

Package: geneticumap (via r-universe)

October 13, 2026

Title Evolutionary Algorithm for UMAP-based Clustering Pipeline
Parameter Optimization for Single-Cell RNA-Seq Data

Version 0.99.3

Date 2026-10-13

Description The geneticumap package searches for single-cell clustering parameters by treating the Seurat preprocessing and clustering workflow as an optimization problem. The UMAP-based clustering pipeline parameters for single-cell RNA-seq data (like resolution, number of PCs, or neighbor counts) evolve using an evolutionary algorithm to systematically search for an optimal set of parameters for a given dataset. The most appropriate configuration is objectively identified by the Calinski-Harabasz index. The package is developed at the [Institute of Medical Systems Biology of Ulm University](https://sysbio.uni-ulm.de).

License MIT + file LICENSE

Depends R (>= 4.5.0)

Imports ecr, clusterCrit, Seurat, BBmisc, checkmate, SeuratObject, jsonlite, parallelMap, plotly, ggplot2, parallelly

Suggests hdf5r, testthat (>= 3.0.0), knitr, rmarkdown, htmlwidgets

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

VignetteBuilder knitr

biocViews GeneExpression, SingleCell, Transcriptomics, Clustering, DimensionReduction, Preprocessing

URL <https://github.com/sysbio-bioinf/geneticumap>

BugReports <https://github.com/sysbio-bioinf/geneticumap/issues>

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev libglu1-mesa-dev make libharfbuzz-dev texlive libicu-dev libpng-dev libuv1-dev libxml2-dev libgl1-mesa-dev libssl-dev python3 zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-10-13 10:35:45 UTC

RemoteUrl <https://github.com/BiocStaging/geneticumap>

RemoteRef HEAD

RemoteSha 717508293cf3311b3135a65e05e9e1cbcc6c0bb8

Contents

run_optimization	2
Index	7

run_optimization	<i>Optimizes UMAP-based clustering pipeline parameters for single-cell RNA-seq data using an evolutionary algorithm.</i>
------------------	--

Description

This function optimizes UMAP-based clustering pipeline parameters for single-cell RNA-seq data. Therefore, it treats the Seurat preprocessing and clustering workflow as an optimization problem, identifying the most appropriate configuration by the Calinski-Harabasz index. It evaluates clustering quality by measuring the ratio of between-cluster dispersion to within-cluster dispersion, given a set of data points in UMAP space and their cluster labels, which are derived from the PCA space.

Usage

```
run_optimization(  
  data_dir,  
  results_dir = file.path(data_dir, "results"),  
  gene_column = 1,  
  max_iterations = 10,  
  mu = 5,  
  lambda = 10,  
  survival_strategy = "plus",  
  probability_mutation = 1,  
  adapt_mutation_rate = 0.7,  
  scale_factor_min = 1000,  
  nfeatures_min = 500,  
  dimensionality_min = 5,  
  resolution_min = 0.1,  
  umap_neighbors_min = 5,  
  umap_min_dist_min = 0.01,  
  scale_factor_max = 50000,
```

```

nfeatures_max = 5000,
dimensionality_max = 50,
resolution_max = 2,
umap_neighbors_max = 50,
umap_min_dist_max = 0.7,
num_processes = min(lambda, parallelly::availableCores())
)

```

Arguments

data_dir	the directory with a .h5 file or 10X Genomics dataset (expected files: bar-codes.tsv, genes.tsv, matrix.mtx) to run the optimization with
results_dir	the directory to save the result files of the optimization, the directory is created if it does not exist
gene_column	the position of the gene names in the genes.tsv file on a 10x Genomics dataset
max_iterations	the maximum number of iterations of the evolution
mu	(greek letter μ) the number of individuals in the population (commonly referred to as the parent population size)
lambda	the number of individuals generated in each generation
survival_strategy	the survivor-selection strategy: "plus" (mu + lambda) selects the next population of mu individuals from the parents and offspring combined, so parents may survive; "comma" (mu, lambda) selects the next population only from the lambda offspring, so parents do not survive directly. For "comma", lambda must be at least mu.
probability_mutation	the probability that a mutation operator will be applied to a child - refers only to the application of the mutation operator, not to the probability of mutating individual genes of the respective child
adapt_mutation_rate	controls how fast mutation rates can evolve
scale_factor_min	the lower bound for the scale_factor parameter in the optimization
nfeatures_min	the lower bound for the nfeatures parameter in the optimization
dimensionality_min	the lower bound for the dimensionality parameter in the optimization
resolution_min	the lower bound for the resolution parameter in the optimization
umap_neighbors_min	the lower bound for the umap_neighbors parameter in the optimization
umap_min_dist_min	the lower bound for the umap_min_dist parameter in the optimization
scale_factor_max	the upper bound for the scale_factor parameter in the optimization
nfeatures_max	the upper bound for the nfeatures parameter in the optimization

dimensionality_max	the upper bound for the dimensionality parameter in the optimization
resolution_max	the upper bound for the resolution parameter in the optimization
umap_neighbors_max	the upper bound for the umap_neighbors parameter in the optimization
umap_min_dist_max	the upper bound for the umap_min_dist parameter in the optimization
num_processes	the number of processes to run the fitness evaluation with - default is the minimum of number of individuals and available cores

Details

The core idea is:

1. Load the 10X or h5 single-cell dataset.
2. Run a fixed Seurat workflow with a small set of tunable parameters.
3. Extract UMAP coordinates and cluster identities.
4. Score the resulting clustering with the Calinski-Harabasz index.
5. Use an evolutionary algorithm to explore the parameter space.
6. Save the search result and export reporting artifacts for downstream analysis.

Value

the optimized parameters of the evolution as named vector

Parameter Optimization

The optimization includes the following parameters:

- scale_factor
- nfeatures
- dimensionality
- resolution
- umap_neighbors
- umap_min_dist

Each parameter is optimized within predefined upper and lower bounds which can be set with the `[parameter]_min` and `[parameter]_max` function parameters. If `[parameter]_min = [parameter]_max`, the parameter does not evolve. Note that the default parameters are always evaluated nonetheless. The default parameter values are based on the [Seurat PBMC 3K tutorial](#). Although all six parameters are numerically encoded as floating-point values during optimization, some are interpreted as integers later: dimensionality is rounded before use and nfeatures and umap_neighbors for analysis views.

Evolution Configuration

The evolution and clustering behavior can be modified by the function parameters:

- which parameters are fixed and therefore excluded from exploration
- budget and population sizes for the evolutionary algorithm
- parent and survival selectors
- mutation regime The speed of mutation-rate adaptation is controlled by the `adapt_mutation_rate` parameter.

The **parent and survival selection** uses a tournament selection for mating and a greedy replacement for survival. The survival strategy itself is configurable with the `survival_strategy` parameter:

- plus means ($\mu + \lambda$) survival
- comma means (μ, λ) survival

The default is plus, meaning parents and offspring compete together for survival.

Dataset

The dataset to run the optimization with may be an h5 file or a 10X Genomics dataset. Both types are accepted by Seurat and read in by `Read10X(...)` / `Read10X_h5(...)` and `CreateSeuratObject(...)`. After loading of the dataset, quality control filtering is applied:

- mitochondrial percentage is computed with pattern `^MT-`
- cells are retained only if:
 - `nFeature_RNA > 200`
 - `nFeature_RNA < 2500`
 - `percent_mt < 5`

These QC filters are hard-coded and apply to all datasets, which is an important assumption of the project.

Seurat Pipeline Execution

For a candidate parameter set, the routine performs the following sequence:

1. Normalize counts using the candidate `scale_factor`
2. Select variable features using candidate `nfeatures`
3. Scale all genes
4. Run PCA
5. Build the neighbor graph using the first dimensionality PCs
6. Cluster cells using candidate `resolution`
7. Compute UMAP using candidate `umap_neighbors` and `umap_min_dist`

This is an important design decision: the search is not optimizing a custom embedding model directly. It is optimizing hyperparameters of a standard Seurat workflow, and all fitness is derived from the resulting UMAP-plus-clustering state.

Parallelization

The evolution uses `parallelMap`, at level `ecr.evaluateFitness` to **parallelize** the fitness evaluation for all individuals of the population. Therefore, each individual can be evaluated in a different parallelization unit. This is recommended because evaluating one individual means running a full Seurat pipeline, which is computationally expensive. The number of processes can be set with the `num_processes` function parameter. You can then disable parallelization by setting `num_processes = 1`.

Examples

```
run_optimization(data_dir = "/Users/myuser/Documents/mydata")
run_optimization(data_dir = "../mydata", results_dir = .. / mydata /
optimization - results, mu = 2, lambda = 4, max_iterations = 8)
run_optimization(data_dir = "../mydata", dimensionality_min = 6,
dimensionality_max = 6, num_processes = 1)
```

Index

`run_optimization`, [2](#)