scikit-allel Documentation

Release 1.0.3

Alistair Miles

November 01, 2016

Contents

This package provides utilities for exploratory analysis of large scale genetic variation data. It is based on numpy, scipy and other established Python scientific libraries.

- Source: https://github.com/cggh/scikit-allel
- Documentation: http://scikit-allel.readthedocs.org/
- Download: https://pypi.python.org/pypi/scikit-allel
- Gitter: https://gitter.im/cggh/pygenomics

Please feel free to ask questions via cggh/pygenomics on Gitter. Release announcements are posted to the cggh/pygenomics Gitter channel and the biovalidation mailing list. If you find a bug or would like to suggest a feature, please raise an issue on GitHub.

This site provides reference documentation for *scikit-allel*. For worked examples with real data, see the following articles:

- Introducing scikit-allel
- A tour of scikit-allel
-
- Fast PCA

If you would like to cite *scikit-allel* please use the DOI below.

CHAPTER 1

Why “scikit-allel”?

“SciKits” (short for SciPy Toolkits) are add-on packages for SciPy, hosted and developed separately and independently from the main SciPy distribution.

“Allel” (Greek $\lambda\lambda\lambda$) is the root of the word “allele” short for “allelomorph”, a word coined by William Bateson to mean variant forms of a gene. Today we use “allele” to mean any of the variant forms found at a site of genetic variation, such as the different nucleotides observed at a single nucleotide polymorphism (SNP).

CHAPTER 2

Installation

Install pre-built binaries via conda:

```
$ conda install -c conda-forge scikit-allel
```

Install and compile source code via pip:

```
$ pip install scikit-allel
```

N.B., this package requires numpy, scipy, matplotlib, seaborn, pandas, scikit-learn, h5py, numexpr, bcolz, zarr and dask. If installing via conda, these should be installed automatically. If installing via pip, please install these dependencies first, then use pip to install scikit-allel.

CHAPTER 3

Contents

3.1 Data structures

3.1.1 In-memory data structures

GenotypeArray

GenotypeVector

Genotypes

Methods available on both `GenotypeArray` and `GenotypeVector` classes:

HaplotypeArray

AlleleCountsArray

GenotypeAlleleCountsArray

GenotypeAlleleCountsVector

GenotypeAlleleCounts

Methods available on both `GenotypeAlleleCountsArray` and `GenotypeAlleleCountsVector` classes:

VariantTable

FeatureTable

SortedIndex

SortedMultiIndex

UniqueIndex

3.1.2 Chunked arrays

GenotypeChunkedArray

HaplotypeChunkedArray

AlleleCountsChunkedArray

VariantChunkedTable

FeatureChunkedTable

AlleleCountsChunkedTable

3.1.3 Dask arrays

GenotypeDaskArray

HaplotypeDaskArray

AlleleCountsDaskArray

3.1.4 Utility functions

3.2 Statistics and plotting

3.2.1 Diversity & divergence

3.2.2 F-statistics

3.2.3 Hardy-Weinberg equilibrium

3.2.4 Linkage disequilibrium

3.2.5 Site frequency spectra

3.2.6 Pairwise distance and ordination

3.2.7 Principal components analysis

3.2.8 Admixture

3.2.9 Selection

3.2.10 Mendelian inheritance

backwards-compatible but in some cases could break existing code, hence the major version number has been incremented. Also included in this release are some new functions related to Mendelian inheritance and calling runs of homozygosity, further details below.

Mendelian errors and phasing by transmission

This release includes a new `allel.stats.mendel` module with functions to help with analysis of related individuals. The function `allel.stats.mendel.mendel_errors()` locates genotype calls within a trio or cross that are not consistent with Mendelian segregation of alleles. The function `allel.stats.mendel.phase_by_transmission()` will resolve unphased diploid genotypes into phased haplotypes for a trio or cross using Mendelian transmission rules. The function `allel.stats.mendel.paint_transmission()` can help with evaluating and visualizing the results of phasing a trio or cross.

Runs of homozygosity

A new `allel.stats.roh.roh_mhmm()` function provides support for locating long runs of homozygosity within a single sample. The function uses a multinomial hidden Markov model to predict runs of homozygosity based on the rate of heterozygosity over the genome. The function can also incorporate information about which positions in the genome are not accessible to variant calling and hence where there is no information about heterozygosity, to reduce false calling of ROH in regions where there is patchy data. We've run this on data from the Ag1000G project but have not performed a comprehensive evaluation with other species, feedback is very welcome.

Changes to data structures

The `allel.model.ndarray` module includes a new `allel.model.ndarray.GenotypeVector` class. This class represents an array of genotype calls for a single variant in multiple samples, or for a single sample at multiple variants. This class makes it easier, for example, to locate all variants which are heterozygous in a single sample.

Also in the same module are two new classes `allel.model.ndarray.GenotypeAlleleCountsArray` and `allel.model.ndarray.GenotypeAlleleCountsVector`. These classes provide support for an alternative encoding of genotype calls, where each call is stored as the counts of each allele observed. This allows encoding of genotype calls where samples may have different ploidy for a given chromosome (e.g., *Leishmania*) and/or where samples carry structural variation within some genome regions, altering copy number (and hence effective ploidy) with respect to the reference sequence.

There have also been architectural changes to all data structures modules. The most important change is that all classes in the `allel.model.ndarray` module now **wrap** numpy arrays and are no longer direct sub-classes of the numpy `numpy.ndarray` class. These classes still **behave** like numpy arrays in most respects, and so in most cases this change should not impact existing code. If you need a plain numpy array for any reason you can always use `numpy.asarray()` or access the `.values` property, e.g.:

```
>>> import allel
>>> import numpy as np
>>> g = allel.GenotypeArray([[[0, 1], [0, 0]], [[0, 2], [1, 1]]])
>>> isinstance(g, np.ndarray)
False
>>> a = np.asarray(g)
>>> isinstance(a, np.ndarray)
True
>>> isinstance(g.values, np.ndarray)
True
```

This change was made because there are a number of complexities that arise when sub-classing class:*numpy.ndarray* and these were proving tricky to manage and maintain.

The `allel.model.chunked` and `allel.model.dask` modules also follow the same wrapper pattern. For the `allel.model.dask` module this means a change in the way that classes are instantiated. For example, to create a `allel.model.dask.GenotypeDaskArray`, pass the underlying data directly into the class constructor, e.g.:

```
>>> import allel
>>> import h5py
>>> h5f = h5py.File('callset.h5', mode='r')
>>> h5d = h5f['3R/calldata/genotype']
>>> genotypes = allel.GenotypeDaskArray(h5d)
```

If the underlying data is chunked then there is no need to specify the chunks manually when instantiating a dask array, the native chunk shape will be used.

Finally, the *allel.model.bcolz* module has been removed, use either the `allel.model.chunked` or `allel.model.dask` module instead.

3.6.2 v0.21.2

This release resolves compatibility issues with Zarr version 2.1.

3.6.3 v0.21.1

- Added parameter *min_maf* to `allel.stats.selection.ihs()` to skip IHS calculation for variants below a given minor allele frequency.
- Minor change to calculation of integrated haplotype homozygosity to enable values to be reported for first and last variants if *include_edges* is *True*.
- Minor change to `allel.stats.selection.standardize_by_allele_count()` to better handle missing values.

3.6.4 v0.21.0

In this release the implementations of `allel.stats.selection.ihs()` and `allel.stats.selection.xpehh()` selection statistics have been reworked to address a number of issues:

- Both functions can now integrate over either a genetic map (via the *map_pos* parameter) or a physical map.
- Both functions now accept *max_gap* and *gap_scale* parameters to perform adjustments to integrated haplotype homozygosity where there are large gaps between variants, following the standard approach. Alternatively, if a map of genome accessibility is available, it may be provided via the *is_accessible* parameter, in which case the distance between variants will be scaled by the fraction of accessible bases between them.
- Both functions are now faster and can make use of multiple threads to further accelerate computation.
- Several bugs in the previous implementations of these functions have been fixed (#91).
- New utility functions are provided for standardising selection scores, see `allel.stats.selection.standardize_by_allele_count()` (for use with IHS and NSL) and `allel.stats.selection.standardize()` (for use with XPEHH).

Other changes:

- Added functions `allel.stats.diversity.moving_tajima_d()` and `allel.stats.selection.moving_delta_tajima_d()` (#81, #70).
- Added functions `allel.stats.fst.moving_weir_cockerham_fst()`, `allel.stats.fst.moving_hudson_fst()`, `allel.stats.fst.moving_patterson_fst()`.
- Added functions `allel.stats.admixture.moving_patterson_f3()` and `allel.stats.admixture.moving_patterson_d()`.
- Renamed "blockwise" to "average" in function names in `allel.stats.fst` and `allel.stats.admixture` for clarity.
- Added convenience methods `allel.model.ndarray.AlleleCountsArray.is_biallelic()` and `allel.model.ndarray.AlleleCountsArray.is_biallelic_01()` for locating biallelic variants.
- Added support for zarr in the `allel.chunked` module (#101).
- Changed HDF5 default chunked storage to use gzip level 1 compression instead of no compression (#100).
- Fixed bug in `allel.stats.diversity.sequence_divergence()` (#75).
- Added workaround for chunked arrays if passed as arguments into numpy aggregation functions (#66).
- Protect against invalid coordinates when mapping from square to condensed coords (#83).
- Fixed bug in `allel.stats.sf.plot_sfs_folded()` and added docstrings for all plotting functions in `allel.stats.sf` (#80).
- Fixed bug related to taking views of genotype and haplotype arrays (#77).

3.6.5 v0.20.3

- Fixed a bug in the *count_alleles()* methods on genotype and haplotype array classes that manifested if the *max_allele* argument was provided (#59).
- Fixed a bug in Jupyter notebook *display* method for chunked tables (#57).
- Fixed a bug in site frequency spectrum scaling functions (#54).
- Changed behaviour of *subset* method on genotype and haplotype arrays to better infer argument types and handle None argument values (#55).
- Changed table *eval* and *query* methods to make python the default for expression evaluation, because it is more expressive than numexpr (#58).

3.6.6 v0.20.2

- Changed `allel.util.hdf5_cache()` to resolve issues with hashing and argument order (#51, #52).

3.6.7 v0.20.1

- Changed functions `allel.stats.fst.weir_cockerham_fst()` and `allel.stats.ld.locate_unlinked()` such that chunked implementations are now used by default, to avoid accidentally and unnecessarily loading very large arrays into memory (#50).

3.6.8 v0.20.0

- Added new `allel.model.dask` module, providing implementations of the genotype, haplotype and allele counts classes backed by dask.array (#32).
- Released the GIL where possible in Cython optimised functions (#43).
- Changed functions in `allel.stats.selection` that accept *min_ehh* argument, such that *min_ehh* = *None* should now be used to indicate that no minimum EHH threshold should be applied.

3.6.9 v0.19.0

The major change in v0.19.0 is the addition of the new `allel.model.chunked` module, which provides classes for variant call data backed by chunked array storage (#31). This is a generalisation of the previously available `allel.model.bcolz` to enable the use of both bcolz and HDF5 (via h5py) as backing storage. The `allel.model.bcolz` module is now deprecated but will be retained for backwargs compatibility until the next major release.

Other changes:

- Added function for computing the number of segregating sites by length (nSl), a summary statistic comparing haplotype homozygosity between different alleles (similar to IHS), see `allel.stats.selection.nsl()` (#40).
- Added functions for computing haplotype diversity, see `allel.stats.selection.haplotype_diversity()` and `allel.stats.selection.moving_haplotype_diversity()` (#29).
- Added function `allel.stats.selection.plot_moving_haplotype_frequencies()` for visualising haplotype frequency spectra in moving windows over the genome (#30).
- Added *vstack()* and *hstack()* methods to genotype and haplotype arrays to enable combining data from multiple arrays (#21).
- Added convenience function `allel.stats.window.equally_accessible_windows()` (#16).
- Added methods *from_hdf5_group()* and *to_hdf5_group()* to `allel.model.ndarray.VariantTable` (#26).
- Added `allel.util.hdf5_cache()` utility function.
- Modified functions in the `allel.stats.selection` module that depend on calculation of integrated haplotype homozygosity to return NaN when haplotypes do not decay below a specified threshold (#39).
- Fixed missing return value in `allel.stats.selection.plot_voight_painting()` (#23).
- Fixed return type from array reshape() (#34).

Contributors: alimanfoo, hardingnj

3.6.10 v0.18.1

- Minor change to the Garud H statistics to avoid raising an exception when the number of distinct haplotypes is very low (#20).

3.6.11 v0.18.0

- Added functions for computing H statistics for detecting signatures of soft sweeps, see `allel.stats.selection.garud_h()`, `allel.stats.selection.moving_garud_h()`, `allel.stats.selection.plot_haplotype_frequencies()` (#19).
- Added function `allel.stats.selection.fig_voight_painting()` to paint both flanks either side of some variant under selection in a single figure (#17).
- Changed return values from `allel.stats.selection.voight_painting()` to also return the indices used for sorting haplotypes by prefix (#18).

3.6.12 v0.17.0

- Added new module for computing and plotting site frequency spectra, see `allel.stats.sf` (#12).
- All plotting functions have been moved into the appropriate stats module that they naturally correspond to. The `allel.plot` module is deprecated (#13).
- Improved performance of carray and ctable loading from HDF5 with a condition (#11).

3.6.13 v0.16.2

- Fixed behaviour of take() method on compressed arrays when indices are not in increasing order (#6).
- Minor change to scaler argument to PCA functions in `allel.stats.decomposition` to avoid confusion about when to fall back to default scaler (#7).

3.6.14 v0.16.1

- Added block-wise implementation to `allel.stats.ld.locate_unlinked()` so it can be used with compressed arrays as input.

3.6.15 v0.16.0

- Added new selection module with functions for haplotype-based analyses of recent selection, see `allel.stats.selection`.

3.6.16 v0.15.2

- Improved performance of `allel.model.bcolz.carray_block_compress()`, `allel.model.bcolz.ctable_block_compress()` and `allel.model.bcolz.carray_block_subset()` for very sparse selections.
- Fix bug in IPython HTML table captions.
- Fix bug in addcol() method on bcolz ctable wrappers.

3.6.17 v0.15.1

- Fix missing package in setup.py.

3.6.18 v0.15

- Added functions to estimate Fst with standard error via a block-jackknife: `allel.stats.fst.blockwise_weir_cockerham_fst()`, `allel.stats.fst.blockwise_hudson_fst()`,`allel.stats.fst.blockwise_patterson_fst()`.
- Fixed a serious bug in `allel.stats.fst.weir_cockerham_fst()` related to incorrect estimation of heterozygosity, which manifested if the subpopulations being compared were not a partition of the total population (i.e., there were one or more samples in the genotype array that were not included in the subpopulations to compare).
- Added method `allel.model.AlleleCountsArray.max_allele()` to determine highest allele index for each variant.
- Changed first return value from admixture functions `allel.stats.admixture.blockwise_patterson_f3()` and `allel.stats.admixture.blockwise_patterson_d()` to return the estimator from the whole dataset.
- Added utility functions to the `allel.stats.distance` module for transforming coordinates between condensed and uncondensed forms of a distance matrix.
- Classes previously available from the *allel.model* and *allel.bcolz* modules are now aliased from the root `allel` module for convenience. These modules have been reorganised into an `allel.model` package with submodules `allel.model.ndarray` and `allel.model.bcolz`.
- All functions in the `allel.model.bcolz` module use cparams from input carray as default for output carray (convenient if you, e.g., want to use zlib level 1 throughout).
- All classes in the `allel.model.ndarray` and `allel.model.bcolz` modules have changed the default value for the *copy* keyword argument to *False*. This means that **not** copying the input data, just wrapping it, is now the default behaviour.
- Fixed bug in `GenotypeArray.to_gt()` where maximum allele index is zero.

3.6.19 v0.14

- Added a new module `allel.stats.admixture` with statistical tests for admixture between populations, implementing the f2, f3 and D statistics from Patterson (2012). Functions include `allel.stats.admixture.blockwise_patterson_f3()` and `allel.stats.admixture.blockwise_patterson_d()` which compute the f3 and D statistics respectively in blocks of a given number of variants and perform a block-jackknife to estimate the standard error.

3.6.20 v0.12

- Added functions for principal components analysis of genotype data. Functions in the new module `allel.stats.decomposition` include `allel.stats.decomposition.pca()` to perform a PCA via full singular value decomposition, and `allel.stats.decomposition.randomized_pca()` which uses an approximate truncated singular value decomposition to speed up computation. In tests with real data the randomized PCA is around 5 times faster and uses half as much memory as the conventional PCA, producing highly similar results.
- Added function `allel.stats.distance.pcoa()` for principal coordinate analysis (a.k.a. classical multi-dimensional scaling) of a distance matrix.
- Added new utility module `allel.stats.preprocessing` with classes for scaling genotype data prior to use as input for PCA or PCoA. By default the scaling (i.e., normalization) of Patterson (2006) is used with

principal components analysis functions in the `allel.stats.decomposition` module. Scaling functions can improve the ability to resolve population structure via PCA or PCoA.

- Added method `allel.model.GenotypeArray.to_n_ref()`. Also added `dtype` argument to `allel.model.GenotypeArray.to_n_ref()` and `allel.model.GenotypeArray.to_n_alt()` methods to enable direct output as float arrays, which can be convenient if these arrays are then going to be scaled for use in PCA or PCoA.
- Added `allel.model.GenotypeArray.mask` property which can be set with a Boolean mask to filter genotype calls from genotype and allele counting operations. A similar property is available on the `allel.bcolz.GenotypeCArray` class. Also added method `allel.model.GenotypeArray.fill_masked()` and similar method on the `allel.bcolz.GenotypeCArray` class to fill masked genotype calls with a value (e.g., -1).

3.6.21 v0.11

- Added functions for calculating Watterson's theta (proportional to the number of segregating variants): `allel.stats.diversity.watterson_theta()` for calculating over a given region, and `allel.stats.diversity.windowed_watterson_theta()` for calculating in windows over a chromosome/contig.
- Added functions for calculating Tajima's D statistic (balance between nucleotide diversity and number of segregating sites): `allel.stats.diversity.tajima_d()` for calculating over a given region and `allel.stats.diversity.windowed_tajima_d()` for calculating in windows over a chromosome/contig.
- Added `allel.stats.diversity.windowed_df()` for calculating the rate of fixed differences between two populations.
- Added function `allel.model.locate_fixed_differences()` for locating variants that are fixed for different alleles in two different populations.
- Added function `allel.model.locate_private_alleles()` for locating alleles and variants that are private to a single population.

3.6.22 v0.10

- Added functions implementing the Weir and Cockerham (1984) estimators for F-statistics: `allel.stats.fst.weir_cockerham_fst()` and `allel.stats.fst.windowed_weir_cockerham_fst()`.
- Added functions implementing the Hudson (1992) estimator for Fst: `allel.stats.fst.hudson_fst()` and `allel.stats.fst.windowed_hudson_fst()`.
- Added new module `allel.stats.ld` with functions for calculating linkage disequilibrium estimators, including `allel.stats.ld.rogers_huff_r()` for pairwise variant LD calculation, `allel.stats.ld.windowed_r_squared()` for windowed LD calculations, and `allel.stats.ld.locate_unlinked()` for locating variants in approximate linkage equilibrium.
- Added function `allel.plot.pairwise_ld()` for visualising a matrix of linkage disequilbrium values between pairs of variants.
- Added function `allel.model.create_allele_mapping()` for creating a mapping of alleles into a different index system, i.e., if you want 0 and 1 to represent something other than REF and ALT, e.g., ancestral and derived. Also added methods `allel.model.GenotypeArray.map_alleles()`, `allel.model.HaplotypeArray.map_alleles()` and `allel.model.AlleleCountsArray.map_alleles()` which will perform an allele transformation given an allele mapping.

- Added function `allel.plot.variant_locator()` ported across from anhima.
- Refactored the `allel.stats` module into a package with sub-modules for easier maintenance.

3.6.23 v0.9

- Added documentation for the functions `allel.bcolz.carray_from_hdf5()`, `allel.bcolz.carray_to_hdf5()`, `allel.bcolz.ctable_from_hdf5_group()`, `allel.bcolz.ctable_to_hdf5_group()`.
- Refactoring of internals within the `allel.bcolz` module.

3.6.24 v0.8

- Added *subpop* argument to `allel.model.GenotypeArray.count_alleles()` and `allel.model.HaplotypeArray.count_alleles()` to enable count alleles within a sub-population without subsetting the array.
- Added functions `allel.model.GenotypeArray.count_alleles_subpops()` and `allel.model.HaplotypeArray.count_alleles_subpops()` to enable counting alleles in multiple sub-populations in a single pass over the array, without sub-setting.
- Added classes `allel.model.FeatureTable` and `allel.bcolz.FeatureCTable` for storing and querying data on genomic features (genes, etc.), with functions for parsing from a GFF3 file.
- Added convenience function `allel.stats.distance.pairwise_dxy()` for computing a distance matrix using Dxy as the metric.

3.6.25 v0.7

- Added function `allel.io.write_fasta()` for writing a nucleotide sequence stored as a NumPy array out to a FASTA format file.

3.6.26 v0.6

- Added method `allel.model.VariantTable.to_vcf()` for writing a variant table to a VCF format file.

CHAPTER 4

Acknowledgments

Development of this package is supported by the MRC Centre for Genomics and Global Health.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

a

A