConfig	<i>Check the workflow configuration</i>
-------------------	---

Description

Check the workflow configuration

Usage

```
getWorkflowConfig(workspaceName, workflowName = NULL)
```

Arguments

workspaceName	Name of the workspace
workflowName	Name of the workflow to run. If a single workflow is available under the specified workspace, this function will check the input of that workflow under the default (NULL). If there are multiple workflows available, you should specify the workflow.

Value

A data.frame for the inputs defined in a workflow configuration.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  config <- getWorkflowConfig(workspaceName = "Bioconductor-Workflow-DESeq2")
  config
}
```

getWorkflows	<i>Creates a metadata table of workflows from all workspaces provided</i>
--------------	---

Description

Creates a metadata table of workflows from all workspaces provided

Usage

```
getWorkflows(allWorkspaces)
```

Arguments

allWorkspaces A data frame of all the workspaces you have access to. An output from the getWorkspaces function.

getWorkspaces	<i>Creates a metadata table of all workspaces</i>
---------------	---

Description

Metadata of all the GCP-based workspaces are collected.

Usage

```
getWorkspaces()
```

monitorWorkflow	<i>Check the status of submitted jobs</i>
-----------------	---

Description

Check the status of submitted jobs

Usage

```
monitorWorkflow(workspaceName)
```

Arguments

workspaceName Character(1). Name of the workspace

Value

A tibble summarizing submitted workflow jobs. Contains information such as submission Id, submission date, and submission status.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp())) {
  monitorWorkflow(workspaceName = "Bioconductor-Workflow-DESeq2")
}
```

runWorkflow	<i>Launch Terra workflow</i>
-------------	------------------------------

Description

Launch Terra workflow

Usage

```
runWorkflow(
  workspaceName,
  config,
  workflowName = NULL,
  useCallCache = TRUE,
  inputName = NULL
)
```

Arguments

workspaceName	Name of the workspace that contains the workflow(s) you want to launch.
config	Workflow configuration. Output from the getWorkflowConfig function.
workflowName	Name of the workflow to run. If this input is not provided but there is only a single workflow available, the function will automatically use the only workflow.
useCallCache	A logical. Under the default condition (TRUE), call cache will be used.
inputName	Name of you input entity. If the workflow is using Terra's data model, this is required. The available entities can be found using the findInputName function.

Value

This function will print out whether the call for workflow launching was successful or not.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp()) && "salmon" %in% avworkspaces()$name) {
  runWorkflow(workspaceName = "salmon")
}
```

setCloudEnv

Setup Google Cloud Account and Project

Description

Setup Google Cloud Account and Project

Usage

```
setCloudEnv(
  accountEmail = GCPtools::gcloud_account(),
  billingProjectName = GCPtools::gcloud_project(),
  message = TRUE
)
```

Arguments

accountEmail	Character(1). Email linked to your Terra account.
billingProjectName	Character(1). Name of the billing project, which is the gcloud account.
message	Under the default (TRUE), this function will print out Google Cloud Account and Billing Project set in the working environment

Value

Terra/AnVIL working environment - Google Cloud billing account and the billing project name - will be printed out.

Examples

```
if (GCPtools::gcloud_exists()) {
  setCloudEnv()
}
```

stopWorkflow	<i>Abort submitted job</i>
--------------	----------------------------

Description

Abort submitted job

Usage

```
stopWorkflow(workspaceName, submissionId = NULL, dry = TRUE)
```

Arguments

workspaceName	Name of the workspace
submissionId	A character. Submission ID you want to abort. You can find the submission id using monitorWorkflow function. If it is not defined, the most recent submission will be aborted.
dry	Logical(1) when TRUE (default), report the consequences but do not perform the action requested. When FALSE, perform the action.

Value

This function will print out whether the call for workflow abortion was successful or not. In case it was unsuccessful, the diagnosis will be suggested as a part of the message.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp()) && "salmon" %in% avworkspaces()$name) {
  stopWorkflow(workspaceName = "salmon")
}
```

updateInput	<i>Update the input</i>
-------------	-------------------------

Description

Update the input

Usage

```
updateInput(
  workspaceName,
  inputs,
  config,
  workflowName = NULL,
  dry = TRUE,
  verbose = TRUE
)
```

Arguments

workspaceName	Name of the workspace
inputs	A tibble containing new input values. Provide the modified version of the current input table, which is the output from currentInput function. IMPORTANT: all the attributes should be provided as a character vector format and any string type attributes (inputType = String) should have escaped quotation mark (\").
config	Workflow configuration. Output from the getWorkflowConfig function.
workflowName	Name of the workflow to run. If a single workflow is available under the specified workspace, this function will check the input of that workflow under the default (NULL). If there are multiple workflows available, you should specify the workflow.
dry	Logical(1). When TRUE (default), report the updated configuration but do not perform the action requested in Terra. When FALSE, inputs in Terra/AnVIL will be updated.
verbose	Logical(1). When TRUE (default), this function will print the updated input.

Value

With verbose=TRUE, a list of updated inputs will be printed. A successful execution of the function will update the input configuration of the target workflow in Terra/AnVIL.

Examples

```
library(AnVILBase)
if (has_avworkspace(strict = TRUE, platform = AnVILGCP::gcp()) && "salmon" %in% avworkspaces()$name) {
  config <- getWorkflowConfig(workspaceName = "salmon")
  inputs <- currentInput("salmon", config)
```

```
## Modify the contents of 'inputs' table for your analysis
updateInput("salmon", inputs, config)
}
```

Index

* **internal**

- .biobakery_currentInput, [2](#)
 - .get_workflow_fullname, [4](#)
 - .get_workspace_fullname, [4](#)
 - .nonMetadataOutputs, [5](#)
 - .stop_quietly, [6](#)
- .biobakery_currentInput, [2](#)
- .getAgeMinMax, [3](#)
- .get_workflow_fullname, [4](#)
- .get_workspace_fullname, [4](#)
- .nonMetadataOutputs, [5](#)
- .search_keyword, [5](#)
- .stop_quietly, [6](#)
- .tableToString, [6](#)
- AnVILBrowse, [7](#)
- availableAnalysis, [8](#), [9](#)
- avworkflow_files, [5](#)
- avworkspaces, [13](#)
- cloneWorkspace, [9](#)
- currentInput, [10](#), [20](#)
- findInputName, [11](#)
- getAllDataTables, [11](#)
- getAllWorkflows, [12](#)
- getAllWorkspaces, [13](#)
- getDashboard, [13](#)
- getData, [14](#)
- getOutput, [14](#)
- getWorkflowConfig, [3](#), [10](#), [15](#), [18](#), [20](#)
- getWorkflows, [16](#)
- getWorkspaces, [16](#)
- monitorWorkflow, [17](#)
- runWorkflow, [17](#)
- setCloudEnv, [4](#), [18](#)
- stopWorkflow, [19](#)
- updateInput, [20](#)