

# Package ‘Ibex’

September 24, 2026

**Title** Methods for BCR single-cell embedding

**Version** 1.2.0

**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** 7.3.3

**biocViews** Software, Immunology, SingleCell, Classification, Annotation, Sequencing

**Depends** R (>= 4.5.0)

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

**Suggests** basilisk.utils, BiocStyle, bluster, dplyr, ggplot2, kableExtra, knitr, lifecycle, 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\_23

**git\_last\_commit** 222b168

**git\_last\_commit\_date** 2026-04-28  
**Repository** Bioconductor 3.23  
**Date/Publication** 2026-09-23  
**Author** Nick Borcharding [aut, cre, cph],  
Qile Yang [ctb] (ORCID: <<https://orcid.org/0009-0005-0148-2499>>)  
**Maintainer** Nick Borcharding <[ncborch@gmail.com](mailto:ncborch@gmail.com)>

Contents

|                              |           |
|------------------------------|-----------|
| Ibex-package . . . . .       | 2         |
| combineExpandedBCR . . . . . | 3         |
| CoNGAfy . . . . .            | 4         |
| filter.cells . . . . .       | 5         |
| ibex_example . . . . .       | 6         |
| Ibex_matrix . . . . .        | 6         |
| ibex_vdj . . . . .           | 8         |
| runIbex . . . . .            | 10        |
| <b>Index</b>                 | <b>12</b> |

---

|              |                                                    |
|--------------|----------------------------------------------------|
| Ibex-package | <i>Ibex: Methods for BCR single-cell embedding</i> |
|--------------|----------------------------------------------------|

---

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 Borchering <ncborch@gmail.com> [copyright holder]

Other contributors:

- Qile Yang <qile.yang@berkeley.edu> ([ORCID](#)) [contributor]

### See Also

<https://github.com/BorchLab/Ibex>

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

---

|                    |                                          |
|--------------------|------------------------------------------|
| combineExpandedBCR | <i>combineBCR for CDR1/2/3 sequences</i> |
|--------------------|------------------------------------------|

---

### 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

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>input.data</code> | List of filtered contig annotations.                     |
| <code>samples</code>    | Character vector. Labels of samples (required).          |
| <code>ID</code>         | Character vector. Additional sample labeling (optional). |

|                                  |                                                                                                   |
|----------------------------------|---------------------------------------------------------------------------------------------------|
| <code>call.related.clones</code> | Logical. Whether to call related clones based on nucleotide sequence and V gene. Default is TRUE. |
| <code>threshold</code>           | Numeric. Normalized edit distance for clone clustering. Default is 0.85.                          |
| <code>removeNA</code>            | Logical. Whether to remove any chain without values. Default is FALSE.                            |
| <code>removeMulti</code>         | Logical. Whether to remove barcodes with more than two chains. Default is FALSE.                  |
| <code>filterMulti</code>         | Logical. Whether to select the highest-expressing light and heavy chains. Default is TRUE.        |
| <code>filterNonproductive</code> | 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: <ul style="list-style-type: none"> <li>• "mean" - Computes the mean expression of selected features across cells in each clonotype.</li> <li>• "dist" - Uses PCA reduction to identify the cell with the minimal Euclidean distance within each clonotype group.</li> </ul> |
| 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 | <i>A SingleCellExperiment object with 200 randomly-sampled B cells with BCR sequences from the 10x Genomics 2k_BEAM-Ab_Mouse_HEL_5pv2 dataset.</i> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------|

---

**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 | <i>Ibex Matrix Interface</i> |
|-------------|------------------------------|

---

**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: <ul style="list-style-type: none"> <li>• A Single Cell Object in Seurat or SingleCellExperiment format</li> <li>• The output of <code>scRepertoire::combineBCR()</code> or <code>combineExpandedBCR()</code></li> <li>• <b>[Experimental]</b> A character vector of amino acid sequences. The chain parameter specifies whether these are heavy or light chain sequences. For expanded models (CNN.EXP/VAE.EXP), sequences should be formatted as CDR1-CDR2-CDR3 separated by hyphens. If the vector is named, the names will be used as row names in the output.</li> </ul> |
| chain           | Character. Specifies which chain to analyze: <ul style="list-style-type: none"> <li>• "Heavy" for the heavy chain</li> <li>• "Light" for the light chain</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| method          | Character. The algorithm to use for generating latent vectors: <ul style="list-style-type: none"> <li>• "encoder" - Uses deep learning autoencoders</li> <li>• "geometric" - Uses geometric transformations based on the BLOSUM62 matrix</li> </ul>                                                                                                                                                                                                                                                                                                                                                    |
| encoder.model   | Character. The type of autoencoder model to use: <ul style="list-style-type: none"> <li>• "CNN" - CDR3 Convolutional Neural Network-based autoencoder</li> <li>• "VAE" - CDR3 Variational Autoencoder</li> <li>• "CNN.EXP" - CDR1/2/3 CNN</li> <li>• "VAE.EXP" - CDR1/2/3 VAE</li> </ul>                                                                                                                                                                                                                                                                                                               |
| encoder.input   | Character. Specifies the input features for the encoder model. Options include: <ul style="list-style-type: none"> <li>• Amino Acid Properties: "atchleyFactors", "crucianiProperties", "kideraFactors", "MSWHIM", "tScales", "zScales"</li> <li>• "OHE" for One Hot Encoding</li> </ul>                                                                                                                                                                                                                                                                                                               |
| 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)

# Using a character vector of amino acid sequences
sequences <- c("CARDYW", "CARDSSGYW", "CARDTGYW")
ibex_values <- Ibex_matrix(sequences,
  chain = "Heavy",
  method = "geometric")
```

---

|          |                                                                                  |
|----------|----------------------------------------------------------------------------------|
| ibex_vdj | <i>Full filtered_annotated_contig.csv from the 10x 2k_BEAM-Ab_Mouse_HEL_5pv2</i> |
|----------|----------------------------------------------------------------------------------|

---

**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: <ul style="list-style-type: none"> <li>• A Seurat object</li> <li>• A SingleCellExperiment object</li> </ul>                                                                                                                                        |
| chain         | Character. Specifies the chain to analyze: <ul style="list-style-type: none"> <li>• "Heavy" for the heavy chain</li> <li>• "Light" for the light chain</li> </ul>                                                                                                                        |
| method        | Character. Algorithm to use for generating latent dimensions: <ul style="list-style-type: none"> <li>• "encoder" - Uses deep learning autoencoders</li> <li>• "geometric" - Uses geometric transformations based on the BLOSUM62 matrix</li> </ul>                                       |
| encoder.model | Character. The type of autoencoder model to use: <ul style="list-style-type: none"> <li>• "CNN" - CDR3 Convolutional Neural Network-based autoencoder</li> <li>• "VAE" - CDR3 Variational Autoencoder</li> <li>• "CNN.EXP" - CDR1/2/3 CNN</li> <li>• "VAE.EXP" - CDR1/2/3 VAE</li> </ul> |
| encoder.input | Character. Input features for the encoder model:                                                                                                                                                                                                                                         |

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

|                              |                                                                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>geometric.theta</code> | Numeric. Angle (in radians) for geometric transformation. Used only when <code>method = "geometric"</code> .                                                           |
| <code>reduction.name</code>  | 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. |
| <code>species</code>         | Character. Default is "Human" or "Mouse".                                                                                                                              |
| <code>verbose</code>         | 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

## **\* package**

Ibex-package, [2](#)

combineExpandedBCR, [3](#)

combineExpandedBCR(), [7](#)

CoNGAfy, [4](#)

filter.cells, [5](#)

Ibex (Ibex-package), [2](#)

Ibex-package, [2](#)

ibex\_example, [6](#)

Ibex\_matrix, [6](#)

ibex\_vdj, [8](#)

immApex::geometricEncoder(), [8](#)

immApex::propertyEncoder(), [8](#)

runIbex, [10](#)

scRepertoire::combineBCR(), [3](#), [4](#), [7](#)