Package 'AnVILVRS'

November 10, 2026

Type Package

Title Expose the vrs_anvil_toolkit Python package via R

Version 0.99.19

Description Process Variant Call Format (VCF) files and perform lookup operations on Genomic Variation Representation Service (GA4GH VRS) identifiers. The GA4GH VRS identifiers provide a standardized way to represent genomic variations, making it easier to exchange and share genomic information.

Depends R (>= 4.5.0)

Imports AnVILGCP, BiocBaseUtils, BiocFileCache, reticulate

Suggests AnVILBase, BiocStyle, gert, knitr, readr, rmarkdown, tinytest

biocViews Software, BiologicalQuestion, VariantAnnotation, Infrastructure, ThirdPartyClient

VignetteBuilder knitr

License Artistic-2.0

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

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

Encoding UTF-8

RoxygenNote 7.3.3

Date 2026-04-07

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

git_branch devel

git_last_commit 70f940e

git_last_commit_date 2026-04-07

Repository Bioconductor 3.24

Date/Publication 2026-11-09

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

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

Contents

AnVILVRS-package . . . 2
build_vrs_index . . . 2
get_caf . . . 3
get_pop_descriptor . . . 4
get_seqrepo . . . 5
get_vrs_id . . . 6
install_AnVILVRS . . . 7
setup_vrs_toolkit . . . 8

Index **9**

`AnVILVRS-package` *AnVILVRS: Expose the vrs_anvil_toolkit Python package via R*

Description

Process Variant Call Format (VCF) files and perform lookup operations on Genomic Variation Representation Service (GA4GH VRS) identifiers. The GA4GH VRS identifiers provide a standardized way to represent genomic variations, making it easier to exchange and share genomic information.

Author(s)

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

See Also

Useful links:

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

---

`build_vrs_index` *Generate a VRS-VCF index database*

Description

Use the 'vrsix' command-line tool to create a SQLite index database from a VRS-annotated VCF file.

Usage

```
build_vrs_index(
  vcf = .get_fixture_vcf(),
  dbfile = "1000g_chr1_index.db",
  force = FALSE
)
```

Arguments

vcf	‘character(1)‘ Path to the bgzipped VCF file containing the VRS annotations. Defaults to a 1kGP fixture within the toolkit.
dbfile	‘character(1)‘ The name of the index database file to be created. Defaults to "1000g_chr1_index.db". Note that the dbfile is added in the ‘setup_vrs_toolkit()‘ directory, so the full path to the generated database will be ‘file.path(setup_vrs_toolkit(), dbfile)‘.
force	‘logical(1)‘ Whether to force the generation of the index database even if it already exists. Defaults to ‘FALSE‘.

Details

This function indexes the VRS annotations in a specific 1000 Genomes project VCF fixture from the ‘vrs_anvil_toolkit‘. Note that if the ‘dbfile‘ already exists, ‘vrsix‘ will re-process the VCF file. While it typically avoids duplicating entries in the location table, the re-scanning process can be time-consuming for large files and may increase the database file size due to metadata updates.

This function uses ‘setup_vrs_toolkit()‘ to find the toolkit directory and is influenced by the ‘AnVIL-VRS.toolkit_path‘ option.

Value

‘character(1)‘ The normalized path to the generated index database.

Examples

```
build_vrs_index()
```

---

get_caf *Get the allele frequency from 1000 Genomes Project VCF file*

Description

Get the allele frequency from 1000 Genomes Project VCF file

Usage

```
get_caf(
  vrs_id,
  vcf,
  vcf_index,
  phenotype = "USA",
  pop_desc_file = NULL,
  toolkit_dir = setup_vrs_toolkit()
)
```

Arguments

vrs_id	‘character(1)‘ A GA4GH VRS Allele ID
vcf	‘character(1)‘ Path to a bgzipped VCF file containing 1000 Genomes Project variants with allele frequency annotations.
vcf_index	‘character(1)‘ Path to the index file for the VCF file.
phenotype	‘character(1)‘ Population or super-population code from the 1000 Genomes Project. See ‘anvil-datastorage/AnVIL_1000G_PRIMED-data-model‘ workspace under ‘data/population_descriptor.tsv‘ for valid codes. Defaults to "USA".
pop_desc_file	‘character(1)‘ Path to the population descriptor file downloaded from the AnVIL_1000G_PRIMED-data-model workspace. If ‘NULL‘ (default), the file will be downloaded to a temporary directory.
toolkit_dir	‘character(1)‘ Path to the directory containing the ‘vrs_anvil_toolkit/1000g‘ sub-directory. If ‘NULL‘ (default), the toolkit location is determined by the ‘AnVIL-VRS.toolkit_path‘ option or cloned to a user data directory using ‘setup_vrs_toolkit()‘.

Value

a ‘list()‘ response including ‘$focusAlleleFrequency‘

Examples

```
library(reticulate)
## OR use full path to vrs_env
use_virtualenv("vrs_env", required = TRUE)
toolkit_dir <- setup_vrs_toolkit()
vcf <- AnVILVRS:::.get_fixture_vcf()
vcf_index <- build_vrs_index()
variant_id <- "chr1-20094-TAA-T"
vrs_id <- get_vrs_id(variant_id, "gnomad")
pop_desc <- get_pop_descriptor()
get_caf(
  vrs_id, vcf, vcf_index, "USA",
  pop_desc_file = pop_desc
)
```

---

get_pop_descriptor *Downloads the population descriptor file to the BiocFileCache from a known Google Storage URI.*

Description

Downloads the population descriptor file to the BiocFileCache from a known Google Storage URI.

Usage

```
get_pop_descriptor(uri = .POP_DESC_URI, force = FALSE, ...)
```

Arguments

uri	‘character(1)‘ The URI pointing to a Google Storage location where the ‘population_descriptor.tsv‘ file is hosted.
force	‘logical(1)‘ Whether to force re-download the file even if it is already present in the ‘BiocFileCache‘. Defaults to ‘FALSE‘. If ‘TRUE‘, the file will be re-downloaded.
...	Additional arguments passed to ‘AnVILGCP::avcopy()‘.

Value

‘character(1)‘ The local file path to the downloaded population descriptor file

Examples

```
get_pop_descriptor()
```

---

get_seqrepo *Run rsync on SeqRepo snapshot*

Description

The function will 'rsync' the indicated SeqRepo snapshot using the ‘biocommons.seqrepo‘ Python package. The AnVIL VRS Toolkit relies on local access to the reference sequences provided by SeqRepo. Note that the 'rsync' operation will require significant amounts of disk space and may take a considerable amount of time to complete, depending on the size of the snapshot being downloaded and the speed of your internet connection.

Usage

```
get_seqrepo(destdir = "./", snapshot = "2024-12-20", ..., uri)
```

Arguments

destdir	‘character(1)‘ The local directory where the file should be downloaded. Defaults to the current working directory.
snapshot	‘character(1)‘ The snapshot date of the SeqRepo database to download. Defaults to "2024-12-20". The snapshot date should correspond to a valid snapshot available in the SeqRepo database. You can check available snapshots at the SeqRepo repository or documentation.
...	Additional arguments passed to ‘AnVILGCP::avcopy()‘. Currently ignored as the function uses ‘rsync‘ to download the SeqRepo data instead of ‘avcopy()‘.
uri	‘character(1)‘ DEPRECATED The URI pointing to a Google Storage location where the ‘seqrepo.tar.gz‘ file is hosted.

Details

The function checks for the presence of ‘rsync‘ in the system's ‘PATH‘ and will throw an error if it is not found. If ‘rsync‘ is available, the function will execute the ‘seqrepo pull‘ command to download the reference data to the specified destination directory. The function returns the path to the local directory where the SeqRepo data has been downloaded.

Value

‘character(1)‘ The local directory path where the SeqRepo data has been downloaded.

See Also

<https://dl.biocommons.org/seqrepo/>

Examples

```
get_seqrepo(destdir = tempdir())
```

`get_vrs_id` *Translate a Variant to a VRS ID*

Description

This function uses a Python backend to translate a variant identifier (e.g., gnomad format) into a GA4GH VRS Allele ID.

Usage

```
get_vrs_id(variant_id, from_format = c("gnomad", "spdi", "hgvs", "beacon"))

get_vrs_allele(variant_id, from_format = c("gnomad", "spdi", "hgvs", "beacon"))

get_variant_from_allele(
  allele,
  to_format = c("gnomad", "spdi", "hgvs", "beacon")
)
```

Arguments

`variant_id`	‘character(1)‘ A string of the variant to translate.
`from_format, to_format`	‘character(1)‘ One of "gnomad", "spdi", "hgvs", or "beacon" formats indicating the format of the input ‘variant_id‘. The abbreviations stand for Genome Aggregation Database, Sequence Position Deletion Insertion, Human Genome Variation Society, and Beacon nomenclature, respectively. the respective variant representation. Defaults to "gnomad".
`allele`	A GA4GH VRS ‘Allele‘ object returned by ‘get_vrs_allele()‘

Value

A character string containing the VRS ID.

Examples

```
library(reticulate)
use_virtualenv("vrs_env", required = TRUE)

get_vrs_id("chr7-87509329-A-G", "gnomad")
get_vrs_id("NC_000005.10:80656509:C:TT", "spdi")
get_vrs_id("NC_000005.10:g.80656510delinsTT", "hgvs")
get_vrs_id("5:80656489C>T", "beacon")

get_vrs_allele("NC_000005.10:g.80656510delinsTT", "hgvs")
get_vrs_allele("5-80656489-C-T", "gnomad")

allele <- get_vrs_allele("5:80656489C>T", "beacon")
get_variant_from_allele(allele, "hgvs")

allele <- get_vrs_allele("NC_000005.10:g.80656510delinsTT", "hgvs")
get_variant_from_allele(allele, "spdi")
```

---

install_AnVILVRS *Install AnVILVRS Python Environment*

Description

Install AnVILVRS Python Environment

Usage

```
install_AnVILVRS(envname = "vrs_env", force = FALSE)
```

Arguments

envname	‘character(1)‘ virtual environment in which to install the python zarr module.
force	‘logical(1)‘ force re-installation of AnVILVRS requirements

Value

Reference to the python module, invisibly.

Examples

```
library(reticulate)
has_vrs_env <- virtualenv_exists("vrs_env")
if (!has_vrs_env)
    install_AnVILVRS(envname = "vrs_env")
```

`setup_vrs_toolkit` *Set up the VRS AnVIL Toolkit repository*

Description

The ‘gks-anvil/vrs_anvil_toolkit‘ GitHub repository is cloned to a local data directory if it does not already exist. If the directory exists and ‘update = TRUE‘, the function will pull the latest changes from the remote repository. The function returns the local path to the toolkit repository. Note that ‘gert‘ is used to clone the repository into the local directory.

Usage

```
setup_vrs_toolkit(destdir = NULL, update = FALSE)
```

Arguments

`destdir`	‘character(1)‘ The directory to store the toolkit. Defaults to the location specified by the ‘AnVILVRS.toolkit_path‘ option or a persistent user data directory.
`update`	‘logical(1)‘ Whether to pull the latest changes if the directory already exists.

Details

The repository location is determined in order of priority: 1. The ‘destdir‘ argument (if provided). 2. The ‘AnVILVRS.toolkit_path‘ global option (set via ‘options(AnVILVRS.toolkit_path = "...")‘). 3. A persistent user data directory (via ‘tools::R_user_dir("AnVILVRS")‘).

Value

‘character(1)‘ The local path to the toolkit repository

Examples

```
setup_vrs_toolkit(update = TRUE)
```

Index

∗ **internal**
AnVILVRS-package, 2

AnVILVRS *(*AnVILVRS-package*)*, 2
AnVILVRS-package, 2

build_vrs_index, 2

get_caf, 3
get_pop_descriptor, 4
get_seqrepo, 5
get_variant_from_allele *(*get_vrs_id*)*, 6
get_vrs_allele *(*get_vrs_id*)*, 6
get_vrs_id, 6

install_AnVILVRS, 7

setup_vrs_toolkit, 8