Package: amregtest (via r-universe)

September 11, 2024

Title Runs Allelematch Regression Tests

Version 1.0.3

Description Automates regression testing of package 'allelematch'.
Over 2500 tests covers all functions in 'allelematch',
reproduces the examples from the documentation and includes
negative tests. The implementation is based on 'testthat'.

License MIT + file LICENSE

Depends R (>= 2.10)

Imports allelematch, digest, remotes, testthat (>= 3.0.0), utils,
withr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://torstax.r-universe.dev

RemoteUrl https://github.com/torstax/amregtest

RemoteRef HEAD

RemoteSha 80721c63cb3de9f838e0ce4af8c88ee35abaf2b4

Contents

`amExample1` *Example 1 High quality data set*

Description

This is sample data copied from allelematch::amExampleData in version 5.2.1 of package allelematch. We use this data to test allelematch backwards compatibility.

Format

Data frame with samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

The data in this example is simulated to represent a high quality data set that might result from a laboratory protocol where samples were run multiple times to confirm their identity. It has no genotyping error, a near-zero missing data load, and approximately 60% of the individuals have been artificially resampled more than once.

References

`https://github.com/cran/allelematch`

`amExample2` *Example 2 Good quality data set*

Description

This is sample data copied from allelematch::amExampleData in version 5.2.1 of package allelematch. We use this data to test allelematch backwards compatibility.

Format

Data frame with samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

The data in this example have also been simulated, this time to reflect the qualities of good quality data set, where genotyping error and missing data exist, but these can be confidently handled by allelematch without manual intervention. At each locus a random 4% of heterozygotes lost their second allele to simulate an allele dropout, and a random 4% of samples at each locus had alleles set to missing.

References

`https://github.com/cran/allelematch`

`amExample3` *Example 3 Marginal quality data set*

Description

This is sample data copied from allelematch::amExampleData in version 5.2.1 of package allelematch. We use this data to test allelematch backwards compatibility.

Format

Data frame with samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

The data in this example have been simulated to represent a data set of marginal quality where the use of allelematch combined with careful manual review of the results is required to achieve a confident assessment of the unique genotypes. At each locus a random 4% of heterozygotes lost their second allele to simulate an allele dropout, and a random 10% of samples at each locus had alleles set to missing.

References

`https://github.com/cran/allelematch`

`amExample4` *Example 4 Low quality data set*

Description

This is sample data copied from allelematch::amExampleData in version 5.2.1 of package allelematch. We use this data to test allelematch backwards compatibility.

Format

Data frame with samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

For this example we have simulated a low quality data set where uncertainty created by genotyping error and missing data, combined with a lack of information in the form of allelic diversity across loci will result in a low confidence assessment of the unique genotypes. At each locus a random 6% of heterozygotes lost their second allele to simulate an allele dropout, and a random 20% of samples at each locus had alleles set to missing.

References

https://github.com/cran/allelematch

`amExample5` *Example 5 Wildlife data set*

Description

This is sample data copied from allelematch::amExampleData in version 5.2.1 of package allelematch. We use this data to test allelematch backwards compatibility.

Format

Data frame with samples in rows, and alleles in columns. Missing data is represented as "-99".

Details

In this final example we use real data from the non-invasive sampling of a wildlife population. The data have been anonymized by changing sampling details. A single column giving the gender is also available and we show how this can be used as an extra locus. Missing data is also more common at some loci than at others, with a total load of about 10%.

References

https://github.com/cran/allelematch

`amregtest` *Package Overview*

Description

Package 'amregtest' automates regression testing of package allelematch.

The API is simple. There are only three functions:

artRun	Executes the test, or a subset of the tests
artList	Lists the available tests without running them
artVersion	Shows the installed versions of allelematch and amregtest

The prefix "art" is short for "Allelematch Regression Test".

See artData for a description of data sets used as input.

References

amregtest-package

https://github.com/cran/allelematch

allelematchSuppDoc.pdf

`artData` *Example data used by amregtest*

Description

This example data is used when testing allelematch backwards compatibility using artRun. The tests load this data and passes it to amDataset.

It includes data that was imported from version 5.2.1 of allelematch. It was still unchanged in 5.2.4.

amExample1	Example 1 High quality data set
amExample2	Example 2 Good quality data set
amExample3	Example 3 Marginal quality data set
amExample4	Example 4 Low quality data set
amExample5	Example 5 Wildlife data set

See allelematchSuppDoc.pdf for a more detailed description.

It also includes a large data set gathered from field work:

ggSample	Very large wildlife data set

Format

Data frames with varying numbers of samples in rows, and alleles in columns. Missing data is represented as "-99".

References

`https://github.com/cran/allelematch`

allelematchSuppDoc.pdf

`artList` *Lists available tests in* `amregtest` *without running them*

Description

Use the output to select a value for parameter `filter` to artRun. Useful when debugging.

Usage

```
artList(verbose = TRUE)
```

Arguments

`verbose`	logical. If TRUE, prints additional info to stdout

Value

A character vector containing the names of all the tests

See Also

artVersion and artRun

Examples

```
# See what version of packages 'allelematch' and 'amregtest'
# are currently installed:
artVersion()

# List the available tests:
artList()

# Run all the tests:
# artRun()  # Takes several minutes

# Run the first of the available tests:
artRun(filter="allelematch_1-amDataset$")
```

artRun — *Runs the regression test*

Description

Runs allelematch regression tests to make sure it is backwards compatible.

The full set of tests will take a couple of minutes.

Call artList to see the available tests with without running them.

Usage

```
artRun(filter = "", verbose = TRUE)
```

Arguments

`filter`	If specified, only tests with names matching this perl regular expression will be executed. Character vector of length 1. See also artList
`verbose`	logical. If TRUE, prints version of tested allelematch to stdout

Details

If any of the test executed with artRun should fail, then we want to be able to run that specific test under the debugger. Character vector of length one.

Set a breakpoint in `allelematch.R` and call `artRun(filter="<the test that reproduces the problem>")`

Note that it is the last installed version of `allelematch` that will be executed, not the last edited. In RStudio, CTRL+SHIFT+B will build and install.

Value

A list (invisibly) containing data about the test results as returned by testthat::test_package

See Also

artVersion and artList

Examples

```
# See what version of packages 'allelematch' and 'amregtest'
# are currently installed:
artVersion()

# List the available tests:
artList()
```

```
# Run all the tests:
# artRun()  # Takes several minutes

# Run the first of the available tests:
artRun(filter="allelematch_1-amDataset$")
```

artVersion *Returns package version*

Description

Returns version of this package (amregtest).

The version is specified in the file DESCRIPTION, tag "Version: ".

Usage

```
artVersion(verbose = TRUE)
```

Arguments

verbose	logical. If TRUE, prints additional info to stdout, including version of allelematch-package

Value

The installed version of this package (amregtest-package) in a character vector of length one

See Also

artList, artRun and amregtest

Examples

```
# See what version of packages 'allelematch' and 'amregtest'
# are currently installed:
artVersion()

# List the available tests:
artList()

# Run all the tests:
# artRun()  # Takes several minutes

# Run the first of the available tests:
artRun(filter="allelematch_1-amDataset$")
```

Index