Package: tidygenclust (via r-universe)

May 30, 2026

Title Clustering for Population Genetics in R

Version 0.1.1

Description A tidy interface to clustering in population genetics. This package provides a set of functions to perform clustering on genetic data, and to visualize the results, both for single runs and for multiple repeats of the same analysis. 'tidygenclust' ports the 'fastmixture' and 'clumppling' python modules to R, and it is built on top of the 'tidypopgen' package. Currently it works only on Linux and OSX (you can use the WSL on Windows).

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 3.5.0)

Imports dplyr, generics, ggplot2, patchwork, reticulate, rlang, tibble, tidypopgen, tidyr, utils

Suggests knitr, readr, rmarkdown, testthat (>= 3.0.0)

LazyData true

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libpng-dev libssl-dev libproj-dev python3 libsqlite3-dev libudunits2-dev zlib1g-dev

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

Date/Publication 2026-05-30 21:28:12 UTC

RemoteUrl https://github.com/EvolEcolGroup/tidygenclust

RemoteRef main

RemoteSha 36122fa6bac3b185f5234c659a823f002f0cbfc3

Contents

`autoplot.gt_clumppling` *autoplot for clumppling objects*

Description

An autoplot method to generate quick visualisations for `gt_clumppling` objects. Available types are:

- 'modes': all aligned modes in structure plots over a multipartite graph, where better alignment between the modes is indicated by the darker color of the edges connecting their structure plots, and the cost of optimal alignment is labelled on each edge.
- 'modes_within_K': A set of figures, one for each number of clusters, with all modes with the same number of clusters in structure plots in one figure.
- 'major_modes': the major modes of each K aligned in a series of structure plots.
- 'all_modes': all aligned modes in a series of structure plots.

Usage

```
## S3 method for class 'gt_clumppling'
autoplot(
  object,
  type = c("modes", "modes_within_k", "major_modes", "all_modes"),
  group = NULL,
  k = NULL,
  ...
)
```

Arguments

`object`	a `gt_clumppling` object
`type`	the type of plot, one of 'modes', 'modes_within_K', 'major_modes' or 'all_modes'.
`group`	a vector of membership to a-priori groups (e.g. populations). Note that individuals from the same group need to be adjacent to each other

k	the k value to be plotted if 'type' is 'modes_within_k'
...	not used at the moment

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

If you would like to generate an annotated autoplot, ensure that all individuals from the same population are adjacent to one another in the Q-matrix or gt_admix object supplied to gt_clumppling. Autoplot 'group' argument requires that all individuals from the same group are adjacent.

Value

a plot

---

capeverde_pops *Population membership for the Capeverde dataset*

Description

Membership to populations for a dataset of 399 individuals, including 44 Cape Verdean. This dataset is used as an example for clumppling.

Usage

```
capeverde_pops
```

Format

A vector of length 399

---

gt_clumppling *run clumppling*

Description

This function runs the clumppling algorithm.

Usage

```
gt_clumppling(
  input_path,
  output_path = tempfile("clump_out"),
  input_format = "admixture",
  use_rep = TRUE,
  merge = TRUE,
  cd_method = "louvain",
  use_best_pair = TRUE,
  extension = ".Q"
)
```

Arguments

input_path	the path where the Q files are stored, either a directory or a zip archive, or a q_matrix_list object
output_path	(optional) the clumppling functions in python save everything to file. By default, R stores the information in objects in the environment, and sends those files to a temporary directory that will be cleared at the end of a session. output_path allows to change the location of those files. This is only useful to those interested in recovering the same files as created by the python clumppling module, or for debugging.
input_format	a string defining the format of the input files, one of 'admixture' (default),'structure','fastStructure' or 'generalQ'
use_rep	boolean, whether to use representative modes (alternative: average), defaults to TRUE
merge	boolean, whether to merge two clusters when aligning K+1 to K, defaults to TRUE
cd_method	the community detection method to use, one of 'louvain' (default), 'leiden', 'infomap', 'markov_clustering', 'label_propagation', 'walktrap', 'custom'
use_best_pair	boolean, whether to use best pair as anchor for across-K alignment (alternative: major), defaults to TRUE
extension	(optional) if loading from files rather than a q_matrix_list object specify the extension e.g. ".Q" or ".indivq"

Details

If you would like to generate an annotated autoplot from your gt_clumppling object, ensure that all individuals from the same population are adjacent to one another in the Q-matrix or gt_admix object supplied to gt_clumppling. Autoplot 'group' argument requires that all individuals from the same group are adjacent.

Value

a list of class gt_clumppling containing:

- N: number of individuals

- K_range: vector of K values analyzed
- mode_replicates: a list of replicate indices for each mode
- cost_acrossK: a named list of costs for each pairwise K alignment
- aligned_modes: a list of data.frames, each data.frame is a Q-matrix

---

`gt_fastmixture` *fastmixture algorithm for population genetics clustering*

Description

This function implements the fastmixture algorithm for population genetics clustering by calling the python module. If you use this function, make sure that you cite the relevant paper by Santander, Refoyo-Martínez, and Meisner (2024).

Usage

```
gt_fastmixture(
  x,
  k,
  n_runs = 1,
  threads = 1,
  seed = 42,
  iter = 1000,
  tole = 1e-09,
  batches = 32,
  supervised = NULL,
  check = 5,
  power = 11,
  chunk = 8192,
  subsample = 0.7,
  min_subsample = 50000,
  max_subsample = 5e+05,
  als_iter = 1000,
  als_tole = 1e-04,
  no_freqs = TRUE,
  random_init = TRUE,
  safety = TRUE,
  cv = NULL,
  cv_tole = 1e-07
)
```

Arguments

x	either a `tidypopgen::gen_tibble`, or the name of the binary plink file (without the .bed extension)
k	the number of ancestral components (clusters), either a single value or a vector

n_runs	the number of repeats for each k value
threads	the number of threads to use (1)
seed	the random seed (defaults to 42);it should be a vector of length repeats
iter	the maximum number of iterations (1000)
tole	the tolerance in log-likelihood units between iterations (1e-9)
batches	the number of maximum mini-batches (32)
supervised	the name fo the file with the supervised labels (NULL)
check	the number of iterations to check for convergence (5)
power	number of power iterations in randomised SVD (11)
chunk	the number of SPs in chunk operations (8192)
subsample	Fraction of SNPs to subsample in SVD/ALS (0.7)
min_subsample	Minimum number of SNPs to subsample in SVD/ALS (50000)
max_subsample	Maximum number of SNPs to subsample in SVD/ALS (500000)
als_iter	the maximum number of iterations in the ALS algorithm (1000)
als_tole	the tolerance for the RMSE of P between iterations (1e-4)
no_freqs	do not save P-matrix (TRUE)
random_init	random initialisation of parameters (TRUE)
safety	add extra safety steps in unstable optimizations (TRUE)
cv	the number of cross-validation folds (0)
cv_tole	the tolerance for the cross-validation error in scaled log-likelihood units (1e-7)

Details

This function returns a q_matrix that can be plotted with autoplot, and tidied with tidy methods from the tidypopgen package. Cross-validation is set to 0 as default, if you want to include cross-validation you can set cv to a value greater than 1 (ADMIXTURE performs 5-fold cv as default).

Value

an object of class gt_admix. See tidypopgen::gt_admixture() for details.

References

C. G. Santander, A. Refoyo Martinez, J. Meisner (2024) Faster model-based estimation of ancestry proportions. bioRxiv 2024.07.08.602454; doi: https://doi.org/10.1101/2024.07.08.602454

subset_gt_clumppling *Subset a* gt_clumppling *object*

Description

This function subsets gt_clumppling objects to a set of individuals or a set of values of K. This is intended to create plot insets, or to visualise a subset of individuals during data analysis. To understand the modes within a subset of individuals in your data, you should subset your gt_admix object and re-run gt_clumppling.

Usage

```
subset_gt_clumppling(x, k = NULL, indivs = NULL)
```

Arguments

x	a gt_clumppling object
k	a vector of k values to subset to
indivs	a vector of individual indices to keep

Value

a gt_clumppling object subsetted to the individuals specified

tgc_tools_install *Install tools for* tidygenclust

Description

tidygenclust relies on ADMIXTURE and on python packages fastmixture and clumppling for a number of functionalities. We use reticulate to install them in conda environments. As their dependencies are incompatible, we use two separate conda environments, ctidygenclust (for fastmixture and admixture) and cclumppling (for clumppling). Additionally, for silicon Macs, ADMIXTURE is installed in a separate conda environment cadmixture86, as it is only available for OSX as x86 in bioconda.

Usage

```
tgc_tools_install(
  reset = FALSE,
  fastmixture_hash = "29e04339ce6ddf750ee4e06f8aabe40335e0d0ee",
  clumppling_hash = "2d24e0b2f6ddfcb51a436df96a06d5f57d18d20a",
  conda_method = c("reticulate", "conda_yaml"),
  ci_install = FALSE
)
```

Arguments

reset	a boolean used to reset the virtual environment. Only set to TRUE if you have a broken virtual environment that you want to reset.
fastmixture_hash	a string with the commit hash of the fastmixture version to install. Default is the latest tested version.
clumppling_hash	a string with the commit hash of the clumppling version to install. Default is the latest tested version.
conda_method	a string indicating the method to create the environment used for fastmixture. Default is reticulate, which uses the reticulate::conda_run2() function to run the installation commands (this is the default, and the only method for Linux. Alternatively, for OSX, you can use conda_yaml, which will create a conda environment directly with conda. Use this second method if "reticulate" fails whilst trying to install on OSX.
ci_install	a boolean indicating if the installation is being run on continuous integration (CI) services. Default is FALSE. If TRUE, the function will look for the conda yaml file in the inst/python folder of the package source directory, rather than in the installed package directory. This is useful when testing the package on CI services.

Details

For each tool, default to the latest tested version of these packages that have been tested to work with tidegenclust. It is possible to provide a more recent github commit for a specific tool, but this might lead to incompatibilities and errors.

We have found installation on OSX to be tricky, so we provide two methods for installing fastmixture on OSX: reticulate and conda_yaml. The reticulate method uses the reticulate::conda_run2() function to run installation commands, while the conda_yaml method creates a conda environment directly with conda. If the reticulate method fails, you can use the conda_yaml method to create the environment directly with conda. For OSX, you might also need to install a suitable compiler for openmp using brew in bash, setting the correct paths to use it:
brew install llvm libomp

---

tgc_tools_version *Print version of the python tools installed by* tidygenclust

---

Description

This function prints the version of the python tools installed by tidygenclust

Usage

```
tgc_tools_version()
```

Value

A list with the version of the python tools installed by `tidygenclust`

`tidy.gt_clumppling` *Tidy a* `gt_clumppling` *object*

Description

A `tidy` method to extract information from a `gt_clumppling` object, and return it as a tibble. It can extract:

- 'modes': all the modes detected by `gt_clumppling()`. The models have label 'KxMy', where 'x' and 'y' represent the K value and the mode rank.
- 'major_modes': modes of rank 1 for each K.
- 'Q_modes', 'q_modes': a list of q matrices, one per mode, each tidied into a tibble
- 'Q_major_modes', 'q_major_modes': the same output as 'Q_modes' but subsetted to only the major modes.

Usage

```
## S3 method for class 'gt_clumppling'
tidy(
  x,
  matrix = c("modes", "major_modes", "Q_modes", "q_modes", "Q_major_modes",
    "q_major_modes"),
  ...
)
```

Arguments

`x`	the `gt_clumppling` object
`matrix`	a string defining the information to be extracted, one of: "modes", "major_modes", "Q_modes", "Q_major_modes".
`...`	Additional arguments. Not used. Needed to match generic signature only.

Value

a tibble::tibble of the information of interest

Index