Package 'Ibex'

November 3, 2025

Title Methods for BCR single-cell embedding

Version 1.0.0

Description Implementation of the Ibex algorithm for single-cell embedding based on BCR sequences. The package includes a standalone function to encode BCR sequence information by amino acid properties or sequence order using tensorflow-based autoencoder. In addition, the package interacts with SingleCellExperiment or Seurat data objects.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

biocViews Software, ImmunoOncology, SingleCell, Classification, Annotation, Sequencing

Depends R (>= 4.5.0)

Imports basilisk, immApex (>= 1.3.2), methods, Matrix, reticulate (>= 1.43.0), rlang, SeuratObject, scRepertoire, SingleCellExperiment, stats, SummarizedExperiment, tensorflow, tools

Suggests basilisk.utils, BiocStyle, bluster, dplyr, ggplot2, kableExtra, knitr, markdown, mumosa, patchwork, Peptides, rmarkdown, scater, spelling, testthat (>= 3.0.0), utils, viridis

SystemRequirements Python (via basilisk)

VignetteBuilder knitr

Language en-US

URL https://github.com/BorchLab/Ibex/

BugReports https://github.com/BorchLab/Ibex/issues

Roxygen list(markdown = TRUE)

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

git_branch RELEASE_3_22

git_last_commit 92396da

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-11-02

Author Nick Borcherding [aut, cre]

Maintainer Nick Borcherding <ncborch@gmail.com>

Contents

`Ibex-package` *Ibex: Methods for BCR single-cell embedding*

Description

Ibex implements methods for embedding B-cell receptor (BCR) sequences from single-cell assays into a continuous latent space. It supports amino-acid property–based and sequence-order encodings via a TensorFlow autoencoder, and interoperates with common single-cell containers such as **SingleCellExperiment** and **SeuratObject**.

Details

Key features

- Encode BCR sequence information using biochemical properties or raw sequence order (TensorFlow autoencoder).
- Interoperate with **SingleCellExperiment** and **SeuratObject** for downstream analysis and visualization.
- Utilities for loading pretrained models and managing dependencies in an isolated **basilisk** environment.

Getting started

```
browseVignettes("Ibex")
```

Models and caching Pretrained encoders can be retrieved with `aa.model.loader()`, which validates against internal metadata and caches downloaded artifacts; see the function help for cache location and behavior.

Python/TensorFlow note Ibex uses **basilisk** to provision an isolated Python environment at runtime; no manual setup is usually required.

Author(s)

Maintainer: Nick Borcherding <ncborch@gmail.com>

See Also

https://github.com/BorchLab/Ibex
https://github.com/BorchLab/Ibex/issues

`combineExpandedBCR` *combineBCR for CDR1/2/3 sequences*

Description

This function enhances BCR processing by incorporating additional sequence information from CDR1 and CDR2 regions before applying the BCR combination logic. The function depends on `scRepertoire::combineBCR()`.

Usage

```
combineExpandedBCR(
  input.data,
  samples = NULL,
  ID = NULL,
  call.related.clones = TRUE,
  threshold = 0.85,
  removeNA = FALSE,
  removeMulti = FALSE,
  filterMulti = TRUE,
  filterNonproductive = TRUE
)
```

Arguments

`input.data`	List of filtered contig annotations.
`samples`	Character vector. Labels of samples (required).
`ID`	Character vector. Additional sample labeling (optional).
`call.related.clones`	Logical. Whether to call related clones based on nucleotide sequence and V gene. Default is `TRUE`.
`threshold`	Numeric. Normalized edit distance for clone clustering. Default is `0.85`.
`removeNA`	Logical. Whether to remove any chain without values. Default is `FALSE`.
`removeMulti`	Logical. Whether to remove barcodes with more than two chains. Default is `FALSE`.
`filterMulti`	Logical. Whether to select the highest-expressing light and heavy chains. Default is `TRUE`.
`filterNonproductive`	Logical. Whether to remove nonproductive chains. Default is `TRUE`.

Value

A list of consolidated BCR clones with expanded CDR sequences.

See Also

scRepertoire::combineBCR()

Examples

```
#' # Get Data
ibex_vdj <- get(data("ibex_vdj"))

combined.BCR <- combineExpandedBCR(list(ibex_vdj),
                                   samples = "Sample1",
                                   filterNonproductive = TRUE)
```

CoNGAfy *Reduce a Single-Cell Object to Representative Cells*

Description

This function generates a single-cell object with a reduced representation of RNA expression by clone. The approach is inspired by the method introduced in CoNGA. Users can generate either a mean representation of features by clone or identify a representative cell using count-based minimal Euclidean distance. Please read and cite the original work by the authors of CoNGA.

Usage

```
CoNGAfy(
  input.data,
  method = "dist",
  features = NULL,
  assay = "RNA",
  meta.carry = c("CTaa", "CTgene")
)
```

Arguments

`input.data`	A single-cell dataset in Seurat or SingleCellExperiment format.
`method`	Character. Specifies the method to reduce the dataset: • "mean" - Computes the mean expression of selected features across cells in each clonotype. • "dist" - Uses PCA reduction to identify the cell with the minimal Euclidean distance within each clonotype group.
`features`	Character vector. Selected genes for the reduction. If `NULL` (default), all genes are used.
`assay`	Character. The name of the assay or assays to include in the output. Defaults to the active assay.
`meta.carry`	Character vector. Metadata variables to carry over from the input single-cell object to the output.

Value

A reduced single-cell object where each clonotype is represented by a single cell.

Examples

```
#' # Get Data
ibex_example <- get(data("ibex_example"))

ibex.clones <- CoNGAfy(ibex_example,
                       method = "dist")

ibex.clones <- CoNGAfy(ibex_example,
                       method = "mean")
```

`filter.cells` *Filter Single-Cell Data Based on CDR3 Sequences*

Description

This function subsets a Seurat or SingleCellExperiment object, removing cells where the `CTaa` column is missing or contains unwanted patterns.

Usage

```
filter.cells(sc.obj, chain)
```

Arguments

`sc.obj`	A Seurat or SingleCellExperiment object.
`chain`	Character. Specifies the chain type ("Heavy" or "Light").

Value

A filtered Seurat or SingleCellExperiment object.

`ibex_example` *A SingleCellExperiment object with 200 randomly-sampled B cells with BCR sequences from the 10x Genomics 2k_BEAM-Ab_Mouse_HEL_5pv2 dataset.*

Description

This object includes normalized gene expression values, metadata annotations, and B cell clonotype information derived from 10x V(D)J sequencing. It is intended as a small example dataset for testing and demonstration purposes.

Format

A SingleCellExperiment object with 32,285 genes (rows) and 200 cells (columns).

assays List of matrices containing expression values: counts (raw counts) and logcounts (log-transformed).

rowData Empty in this example (no gene-level annotations).

colData A DataFrame with 14 columns of cell metadata, including: - orig.ident: Original sample identity. - nCount_RNA: Total number of counts per cell. - nFeature_RNA: Number of detected genes per cell. - cloneSize: Size of each clone. - ident: Cluster assignment.

reducedDims Contains dimensionality reductions: PCA, pca, and apca.

altExp One alternative experiment named BEAM containing additional expression data.

Ibex_matrix *Ibex Matrix Interface*

Description

This function runs the Ibex algorithm to generate latent vectors from input data. The output can be returned as a matrix, with options to choose between deep learning autoencoders or geometric transformations based on the BLOSUM62 matrix.

Usage

```
Ibex_matrix(
  input.data,
  chain = c("Heavy", "Light"),
  method = c("encoder", "geometric"),
  encoder.model = c("CNN", "VAE", "CNN.EXP", "VAE.EXP"),
 encoder.input = c("atchleyFactors", "crucianiProperties", "kideraFactors", "MSWHIM",
    "tScales", "OHE"),
  geometric.theta = pi/3,
  species = "Human",
  verbose = TRUE
)
```

Arguments

input.data	Input data, which can be: • A Single Cell Object in Seurat or SingleCellExperiment format • The output of scRepertoire::combineBCR() or combineExpandedBCR()
chain	Character. Specifies which chain to analyze: • "Heavy" for the heavy chain • "Light" for the light chain
method	Character. The algorithm to use for generating latent vectors: • "encoder" - Uses deep learning autoencoders • "geometric" - Uses geometric transformations based on the BLOSUM62 matrix

`encoder.model` Character. The type of autoencoder model to use:

- "CNN" - CDR3 Convolutional Neural Network-based autoencoder
- "VAE" - CDR3 Variational Autoencoder
- "CNN.EXP" - CDR1/2/3 CNN
- "VAE.EXP" - CDR1/2/3 VAE

`encoder.input` Character. Specifies the input features for the encoder model. Options include:

- Amino Acid Properties: "atchleyFactors", "crucianiProperties", "kideraFactors", "MSWHIM","tScales", "zScales"
- "OHE" for One Hot Encoding

`geometric.theta`
Numeric. Angle (in radians) for the geometric transformation. Only used when `method = "geometric"`.

`species` Character. Default is "Human" or "Mouse".

`verbose` Logical. Whether to print progress messages. Default is TRUE.

Value

A matrix of latent vectors generated by the specified method.

See Also

`immApex::propertyEncoder()`, `immApex::geometricEncoder()`

Examples

```
# Get Data
ibex_example <- get(data("ibex_example"))

# Using the encoder method with a variational autoencoder
ibex_values <- Ibex_matrix(ibex_example,
                           chain = "Heavy",
                           method = "encoder",
                           encoder.model = "VAE",
                           encoder.input = "atchleyFactors")

# Using the geometric method with a specified angle
ibex_values <- Ibex_matrix(ibex_example,
                           chain = "Heavy",
                           method = "geometric",
                           geometric.theta = pi)
```

---

ibex_vdj *Full filtered_annotated_contig.csv from the 10x 2k_BEAM-Ab_Mouse_HEL_5pv2*

Description

This dataset contains single-cell V(D)J sequencing annotations from the 10x Genomics BEAM-Ab Mouse dataset. It includes V(D)J gene calls, CDR regions, productivity information, and clonotype assignments for each contig.

Format

A data frame with 6 rows and 35 columns:

barcode Character. Unique cell barcode.

is_cell Logical. Whether the barcode is identified as a cell.

contig_id Character. Unique identifier for each contig.

high_confidence Logical. Whether the contig is high confidence.

length Integer. Length of the contig.

chain Character. Chain type (e.g., IGH, IGK).

v_gene Character. V gene annotation.

d_gene Character. D gene annotation.

j_gene Character. J gene annotation.

c_gene Character. C gene annotation.

full_length Logical. Whether the contig is full-length.

productive Logical. Whether the contig is productive.

fwr1 Character. Amino acid sequence for Framework Region 1.

fwr1_nt Character. Nucleotide sequence for FWR1.

cdr1 Character. Amino acid sequence for CDR1.

cdr1_nt Character. Nucleotide sequence for CDR1.

fwr2 Character. Amino acid sequence for FWR2.

fwr2_nt Character. Nucleotide sequence for FWR2.

cdr2 Character. Amino acid sequence for CDR2.

cdr2_nt Character. Nucleotide sequence for CDR2.

fwr3 Character. Amino acid sequence for FWR3.

fwr3_nt Character. Nucleotide sequence for FWR3.

cdr3 Character. Amino acid sequence for CDR3.

cdr3_nt Character. Nucleotide sequence for CDR3.

fwr4 Character. Amino acid sequence for FWR4.

fwr4_nt Character. Nucleotide sequence for FWR4.

reads Integer. Number of reads supporting the contig.

umis Integer. Number of UMIs supporting the contig.

raw_clonotype_id Character. Clonotype ID from 10x output.

raw_consensus_id Character. Consensus ID from 10x output.

exact_subclonotype_id Integer. Exact subclonotype grouping.

runIbex *Ibex Single-Cell Calculation*

Description

This function applies the Ibex algorithm to single-cell data, integrating seamlessly with Seurat or SingleCellExperiment pipelines. The algorithm generates latent dimensions using deep learning or geometric transformations, storing the results in the dimensional reduction slot. `runIbex` will automatically subset the single-cell object based on amino acid sequences present for the given chain selection.

Usage

```
runIbex(
  sc.data,
  chain = "Heavy",
  method = "encoder",
  encoder.model = "VAE",
  encoder.input = "atchleyFactors",
  geometric.theta = pi,
  reduction.name = "Ibex",
  species = "Human",
  verbose = TRUE
)
```

Arguments

`sc.data`	A single-cell dataset, which can be: • A Seurat object • A SingleCellExperiment object
`chain`	Character. Specifies the chain to analyze: • "Heavy" for the heavy chain • "Light" for the light chain
`method`	Character. Algorithm to use for generating latent dimensions: • "encoder" - Uses deep learning autoencoders • "geometric" - Uses geometric transformations based on the BLOSUM62 matrix
`encoder.model`	Character. The type of autoencoder model to use: • "CNN" - CDR3 Convolutional Neural Network-based autoencoder • "VAE" - CDR3 Variational Autoencoder • "CNN.EXP" - CDR1/2/3 CNN • "VAE.EXP" - CDR1/2/3 VAE
`encoder.input`	Character. Input features for the encoder model: • Amino Acid Properties: "atchleyFactors", "crucianiProperties", "kideraFactors", "MSWHIM", "tScales" • "OHE" - One Hot Encoding

`geometric.theta`
: Numeric. Angle (in radians) for geometric transformation. Used only when `method = "geometric"`.

`reduction.name`
: Character. The name to assign to the dimensional reduction. This is useful for running Ibex with multiple parameter settings and saving results under different names.

`species`
: Character. Default is "Human" or "Mouse".

`verbose`
: Logical. Whether to print progress messages. Default is TRUE.

Value

An updated Seurat or SingleCellExperiment object with Ibex dimensions added to the dimensional reduction slot.

Examples

```
# Get Data
ibex_example <- get(data("ibex_example"))

# Using the encoder method with a variational autoencoder
ibex_example <- runIbex(ibex_example,
                        chain = "Heavy",
                        method = "encoder",
                        encoder.model = "VAE",
                        encoder.input = "atchleyFactors")

# Using the geometric method with a specified angle
ibex_example <- runIbex(ibex_example,
                        chain = "Heavy",
                        method = "geometric",
                        geometric.theta = pi)
```

Index