# Package 'FiRE'

January 2, 2019

**Title** Finder of Rare Entities (FiRE)

**Version** 1.0

**Description** The algorithm assigns rareness/ outlierness score to every sample in voluminous datasets. The algorithm makes multiple estimations of the proximity between a pair of samples, in low-dimensional spaces. To compute proximity, FiRE uses Sketching, a variant of locality sensitive hashing. For more details: Jindal, A., Gupta, P., Jayadeva and Sengupta, D., 2018. Discovery of rare cells from voluminous single cell expression data. Nature Communications, 9(1), p.4719. <doi:10.1038/s41467-018-07234-6>.

**URL** https://github.com/princethewinner/FiRE

**BugReports** https://github.com/princethewinner/FiRE/issues

**Note** While reporting issues on github start subject line with [cran-package]

**Depends** R (>= 3.2.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** methods, Rcpp (>= 0.12.19)

**LinkingTo** Rcpp, BH

**NeedsCompilation** yes

**Author** Prashant Gupta [aut, cre],
Aashi Jindal [aut],
Jayadeva [aut],
Debarka Sengupta [aut]

**Maintainer** Prashant Gupta <prashant10991@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-01-02 13:50:07 UTC

# R topics documented:



---

`FiRE-package` *Finder of Rare Entities (FiRE) - To assign rareness/ outlierness score to every sample*

---

## Description

This package sorts samples as per their rareness/ oultierness. Instead of dichotomized decisions, FiRE assigns rareness/ outlierness score to every sample. These scores can then be used to identify rare samples or outliers, with varying degrees of rareness. FiRE takes multiple estimations of the proximity between a pair of samples, in low-dimensional spaces to compute scores.

## Details

FiRE is written in c++ and wrapped with Rcpp to create an R interface. To use FiRE, an object with the number of estimators (L), number of dimensions to be sampled per estimator (M), number of bins per estimator (H = 1017881), seed for random number generator (seed = 0) and verbose level (verbose = 0/1) needs to be created. Once the object is created, `fit` function needs to be called to hash all the samples into bins. Once hashing is done, `score` function needs to be called to retrieve score for every sample. Sample commands may be seen in the Examples section. The resulting model has four model parameters which may be accessed, dimensions (d) of size M sampled per estimator, thresholds (ths) of size M per d, weights(w) of size M generated per estimator and bin (b) of size H per estimator.

## Author(s)

Prashant Gupta, prashant10991@gmail.com

Aashi Jindal, jindal.aashi21@gmail.com

Jayadeva, iitjd4@gmail.com

Debarka Sengupta, debarka@gmail.com

Maintainer: Prashant Gupta <prashant10991@gmail.com>

## References

[1]Jindal, A., Gupta, P., Jayadeva and Sengupta, D., 2018. Discovery of rare cells from voluminous single cell expression data. Nature Communications, 9(1), p.4719.

[2]Wang, Z., Dong, W., Josephson, W., Lv, Q., Charikar, M. and Li, K., 2007, June. Sizing sketches: a rank-based analysis for similarity search. In ACM SIGMETRICS Performance Evaluation Review (Vol. 35, No. 1, pp. 157-168). ACM.

## Examples

```
## L <- number of estimators
## M <- Number of dims to sample
## H <- Number of bins
## seed <- seed for random number generator
## verbose <- verbose level

library('FiRE')
data(sample_data) #Samples * Features

data(sample_label)
## Samples with label '1' represent abundant,
## while samples with label '2' represent rare.

## Saving rownames and colnames
rnames <- rownames(sample_data)
cnames <- colnames(sample_data)

## Converting data.frame to matrix
sample_data <- as.matrix(sample_data)
sample_label <- as.matrix(sample_label)

L <- 100 # Number of estimators
M <- 50 # Dims to be sampled

# Model creation without optional parameter
model <- new(FiRE::FiRE, L, M)

## There are 3 more optional parameters they can be passed as
## model <- new(FiRE::FiRE, L, M, H, seed, verbose)

## Hashing all samples
model$fit(sample_data)

## Computing score of each sample
rareness_score <- model$score(sample_data)

## Apply IQR-based criteria to identify rare samples for further downstream analysis.
q3 <- quantile(rareness_score, 0.75)
iqr <- IQR(rareness_score)
th <- q3 + (1.5*iqr)
```

```
## Select indexes that satisfy IQR-based thresholding criteria.
indIqr <- which(rareness_score >= th)

## Create a vector for binary predictions
predictions <- rep(1, dim(sample_data)[1])
predictions[indIqr] <- 2 #Replace predictions for rare samples with '2'.

## To access model parameters
## Parameters - generated weights
# model$w

## Parameters - sample dimensions
# model$d

## Parameters - generated thresholds
# model$ths

## Parameters - Bins
# model$b
```

---

FiRE *Constructor for Class*

---

## Description

Constructor to create an instance of FiRE class. There are two ways in which constructor can be created.

## Arguments

| | |
|---|---|
| L | Number of estimators. |
| M | Number of dimensions to be sampled per estimator. |
| H | Number of bins. Default=1017881 |
| seed | seed for random number generator. Default=0 |
| verbose | verbose level. Default=1 |

## Details

For usage see example.

## Examples

```
L <- 100
M <- 50
H <- 107881
seed <- 0
verbose <- 1
```

```
## Creating class object with required arguments
model <- new(FiRE::FiRE, L, M)

## Creating class object with all arguments
model <- new(FiRE::FiRE, L, M, H, seed, verbose)
```

---

`fit` *Hash samples in bins*

---

## Description

Hashing samples into bins, using Sketching as hash function. This function repeats hashing process `L` times by sampling `M` dimensions at a time and hashes samples into one of `H` bins.

## Arguments

| | |
|---|---|
| `data` | On which rarity score needs to be computed. Required to be a `matrix`. (can be converted to matrix with `as.matrix`) |

## Details

For usage see example.

## Note

This function does not do any preprocessing, so preprocessed data must be passed if required.

## Examples

```
## Not run:

   ## Creating class object with required arguments
   model <- new(FiRE::FiRE, L, M)
   model$fit(data)


## End(Not run)
```

## sample_data *Preprocessed 293T–Jurkat scRNA-seq dataset*

### Description

This data set has been generated from 293T and Jurkat cell data containing a total of ~3200 cells, with an almost equal number of representative transcriptomes of each type. The cells were mixed in vitro at equal proportions. Authors of the study resolved the cell types bioinformatically exploiting their SNV profiles. With this data, we mimicked the rare cell phenomenon by bioinformatically diluting Jurkat cell proportion to 2.5% in the data.

### Usage

```
data(sample_data)
```

### Format

A matrix containing 1580 observations (in rows), with 1000 features per observation (in columns).

### Source

10xgenomics, https://support.10xgenomics.com/single-cell-gene-expression/datasets

### References

Zheng, G. X. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* 8, 14049 (2017).

## sample_label *Labels of preprocessed 293T–Jurkat scRNA-seq dataset*

### Description

This contains labels for 1580 observations. There are two different clusters, one is abundant (denoted by 1) and the other is rare (denoted by 2). Rare cluster consists of Jurkat cells, which form 2.5% of the total population. The abundant cluster consists of 293T cells.

### Usage

```
data(sample_label)
```

### Format

A vector containing 1580 observations.

## Source

10xgenomics, https://support.10xgenomics.com/single-cell-gene-expression/datasets

## References

Zheng, G. X. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* 8, 14049 (2017).

---

`score` *Compute score on hashed sample.*

---

## Description

Once hashing is done using `fit`, compute score via proximity estimation.

## Arguments

| | |
|---|---|
| `data` | On which rarity score needs to be computed. |

## Details

For usage see example.

## Examples

```
## Not run:

   ## Creating class object with required arguments
   model <- new(FiRE::FiRE, L, M)
   model$fit(data)
   score <- model$score(data)


## End(Not run)
```

# Index