Package ‘scAlign’

October 16, 2019

Version 1.0.0

Date 2019-02-11

Title An alignment and integration method for single cell genomics

Description An unsupervised deep learning method for data alignment, integration and estimation of per-cell differences in -omic data (e.g. gene expression) across datasets (conditions, tissues, species). See Johansen and Quon (2019) <doi:10.1101/504944> for more details.

URL https://github.com/quon-titative-biology/scAlign

BugReports https://github.com/quon-titative-biology/scAlign/issues

biocViews SingleCell, Transcriptomics, DimensionReduction,
NeuralNetwork

Depends R (>= 3.5)

Imports SingleCellExperiment (>= 1.4), Seurat (>= 2.3.4), tensorflow,
purrr, irlba, Rtsne, ggplot2, methods, utils, FNN

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

SystemRequirements python (< 3.7), tensorflow

RoxygenNote 6.1.1

License GPL-3

LazyData false

Encoding UTF-8

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

git_branch RELEASE_3_9

git_last_commit ef247aa

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author Nelson Johansen [aut, cre],
Gerald Quon [aut]

Maintainer Nelson Johansen <njjohansen@ucdavis.edu>

R topics documented:

`cellbench` *CellBench data for benchmarking*

Description

Data from RNA mixture experiments of CellBench which prove to be more difficult alignment tasks.

Usage

```
data(cellbench)
```

Format

An matrix object.

Source

CellBench

`gaussianKernel` *Computes gaussian kernel matrix*

Description

Tensorflow implementation of tSNE's gaussian kernel.

Usage

```
gaussianKernel(data, data_shape, labels = NULL, method = NULL,
  perplexity = 30, diag = "zero")
```

Arguments

`data`	cell x feature data matrix
`data_shape`	number of features for data
`labels`	cell x 1 annotation (label) vector
`method`	Kernel to compute pairwise cell similarity
`perplexity`	neighborhood parameter for gaussian kernel
`diag`	indicator for self similarity

Value

Tensorflow op

Examples

```
## Input data, 100 cells x 10 features
data = matrix(sample.int(1000, 100*10, TRUE), 100, 10)
rownames(data) = paste0("cell", seq_len(100))
colnames(data) = paste0("gene", seq_len(10))

result = gaussianKernel(data, nrow(data))
```

PlotTSNE *Creates tsne plot*

Description

Helper function to plot aligned data from the results of running scAlign().

Usage

```
PlotTSNE(object, data.use, labels.use = "scAlign.labels", cols = NULL,
  title = "", legend = "none", seed = 1234, ...)
```

Arguments

`object`	scAlign class object with aligned data
`data.use`	Specifies which alignment results to use.
`labels.use`	Specifies "dataset" or "celltype" labeling from object meta.data.
`cols`	Colours for plot
`title`	ggplot title
`legend`	Determines if legend should be drawn
`seed`	Random seed for reproducability
`...`	Additional arguments to Rtsne function
`labels`	Object labels

Value

ggplot2 object

Examples

```
library(SingleCellExperiment)

## Input data, 1000 genes x 100 cells
data = matrix( rnorm(1000*100,mean=0,sd=1), 1000, 100)
rownames(data) = paste0("gene", seq_len(1000))
colnames(data) = paste0("cell", seq_len(100))

age    = c(rep("young",50), rep("old",50))
labels = c(c(rep("type1",25), rep("type2",25)), c(rep("type1",25), rep("type2",25)))

ctrl.data = data[,which(age == "young")]
stim.data = data[,which(age == "old")]

## Build the SCE object for input to scAlign using Seurat preprocessing and variable gene selection
ctrlSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == "young")]))

stimSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == "old")]))

## Build the scAlign class object and compute PCs
scAlignHSC = scAlignCreateObject(sce.objects = list("YOUNG"=ctrlSCE,
                                                    "OLD"=stimSCE),
                                 labels = list(labels[which(age == "young")],
                                               labels[which(age == "old")]),
                                 pca.reduce = FALSE,
                                 cca.reduce = FALSE,
                                 project.name = "scAlign_Kowalcyzk_HSC")

## Run scAlign with high_var_genes
scAlignHSC = scAlign(scAlignHSC,
                   options=scAlignOptions(steps=100,
                                          log.every=100,
                                          norm=TRUE,
                                          early.stop=FALSE),
                   encoder.data="scale.data",
                   supervised='none',
                   run.encoder=TRUE,
                   run.decoder=FALSE,
                   log.results=FALSE,
                   log.dir=file.path('~/models','gene_input'),
                   device="CPU")

## Plot alignment for 3 input types
example_plot = PlotTSNE(scAlignHSC,
                          "ALIGNED-GENE",
                          "scAlign.labels",
                          title="scAlign-Gene",
                          perplexity=30)
```

scAlign *Run scAlign framework*

Description

Main function for scAlign that runs encoder and decoder networks

Usage

```
scAlign(sce.object, options = scAlignOptions(), encoder.data,
  decoder.data = encoder.data, supervised = "none",
  run.encoder = TRUE, run.decoder = FALSE, log.dir = "./models/",
  log.results = FALSE, device = "CPU")
```

Arguments

sce.object	scAlign object.
options	Training options for scAlign.
encoder.data	Which data format to use for alignment.
decoder.data	Which data format to use for interpolation.
supervised	Run scAlign in supervised mode, requires labels.
run.encoder	Run scAlign alignment procedure.
run.decoder	Run scAlign projection through paired decoders.
log.dir	Location to save results.
log.results	Determines if results should be written to log.dir.
device	Specify hardware to use. May not work on all systems, manually set CUDA_VISIBLE_DEVICES if necessary.

Value

SingleCellExperiment

Examples

```
library(Seurat)
library(SingleCellExperiment)

## Input data, 1000 genes x 100 cells
data = matrix(sample.int(10000, 1000*100, TRUE), 1000, 100)
rownames(data) = paste0("gene", seq_len(1000))
colnames(data) = paste0("cell", seq_len(100))

age    = c(rep("young",50), rep("old",50))
labels = c(c(rep("type1",25), rep("type2",25)), c(rep("type1",25), rep("type2",25)))

## Build the SCE object for input to scAlign using Seurat preprocessing and variable gene selection
ctrlSCE <- SingleCellExperiment(
             assays = list(scale.data = data[,which(age == "young")]))

stimSCE <- SingleCellExperiment(
             assays = list(scale.data = data[,which(age == "old")]))

## Build the scAlign class object and compute PCs
scAlignHSC = scAlignCreateObject(sce.objects = list("YOUNG"=ctrlSCE,
                                                   "OLD"=stimSCE),
                                 labels = list(labels[which(age == "young")],
                                               labels[which(age == "old")]),
                                 pca.reduce = TRUE,
                                 pcs.compute = 50,
                                 cca.reduce = TRUE,
                                 ccs.compute = 15,
                                 project.name = "scAlign_Kowalcyzk_HSC")

## Run scAlign with high_var_genes
scAlignHSC = scAlign(scAlignHSC,
               options=scAlignOptions(steps=1, log.every=1, norm=TRUE, early.stop=FALSE),
                   encoder.data="scale.data",
                   supervised='none',
                   run.encoder=TRUE,
                   run.decoder=FALSE,
                   log.dir=file.path('~/models','gene_input'),
                   device="CPU")
```

`scAlignCreateObject` *Creates scAlign object*

Description

Creates scAlign object

Usage

```
scAlignCreateObject(sce.objects, labels = list(), meta.data = NULL,
  pca.reduce = FALSE, pcs.compute = 20, cca.reduce = FALSE,
  ccs.compute = 15, data.use = "scale.data",
  project.name = "scAlignProject")
```

Arguments

`sce.objects`	List of Seurat or Matrix objects; sample x feature.
`labels`	List of labels for each object.
`meta.data`	Additional meta.data to add.
`pca.reduce`	Initial step of dimensionality be performced by PCA.
`pcs.compute`	Number of PCs to retrain for alignment.
`cca.reduce`	Initial step of dimensionality be performced by CCA.
`ccs.compute`	Number of CCs to retrain for alignment.
`data.use`	Specificies which data to use from a Seurat object for dimensionality reduction.
`project.name`	Name for current scAlign project.

Value

Initialized scAlign object

Examples

```
library(Seurat)
library(SingleCellExperiment)

## Input data, 1000 genes x 100 cells
data = matrix(sample.int(10000, 1000*100, TRUE), 1000, 100)
rownames(data) = paste0("gene", seq_len(1000))
colnames(data) = paste0("cell", seq_len(100))

age    = c(rep("young",50), rep("old",50))
labels = c(c(rep("type1",25), rep("type2",25)), c(rep("type1",25), rep("type2",25)))

ctrl.data = data[,which(age == "young")]
stim.data = data[,which(age == "old")]

ctrlSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == "young")]))

stimSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == "old")]))

## Build the scAlign class object and compute PCs
scAlignHSC = scAlignCreateObject(sce.objects = list("YOUNG"=ctrlSCE,
                                                    "OLD"=stimSCE),
                                 labels = list(labels[which(age == "young")],
                                               labels[which(age == "old")]),
                                 pca.reduce = TRUE,
                                 pcs.compute = 50,
                                 cca.reduce = TRUE,
                                 ccs.compute = 15,
                                 project.name = "scAlign_Kowalcyzk_HSC")
```

`scAlignMulti` *Run scAlignMultiway alignment*

Description

Main function for scAlignMulti that aligns multiple datasets

Usage

```
scAlignMulti(sce.object, options = scAlignOptions(), encoder.data,
  decoder.data = encoder.data, reference.data = "NULL",
  supervised = "none", run.encoder = TRUE, run.decoder = FALSE,
  log.dir = "./models/", log.results = FALSE, device = "CPU")
```

Arguments

`sce.object`	scAlign object.
`options`	Training options for scAlign.
`encoder.data`	Which data format to use for alignment.

`decoder.data` Which data format to use for interpolation.

`reference.data` Name of assay or reducedDim slot to use as reference. (Disables all pairs alignment and only aligns to a single reference)

`supervised` Run scAlign in supervised mode, requires labels.

`run.encoder` Run scAlign alignment procedure.

`run.decoder` Run scAlign projection through paired decoders.

`log.dir` Location to save results.

`log.results` Determines if results should be written to log.dir.

`device` Specify hardware to use. May not work on all systems, manually set CUDA_VISIBLE_DEVICES if necessary.

Value

SingleCellExperiment

Examples

```
library(Seurat)
library(SingleCellExperiment)

## Input data, 1000 genes x 100 cells
data = matrix(sample.int(10000, 1000*100, TRUE), 1000, 100)
rownames(data) = paste0(”gene”, seq_len(1000))
colnames(data) = paste0(”cell”, seq_len(100))

age    = c(rep(”young”,50), rep(”old”,50))
labels = c(c(rep(”type1”,25), rep(”type2”,25)), c(rep(”type1”,25), rep(”type2”,25)))

## Build the SCE object for input to scAlign using Seurat preprocessing and variable gene selection
ctrlSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == ”young”)]))

stimSCE <- SingleCellExperiment(
              assays = list(scale.data = data[,which(age == ”old”)]))

## Build the scAlign class object and compute PCs
scAlign = scAlignCreateObject(sce.objects = list(”YOUNG”=ctrlSCE,
                                                 ”OLD”=stimSCE),
                                 labels = list(labels[which(age == ”young”)],
                                               labels[which(age == ”old”)]),
                                 pca.reduce = TRUE,
                                 pcs.compute = 50,
                                 cca.reduce = TRUE,
                                 ccs.compute = 15,
                                 project.name = ”scAlign_Kowalcyzk_HSC”)

## Run scAlign with high_var_genes
scAlign = scAlignMulti(scAlign,
              options=scAlignOptions(steps=1, log.every=1, norm=TRUE, early.stop=FALSE),
                   encoder.data=”scale.data”,
                   reference.data=”YOUNG”,
                   supervised='none',
                   run.encoder=TRUE,
                   run.decoder=FALSE,
                   log.dir=file.path('~/models','gene_input'),
                   device="CPU")
```

scAlignOptions *Set training options*

Description

Defines parameters for optimizer and training procedure.

Usage

```
scAlignOptions(steps = 15000, batch.size = 300,
  learning.rate = 1e-04, log.every = 5000, architecture = "large",
  num.dim = 32, perplexity = 30, norm = TRUE, full.norm = FALSE,
  early.stop = FALSE, walker.loss = TRUE, reconc.loss = FALSE,
  walker.weight = 1, classifier.weight = 1, classifier.delay = NA,
  gpu.device = "0", seed = 1234)
```

Arguments

`steps` (default: 15000) Number of training iterations for neural networks.

`batch.size` (default: 150) Number of input samples per training batch.

`learning.rate` (default: 1e-4) Initial learning rate for ADAM.

`log.every` (default: 5000) Number of steps before saving results.

`architecture` (default: "small") Network function name for scAlign.

`num.dim` (default: 32) Number of dimensions for joint embedding space.

`perplexity` (default: 30) Determines the neighborhood size for each sample.

`norm` (default: TRUE) Normalize the data mini batches while training scAlign (repeated).

`full.norm` (default: FALSE) Normalize the data matrix prior to scAlign (done once).

`early.stop` (default: TRUE) Early stopping during network training.

`walker.loss` (default: TRUE) Add walker loss to model.

`reconc.loss` (default: FALSE) Add reconstruction loss to model during alignment.

`walker.weight` (default: 1.0) Weight on walker loss component

`classifier.weight`
(default: 1.0) Weight on classifier loss component

`classifier.delay`
(default: NULL) Delay classifier component of loss function until specific training step. Defaults to (2/3)*steps.

`gpu.device` (default: ’0’) Which gpu to use.

`seed` (default: 1245) Sets graph level random seed in tensorflow.

Value

Options data.frame

Examples

```
options=scAlignOptions(steps=15000,
                       log.every=5000,
                       early.stop=FALSE,
                       architecture="large")
```

Index