

# Package ‘StatescopeR’

April 9, 2026

**Type** Package

**Title** StatescopeR framework for discovery of cell states from cell type-specific gene expression profiles inferred from bulk mRNA profiles

**Version** 0.99.36

**Depends** R (>= 4.6.0)

**Imports** S4Vectors, SummarizedExperiment, reticulate, methods, SingleCellExperiment, matrixStats, scran, basilisk, Matrix, ComplexHeatmap, ggplot2, cowplot, utils

**Suggests** BiocStyle, knitr, RefManageR, rmarkdown, sessioninfo, scRNAseq, scuttle, testthat

## Description

StatescopeR is an R wrapper around Statescope, a computational framework designed to discover cell states from cell type-specific gene expression profiles inferred from bulk RNA profiles.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**StagedInstall** no

**biocViews** GeneExpression, RNASeq, SingleCell, Bayesian, Transcriptomics, Software

**URL** <https://github.com/tgac-vumc/StatescopeR>

**BugReports** <https://github.com/tgac-vumc/StatescopeR/issues>

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/StatescopeR>

**git\_branch** devel

**git\_last\_commit** 9120937

**git\_last\_commit\_date** 2026-04-08

**Repository** Bioconductor 3.23

**Date/Publication** 2026-04-08

**Author** Mischa Steketeer [aut, cre] (ORCID: <https://orcid.org/0000-0001-7138-7554>),  
 Bauke Ylstra [ths] (ORCID: <https://orcid.org/0000-0001-9479-3010>),  
 Yongsoo Kim [ths] (ORCID: <https://orcid.org/0000-0002-2995-2131>),  
 KWF Kankerbestrijding [fnd]

**Maintainer** Mischa Steketeer <m.f.b.steketeer@amsterdammc.nl>

## Contents

StatescopeR-package . . . . .	2
barplot_stateloadings . . . . .	3
BLADE_deconvolution . . . . .	4
create_signature . . . . .	6
fetch_signature . . . . .	7
fraction_eval . . . . .	7
fraction_heatmap . . . . .	8
gather_true_fractions . . . . .	9
Refinement . . . . .	10
select_genes . . . . .	11
StateDiscovery . . . . .	12
statescope . . . . .	14
<b>Index</b>	<b>15</b>

---

StatescopeR-package     *The StatescopeR package*

---

## Description

framework for discovery of cell states from bulk mRNA profiles

## Details

StatescopeR starts from lognormalized cp10k single cell data and a bulk RNA dataset to be used for deconvolution and cell state analysis. From this point StatescopeR has some functions to create a signature and select genes for deconvolution. `create_signature` creates a signature of lognormalized cp10k single cell data for deconvolution and cell state analysis. `select_genes` selects genes which best discriminate cell types for use in deconvolution. Alternatively you can also use: `fetch_signature` to fetch one of the premade signatures + selected\_genes from <https://github.com/tgac-vumc/StatescopeData> After these steps and proper cp10k normalization of the bulk mRNA deconvolution and cell state analysis can be done with the following functions: `BLADE_deconvolution` estimates cell fractions from bulk mRNA. `Refinement` refines cell type-specific gene expression profile estimates to better capture inter-sample variability. `StateDiscovery` discovery cell states from inferred cell type-specific gene expression profiles.

After these steps the following functions provide some evaluation and plotting functions: `gather_true_fractions` gets true cell fractions from scRNAseq data. `fraction_eval` plots the correlation of estimated vs true cell fraction over all samples. `fraction_heatmap` plots a heatmap of estimated cell fractions. `barplot_stateloadings` plots a barplot showing the top n most important stateloadings.

## Author(s)

**Maintainer:** Mischa Steketee <m.f.b.steketee@amsterdamumc.nl> ([ORCID](#))

Other contributors:

- Bauke Ylstra <b.ylstra@amsterdamumc.nl> ([ORCID](#)) [thesis advisor]
- Yongsoo Kim <yo.kim@amsterdamumc.nl> ([ORCID](#)) [thesis advisor]
- KWF Kankerbestrijding [funder]

## See Also

Useful links:

- <https://github.com/tgac-vumc/StatescopeR>
- Report bugs at <https://github.com/tgac-vumc/StatescopeR/issues>

---

barplot\_stateloadings *Create a barplot of top stateloadings*

---

## Description

Create a barplot of top stateloadings

## Usage

```
barplot_stateloadings(Statescope, top_n = 1)
```

## Arguments

Statescope	SummarizedExperiment object from StateDiscovery
top_n	integer selecting how many genes to show per state

## Value

A barplot rendered to the active graphics device

**Examples**

```

#' ## Load Discovered Statescope object
load(system.file("extdata", "example_Statescope_Discovered.RData",
  package = "StatescopeR"
))

## Plot fraction heatmap
barplot_stateloadings(Statescope, top_n = 1)

```

---

BLADE\_deconvolution    *Run BLADE deconvolution*

---

**Description**

BLADE\_deconvolution.R Runs BLADE to estimate cell fractions from bulk mRNA

**Usage**

```

BLADE_deconvolution(
  signature,
  bulk,
  genes,
  prior = NULL,
  cores = 1L,
  Alpha = 1L,
  Alpha0 = 1000L,
  Kappa0 = 1L,
  sY = 1L,
  Nrep = 10L,
  Nrepfinal = 1000L
)

```

**Arguments**

signature	SimpleList with mu and sigma, both nGene x nCelltype dataframes, respectively mean gene expression and mean-variance corrected variance per cell type
bulk	nSample x nGene mRNA to be deconvolved
genes	subset of genes to be used for deconvolution
prior	(optional) nSample x nCelltype matrix with prior fraction expectations
cores	number of cores to use for paralellization
Alpha	BLADE Hyperparameter
Alpha0	BLADE Hyperparameter
Kappa0	BLADE Hyperparameter
sY	BLADE Hyperparameter
Nrep	Number of BLADE initializations
Nrepfinal	Number of maximum optimization iterations

**Value**

SummarizedExperiment object with BLADE output and estimated fractions

**Examples**

```

if (requireNamespace("scRNAseq", quietly = TRUE)) {
  library(scRNAseq)
  library(scuttle)
  ## Load SegerstolpePancreas data set
  scRNAseq <- SegerstolpePancreasData()

  ## remove duplicate genes
  scRNAseq <- scRNAseq[!duplicated(rownames(scRNAseq)), ]
  ## Subset to 1 healthy and 2 type 2 diabetes samples
  scRNAseq <- scRNAseq[, scRNAseq$individual %in% c(
    "H3",
    "T2D1", "T2D2"
  )]
  ## remove cells with no cell type label
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]

  ## remove very rare cell types (<100 cells in total data set)
  celltypes_to_remove <-
  names(table(scRNAseq$`cell type`)[(table(scRNAseq$`cell type`) < 100)])
  scRNAseq <- scRNAseq[, !scRNAseq$`cell type` %in% celltypes_to_remove]

  ## Create pseudobulk and normalize to cp10k
  pseudobulk <- aggregateAcrossCells(scRNAseq, ids = scRNAseq$individual)
  normcounts(pseudobulk) <- calculateCPM(pseudobulk) / 100
  pseudobulk <- as(pseudobulk, "SummarizedExperiment")
  rownames(pseudobulk) <- rownames(scRNAseq)

  ## Load signature
  load(system.file("extdata", "example_signature.RData",
    package = "StatescopeR"
  ))

  ## Load selected_genes
  load(system.file("extdata", "example_selected_genes.RData",
    package = "StatescopeR"
  ))

  ## Load prior
  load(system.file("extdata", "example_prior.RData",
    package = "StatescopeR"
  ))

  ## Perform Deconvolution with BLADE
  Statescope <- BLADE_deconvolution(
    signature, pseudobulk, selected_genes,
    prior, 1L,
    Nrep = 1L
  )

```

```

)
  ## show estimated fractions
  S4Vectors::metadata(Statescope)$fractions
}

```

---

create\_signature      *Create scRNAseq Signature*

---

## Description

create\_signature Creates signature from scRNAseq data

## Usage

```
create_signature(scRNAseq, hvg_genes = FALSE, n_hvg_genes = 3000L, labels)
```

## Arguments

scRNAseq	SingleCellExperiment object of which to make signature
hvg_genes	boolean which chooses if mu and omega should be subset to highly variable genes or not
n_hvg_genes	int which allows the users to choose the number of highly variable genes
labels	character vector for the cell type labels

## Value

SimpleList DataFrames for Mu (mean per gene per cell type) and Omega (variance corrected std.dev per gene per cell type)

## Examples

```

if (requireNamespace("scRNAseq", quietly = TRUE)) {
  library(scRNAseq)
  library(scuttle)
  ## Load scRNAseq
  scRNAseq <- scRNAseq::SeegerstolpePancreasData()

  ## remove NA cells
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]

  ## Normalize (cp10k) and logtransform scRNAseq
  cpm(scRNAseq) <- scuttle::calculateCPM(scRNAseq)
  SingleCellExperiment::logcounts(scRNAseq) <- log1p(cpm(scRNAseq) / 100)

  ## Create signature
  signature <- create_signature(scRNAseq, labels = scRNAseq$`cell type`)
}

```

---

fetch_signature	<i>Fetch scRNAseq Signature</i>
-----------------	---------------------------------

---

**Description**

fetch\_signature Fetches signature from StatescopeData repository

**Usage**

```
fetch_signature(tumor_type, n_celltypes)
```

**Arguments**

tumor_type	string choosing the tumor type from available signatures at: <a href="https://github.com/tgac-vumc/StatescopeData">https://github.com/tgac-vumc/StatescopeData</a>
n_celltypes	integer choosing the number of cell types of the signature from: <a href="https://github.com/tgac-vumc/StatescopeData">https://github.com/tgac-vumc/StatescopeData</a>

**Value**

SimpleList with DataFrames for Mu (mean per gene per cell type) and Omega (variance corrected std.dev per gene per cell type) and vector of selected genes for deconvolution chosen by AutoGeneS

**Examples**

```
signature <- fetch_signature('PDAC', 7)
selected_genes <- signature$selected_genes
```

---

fraction_eval	<i>Create a barplot of TRUE vs estimated cfs</i>
---------------	--

---

**Description**

Create a barplot of TRUE vs estimated cfs

**Usage**

```
fraction_eval(Statescope, true_fractions)
```

**Arguments**

Statescope	SummarizedExperiment object from BLADE_deconvolution
true_fractions	s4 Dataframe with true fractions to compare with estimated fractions

**Value**

A barplot rendered to the active graphics device

**Examples**

```
## Load True fractions
load(system.file("extdata", "example_true_fractions.RData",
  package = "StatescopeR"
))

## Load Deconvolved Statescope object
load(system.file("extdata", "example_Statescope_Deconvolved.RData",
  package = "StatescopeR"
))

## ## Plot fraction correlation and RMSE per ct
fraction_eval(Statescope, true_fractions)
```

---

fraction_heatmap	<i>Create a heatmap of the estimated fractions</i>
------------------	--

---

**Description**

Create a heatmap of the estimated fractions

**Usage**

```
fraction_heatmap(Statescope, ...)
```

**Arguments**

Statescope	SummarizedExperiment object from BLADE_deconvolution
...	other parameters to pass to [ComplexHeatmap::Heatmap()]

**Value**

A heatmap rendered to the active graphics device

**Examples**

```
## Load Deconvolved Statescope object
load(system.file("extdata", "example_Statescope_Deconvolved.RData",
  package = "StatescopeR"
))

## Plot fraction heatmap
fraction_heatmap(Statescope)
```

---

gather\_true\_fractions *Gather true fractions*

---

## Description

gather\_true\_fractions Gathers true fractions of all cell types per sample from scRNAseq data

## Usage

```
gather_true_fractions(scRNAseq, ids, label_col)
```

## Arguments

scRNAseq	SingleCellExperiment object of which to gather fractions per sample
ids	character vector with ids of samples
label_col	character for the column name with the cell type labels

## Value

DataFrame with fractions of all cell types per sample

## Examples

```
if (requireNamespace("scRNAseq", quietly = TRUE)) {  
  library(scRNAseq)  
  ## Load data  
  scRNAseq <- scRNAseq::SegerstolpePancreasData()  
  ## Subset to 1 healthy and 3 type 2 diabetes samples  
  scRNAseq <- scRNAseq[, scRNAseq$individual %in% c(  
    "H3",  
    "T2D1", "T2D2"  
  )]  
  ## remove NA cells  
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]  
  
  ## remove cells with less than 100 in total cohort  
  celltypes_to_remove <- names(table(scRNAseq$`cell type`)  
    [(table(scRNAseq$`cell type`) < 100)])  
  scRNAseq <- scRNAseq[, !scRNAseq$`cell type` %in% celltypes_to_remove]  
  
  true_fractions <- gather_true_fractions(scRNAseq,  
    ids = scRNAseq$individual, label_col = "cell type"  
  )  
}
```

Refinement

*Run Refinement***Description**

```
Refinement.R # Perform Gene Expression Refinement
```

**Usage**

```
Refinement(Statescope, signature, bulk, cores = 1L)
```

**Arguments**

Statescope	SummarizedExperiment object from BLADE_deconvolution
signature	SimpleList with mu and sigma, respectively mean gene expression and mean-variance corrected variance per cell type
bulk	mRNA to be refined
cores	number of cores to use for paralellization

**Details**

This function takes the output from BLADE\_deconvolution and refines the cell type specific gene expression by reoptimizing the initial estimates. It reoptimizes by fixing the estimated fractions and weighing the objective function value in a way that it tries to resemble the bulk RNA expression more than initially

**Value**

updated SummarizedExperiment object with ct\_specific\_gexp added

**Examples**

```
if (requireNamespace("scRNAseq", quietly = TRUE)) {
  library(scRNAseq)
  library(scuttle)
  ## Load SegerstolpePancreas data set
  scRNAseq <- SegerstolpePancreasData()

  ## remove duplicate genes
  scRNAseq <- scRNAseq[!duplicated(rownames(scRNAseq)), ]

  ## Subset to 1 healthy and 2 type 2 diabetes samples
  scRNAseq <- scRNAseq[, scRNAseq$individual %in% c(
    "H3",
    "T2D1", "T2D2"
  )]
  ## remove cells with no cell type label
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]
}
```

```

## remove very rare cell types (<100 cells in total data set)
celltypes_to_remove <- names(table(scRNAseq$`cell type`)
[(table(scRNAseq$`cell type`) < 100)])
scRNAseq <- scRNAseq[, !scRNAseq$`cell type` %in% celltypes_to_remove]

## Create pseudobulk and normalize to cp10k
pseudobulk <- aggregateAcrossCells(scRNAseq, ids = scRNAseq$individual)
normcounts(pseudobulk) <- calculateCPM(pseudobulk) / 100
pseudobulk <- as(pseudobulk, "SummarizedExperiment")
rownames(pseudobulk) <- rownames(scRNAseq)
## Load signature
load(system.file("extdata", "example_signature.RData",
  package = "StatescopeR")
))

## Load Deconvolved Statescope object
load(system.file("extdata", "example_Statescope_Deconvolved.RData",
  package = "StatescopeR")
))

## Run Refinement
Statescope <- Refinement(Statescope, signature, pseudobulk, 2L)

## Show cell type specific gene expression profile estimates
S4Vectors::metadata(Statescope)$ct_specific_gexp
}

```

---

select\_genes

*Select genes using AutoGeneS*


---

## Description

select\_genes.R select genes using AutoGeneS for deconvolution

## Usage

```
select_genes(scRNAseq, fixed_n_features = NA, n_hvg_genes = 3000L, labels)
```

## Arguments

scRNAseq	SingleCellExperiment object to use for gene selection, should be same as signature dataset
fixed_n_features	integer number of genes to pick with autogenes, default is NA which lets autogenes itself pick
n_hvg_genes	int which allows the users to choose the number of highly variable genes
labels	character vector with cell type labels

**Value**

Vector of genes to use for deconvolution

**Examples**

```

if (requireNamespace("scRNAseq", quietly = TRUE)) {
  library(scRNAseq)
  library(scuttle)
  ## Load SegerstolpePancreas data set
  scRNAseq <- SegerstolpePancreasData()

  ## remove duplicate genes
  scRNAseq <- scRNAseq[!duplicated(rownames(scRNAseq)), ]

  ## Subset to 1 healthy and 2 type 2 diabetes samples
  scRNAseq <- scRNAseq[, scRNAseq$individual %in% c(
    "H3",
    "T2D1", "T2D2"
  )]
  ## remove cells with no cell type label
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]

  ## remove rare cell types (<100 cells in total data set)
  celltypes_to_remove <- names(table(scRNAseq$`cell type`)
    [(table(scRNAseq$`cell type`) < 100)])
  scRNAseq <- scRNAseq[, !scRNAseq$`cell type` %in% celltypes_to_remove]

  ## remove NA cells
  scRNAseq <- scRNAseq[, !is.na(scRNAseq$`cell type`)]

  ## Normalize (cp10k) and logtransform scRNAseq
  cpm(scRNAseq) <- scuttle::calculateCPM(scRNAseq)
  logcounts(scRNAseq) <- log1p(cpm(scRNAseq) / 100)

  ## Select genes by autogenes
  selected_genes <- select_genes(scRNAseq, 3L,
    n_hvg_genes = 5L,
    labels = scRNAseq$`cell type`
  ) # 3 genes
}

```

---

StateDiscovery

*Run StateDiscovery*

---

**Description**

StateDiscovery.R Discovers states from refined ct-specific gep

**Usage**

```
StateDiscovery(  
  Statescope,  
  k = NA,  
  max_clusters = 10L,  
  n_iter = 10L,  
  n_final_iter = 100L,  
  min_cophenetic = 0.9,  
  Ncores = 1L  
)
```

**Arguments**

Statescope	SummarizedExperiment object from Statescope Refinement.
k	number of cluster to choose, default is NA for automatic selection
max_clusters	maximum allowed states per cell type.
n_iter	Number of initial cNMF restarts.
n_final_iter	Number of final cNMF restarts.
min_cophenetic	Minimum cophenetic coefficient to determine K.
Ncores	number of cores to use for paralellization.

**Value**

SummarizedExperiment object with statescores per celltype added

**Examples**

```
## Load Refined Statescope object  
load(system.file("extdata", "example_Statescope_Refined.RData",  
  package = "StatescopeR"  
)  
)  
  
## Discover states  
Statescope <- StateDiscovery(Statescope, k = 2L, Ncores = 2L)  
  
## Look at statescores and stateloadings  
S4Vectors::metadata(Statescope)$statescores  
S4Vectors::metadata(Statescope)$stateloadings
```

---

statescope

*Python environments*

---

**Description**

Python environments

**Usage**

statescope

**Format**

An object of class BasiliskEnvironment of length 1.

# Index

- \* **datasets**

- statescope, 14

- \* **internal**

- StatescopeR-package, 2

barplot\_stateloadings, 3, 3

BLADE\_deconvolution, 2, 4

create\_signature, 2, 6

fetch\_signature, 2, 7

fraction\_eval, 3, 7

fraction\_heatmap, 3, 8

gather\_true\_fractions, 3, 9

Refinement, 2, 10

select\_genes, 2, 11

StateDiscovery, 2, 12

statescope, 14

StatescopeR (StatescopeR-package), 2

StatescopeR-package, 2