# Package 'SELEX'

November 28, 2022

**Title** Functions for analyzing SELEX-seq data

**Version** 1.31.0

**Date** 2019-09-01

**Author** Chaitanya Rastogi, Dahong Liu, Lucas Melo, and Harmen J. Bussemaker

**Depends** rJava (>= 0.5-0), Biostrings (>= 2.26.0)

**Imports** stats, utils

**SystemRequirements** Java (>= 1.5)

**Maintainer** Harmen J. Bussemaker <hjb2004@columbia.edu>

**Description** Tools for quantifying DNA binding specificities based on SELEX-seq data.

**License** GPL (>=2)

**biocViews** Software, MotifDiscovery, MotifAnnotation, GeneRegulation, Transcription

**URL** https://bussemakerlab.org/site/software/

**git_url** https://git.bioconductor.org/packages/SELEX

**git_branch** master

**git_last_commit** 5c5c6eb

**git_last_commit_date** 2022-11-01

**Date/Publication** 2022-11-28

## R topics documented:

---

SELEX *SELEX Package*

---

## Description

Functions to assist in discovering transcription factor DNA binding specificities from SELEX-seq experimental data according to the Slattery *et al.* paper. For a more comprehensive example, please look at the vignette. Sample data used in the Slattery, et. al. is stored in the `extdata` folder for the package, and can be accessed using either the base R function `system.file` or the package function `selex.exampledata`.

Functions available:

| | |
|---|---|
| `selex.affinities` | Construct a K-mer affinity table |
| `selex.config` | Set SELEX system parameters |
| `selex.counts` | Construct or retrieve a K-mer count table |
| `selex.countSummary` | Summarize available K-mer count tables |
| `selex.defineSample` | Define annotation for an individual sample |
| `selex.exampledata` | Extract example data files |
| `selex.fastqPSFM` | Construct a diagnostic PSFM for a FASTQ file |
| `selex.getAttributes` | Display sample handle attributes |
| `selex.getRound0` | Obtain round zero sample handle |
| `selex.getSeqfilter` | Display sequence filter attributes |

| | |
|---|---|
| selex.infogain | Compute or retrieve information gain between rounds |
| selex.infogainSummary | Summarize available information gain values |
| selex.jvmStatus | Display current JVM memory usage |
| selex.kmax | Calculate kmax for a dataset |
| selex.kmerPSFM | Construct a PSFM from a K-mer table |
| selex.loadAnnotation | Load a sample annotation file |
| selex.mm | Build or retrieve a Markov model |
| selex.mmProb | Compute prior probability of sequence using Markov model |
| selex.mmSummary | Summarize Markov model properties |
| selex.revcomp | Create forward-reverse complement data pairs |
| selex.run | Run a standard SELEX analysis |
| selex.sample | Create a sample handle |
| selex.sampleSummary | Show samples visible to the current SELEX session |
| selex.saveAnnotation | Save sample annotations to file |
| selex.seqfilter | Create a sequence filter |
| selex.setwd | Set or change the working directory |
| selex.split | Randomly split a dataset |
| selex.summary | Display all count table, Markov model, and information gain summaries |

## Details

| | |
|---|---|
| Package: | SELEX |
| Type: | Package |
| Version: | .99 |
| Date: | 2014-11-3 |
| License: | GPL |

## Author(s)

Chaitanya Rastogi, Dahong Liu, and Harmen Bussemaker

Maintainer: Harmen Bussemaker <hjb2004@columbia.edu>

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## Examples

```
#Initialize the SELEX package
#options(java.parameters="-Xmx1500M")
#library(SELEX)

# Configure the current session
workDir = file.path(".", "SELEX_workspace")
selex.config(workingDir=workDir,verbose=FALSE, maxThreadNumber= 4)

# Extract sample data from package, including XML database
sampleFiles = selex.exampledata(workDir)

# Load & display all sample files using XML database
selex.loadAnnotation(sampleFiles[3])
selex.sampleSummary()

# Create sample handles
r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)
r2 = selex.sample(seqName='R2.libraries', sampleName='ExdHox.R2', round=2)

# Split the r0 sample into testing and training sets
r0.split = selex.split(sample=r0)
r0.split

# Display all currently loaded samples
selex.sampleSummary()

# Find kmax on the test dataset
k = selex.kmax(sample=r0.split$test)

# Build the Markov model on the training dataset
mm = selex.mm(sample=r0.split$train, order=NA, crossValidationSample=r0.split$test)
# See Markov model R^2 values
selex.mmSummary()

# Kmer counting with an offset
t1 =  selex.counts(sample=r2, k=2, offset=14, markovModel=NULL)
# Kmer counting with a Markov model (produces expected counts)
t2 =  selex.counts(sample=r2, k=4, markovModel=mm)
# Display all available kmer statistics
selex.countSummary()

# Calculate information gain
ig =  selex.infogain(sample=r2, k=8, mm)
# View information gain results
selex.infogainSummary()

# Perform the default analysis
selex.run(trainingSample=r0.split$train, crossValidationSample=r0.split$test,
  infoGainSample=r2)

# View all stats
```

```
selex.summary()
```

---

`selex.affinities` *Construct a K-mer affinity table*

---

## Description

A function used to calculate and return the affinities and affinity standard errors of K-mers of length `k` for a given dataset in addition to all the output provided by `selex.counts`. A Markov model is necessary for evaluation.

## Usage

```
selex.affinities(sample, k, minCount=100, top=-1, numSort=TRUE, offset=NULL,
markovModel=NULL, seqfilter=NULL)
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle to the dataset on which K-mer counting should be perfomed. |
| `k` | K-mer length(s) to be counted. |
| `minCount` | The minimum number of counts for a K-mer to be output. |
| `top` | Give the first N K-mers (by count). |
| `numSort` | Sort K-mers in descending order by count. If `FALSE`, K-mers are sorted alphabetically. |
| `offset` | Location of window for which K-mers should be counted for. If not provided, K-mers are counted across all windows. |
| `markovModel` | Markov model handle to use to predict previous round probabilities and expected counts. |
| `seqfilter` | A sequence filter object to include/exclude sequences that are read in from the FASTQ file. |

## Details

When a new `seqfilter` object is provided, K-mer counting and affinity table construction is redone. See `selex.seqfilter` for more details.

See 'References' for more details regarding K-mer counting and affinity calculation.

## Value

`selex.affinities` returns a data frame containing the K-mer sequence, observed counts, predicted prior observation probability, predicted prior observed counts, affinities, and standard errors.

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

`selex.counts`, `selex.infogain`, `selex.kmax`, `selex.mm`

## Examples

```
r2Aff = selex.affinities(sample=r2, k=10, markovModel=mm)
```

---

# selex.config *Set SELEX system parameters*

---

## Description

A function used to set system parameters for the SELEX package. These parameters are stored as global options, which are checked at runtime. They can be permanently set in a .Rprofile file, or changed temporarily by the user by invoking `selex.config`. **It is important to set** `java.parameters` **before loading the SELEX library**. See 'Details' for more information.

## Usage

```
selex.config(workingDir=NULL, verbose=NULL, maxThreadNumber=NULL)
```

## Arguments

| | |
|---|---|
| `workingDir` | Physical location on disk for the current working directory. **The full system path must be specified.** |
| `verbose` | Print more output to terminal. Useful for debugging. |
| `maxThreadNumber` | The maximum number of threads to be used while K-mer counting. If unspecified, 4 threads will be used. |

## Details

The working directory is used to store the output of all analyses performed by the selex package. If a certain calculation has already been performed in the given working directory, the previously computed values are returned. If the specified path does not exist, `selex.config` will attempt to create it. By default, a temporary directory is used.

It is important to set the Java memory options before loading the SELEX library to allocate enough memory to the JVM. You can allocate memory by calling `options` and setting `java.parameters`. `-Xmx` must prefix the memory value; the code below allocates 1500 MB of RAM:

```
options(java.parameters="-Xmx1500M")
```

If the memory option is unspecified, the default JVM memory setting will be used.

## Value

Not applicable

## See Also

`selex.jvmStatus`, `selex.setwd`

## Examples

```
## Initialize SELEX
#options(java.parameters="-Xmx1500M")
#library(SELEX)

## Set working directory and verbose to true
workDir = file.path(".", "SELEX_workspace")
selex.config(workingDir=workDir,verbose=TRUE, maxThreadNumber=4)
```

---

# selex.counts *Construct or retrieve a K-mer count table*

---

## Description

A function used to count and return the number of instances K-mers of length `k` appear within the `sample`'s variable regions. If an `offset` value is provided, K-mer counting takes place at a fixed position within the variable region of the read. If a Markov model is supplied, the expected count and the probability of observing the K-mer are also returned.

## Usage

```
selex.counts(sample, k, minCount=100, top=-1, numSort=TRUE, offset=NULL,
  markovModel=NULL, forceCalculation=FALSE, seqfilter=NULL, outputPath = "")
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle to the dataset on which K-mer counting should be perfomed. |
| `k` | K-mer length(s) to be counted. |
| `minCount` | The minimum number of counts for a K-mer to be output. |
| `top` | Give the first N K-mers (by count). |

| | |
|---|---|
| numSort | Sort K-mers in descending order by count. If FALSE, K-mers are sorted in ascending order. |
| offset | Location of window for which K-mers should be counted for. If not provided, K-mers are counted across all windows. |
| markovModel | Markov model handle to use to predict previous round probabilities and expected counts. |
| forceCalculation | Forces K-mer counting to be performed again, even if a previous result exists. |
| seqfilter | A sequence filter object to include/exclude sequences that are read in from the FASTQ file. |
| outputPath | Prints the computed K-mer table to a plain text file. This is useful when the number of unique K-mers in the dataset exceeds R's memory limit. |

## Details

The offset feature counts K-mers of length k offset bp away from the 5' end in the variable region. For example, if we have 16-mer variable regions and wish to count K-mers of length 12 at an offset of 3 bp, we are looking at the K-mers found only at the position indicated by the bolded nucleotides in the variable region:

5' NNN**NNNNNNNNNNNN**N 3'

Minimum count refers to the lowest count observed for a kmer of length k for a given sample. Total count is the sum of counts over all kmers of length k for a given sample. These statistics can be viewed for all K-mer lengths and samples counting was performed on using selex.countSummary. When a new seqfilter object is provided, K-mer counting is redone. See selex.seqfilter for more details.

See 'References' for more details regarding the K-mer counting process.

## Value

selex.counts returns a data frame containing the K-mer sequence and observed counts for a given sample if a Markov model has not been supplied.

If a Markov model is supplied, a data frame containing K-mer sequence, observed counts, predicted previous round probability, and predicted previous round expected counts is returned.

If the number of unique K-mers exceeds R's memory limit, selex.counts will cause R to crash when returning a data frame containing the K-mers. The outputPath option can be used to avoid such a situation, as the Java code will directly write the table to a plain text file at the specified location instead.

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

selex.affinities, selex.countSummary, selex.infogain, selex.kmax, selex.mm, selex.run

## Examples

```
# Kmer counting for a specific length on a given dataset
t1 = selex.counts(sample=r2, k=8, minCount=1)

# Kmer counting with an offset
t2 = selex.counts(sample=r2, k=2, offset=14, markovModel=NULL)

# Kmer counting with a Markov model (produces expected counts)
t3 = selex.counts(sample=r2, k=4, markovModel=mm)

# Display all available kmer statistics
selex.countSummary()
```

---

# selex.countSummary *Summarize available K-mer count tables*

---

## Description

This function returns the sample, kmer length, offset, minimum/maximum count, total count, and applied filters for either all count tables or a specified sample in the current working directory.

## Usage

```
selex.countSummary(sample=NULL, displayFilter=FALSE)
```

## Arguments

| | |
|---|---|
| sample | A sample handle to the sample for which K-mer count statistics should be returned. |
| displayFilter | Display all applied sequence filter attributes. |

## Details

Minimum count refers to the lowest count observed for a kmer of length k for a given sample. Total count is the sum of counts over all kmers of length k for a given sample.

## Value

selex.countSummary returns a data frame containing the sample, kmer length, offset, minimum/maximum count, total count, and applied filters for all count tables in the current working directory. If sample is provided, the above information is given for the specific sample only.

## See Also

selex.counts, selex.summary

## Examples

```
selex.countSummary()
```

---

selex.defineSample *Define annotation for an individual sample*

---

## Description

A function used to manually load SELEX sample metadata and make it visible to the current session, which can then be used to create a sample handle (see selex.sample. It is functionally identical to selex.loadAnnotation, except constrained to take manual inputs for a single sample.

## Usage

```
selex.defineSample(seqName, seqFile=NULL, sampleName, round, varLength,
  leftBarcode, rightBarcode, leftFlank=NULL, rightFlank=NULL,
  round0SeqName=NULL, round0SampleName=NULL)
```

## Arguments

| | |
|---|---|
| seqName | Desired sequencing run name |
| seqFile | Path to the FASTQ file containing the sample. Can be NULL if seqFile has been previously specified for the seqName. |
| sampleName | Desired sample name |
| round | Sample round |
| varLength | Length of the variable region |
| leftBarcode | Left barcode sequence |
| rightBarcode | Right barcode sequence |
| leftFlank | Left flank sequence |
| rightFlank | Right flank sequence |
| round0SeqName | The sequencing run name of the round 0 data associated with this sample |
| round0SampleName | The sample name of the round 0 data associated with this sample |

### Details

selex.defineSample should be used for rapid testing or when it is not worthwhile to generate a generate a sample annotation file. Unlike selex.loadAnnotation, the unique name requirement is relaxed, requiring only unique seqName, seqFile, and round combinations. Only one seqFile can be specified for a given seqName; for example, the following will throw an error:

```
selex.defineSample('Seq1', 'Seq1.fastq.gz', round=1, rightBarcode='CCAGCTG', ...)
selex.defineSample('Seq1', 'Seq2.fastq.gz', round=2, rightBarcode='CCAGCTG', ...)
```

However, seqFile can be omitted if a new sample is being specified with the same seqName:

```
selex.defineSample('Seq1', 'Seq1.fastq.gz', round=1, rightBarcode='CCAGCTG', ...)
selex.defineSample('Seq1', NULL, round=1, rightBarcode='CCACGTC', ...)
```

The sequencing run name and sample name of the round 0 file associated with a later-round sample can be provided to keep samples and their random pools in order.

### Value

Not applicable

### See Also

selex.getAttributes, selex.loadAnnotation, selex.sample, selex.sampleSummary, selex.saveAnnotation

### Examples

```
selex.defineSample(seqName='R0.libraries', seqFile=exampleFiles[1],
                sampleName='R0.barcodeGC', round = 0, varLength = 16,
                leftBarcode = 'TGG', rightBarcode= 'CCAGCTG')
```

---

## selex.exampledata *Extract example data files*

---

### Description

A function used to extract the sample files embedded in this package. The package contains 2 FASTQ files (R0.fastq.gz and ExdHox.R2.fastq.gz, and an XML configuration file (config.xml).

### Usage

```
selex.exampledata(outputFolder)
```

### Arguments

| | |
|---|---|
| outputFolder | Location on disk where the files should be extracted. |

## Value

Not applicable

## See Also

selex.defineSample

## Examples

```
#Initialize the SELEX package
#options(java.parameters="-Xmx1500M")
#library(SELEX)

# Configure the current session
workDir = file.path(".", "SELEX_workspace")
selex.config(workingDir=workDir,verbose=FALSE, maxThreadNumber= 4)

# Extract sample data from package, including XML database
exampleFiles = selex.exampledata(workDir)
```

---

# selex.fastqPSFM *Construct a diagnostic PSFM for a FASTQ file*

---

## Description

A function used to calculate and return the PFSM (Position Specific Frequency Matrix) for an entire FASTQ file, regardless of barcode or other sequence filtering.

## Usage

```
selex.fastqPSFM(seqName)
```

## Arguments

| | |
|---|---|
| seqName | A sequencing run name for the desired FASTQ file; this must match a sequencing run name of a sample currently visible to the SELEX session. |

## Details

The output can be used by the seqLogo package to create a sequence logo.

## Value

selex.fastqPSFM returns a matrix containing the frequences for each base at every position.

## See Also

selex.kmerPSFM

## Examples

```
# Display all currently loaded samples
selex.sampleSummary()

# Make PSFMs for the two visible FASTQ files:
psfm1 = selex.fastqPSFM(seqName='R0.libraries')
psfm2 = selex.fastqPSFM(seqName='R2.libraries')

# Can make sequence logos using the seqLogo package:
#library(seqLogo)
#seqLogo(makePWM(t(psfm1)))
```

---

# selex.getAttributes *Display sample handle properties*

---

## Description

A function used to output sample handle's sequencing run name, sample name, round, left and right barcode, file path, and variable region length of the sample handle.

## Usage

```
selex.getAttributes(sample)
```

## Arguments

| | |
|---|---|
| sample | A sample handle. |

## Value

`selex.getAttributes` returns a data frame containing the sequencing run name, sample name, round, left and right barcode, file path, and variable region length of the sample handle.

## See Also

`selex.defineSample`, `selex.loadAnnotation`, `selex.sample`, `selex.sampleSummary`

## Examples

```
# Create a sample handle
r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)

# Use the sample handle to display sample properties
selex.getAttributes(r0)
```

## selex.getRound0 *Obtain round zero sample handle*

### Description

A function used to return the sample handle of the round zero file associated with the input `sample`.

### Usage

```
selex.getRound0(sample)
```

### Arguments

| | |
|---|---|
| `sample` | A sample handle. |

### Value

`selex.getRound0` returns a sample handle to the corresponding round zero file. The latter needs to be defined in the annotation table. If not, an error will be generated.

### Examples

```
#Show currently visible samples
selex.sampleSummary()

#Return the matched roound zero file
r2 = selex.sample(seqName='R2.libraries', sampleName='ExdHox.R2', round=2)
r0 = selex.getRound0(r2)
selex.getAttributes(r0)
```

## selex.getSeqfilter *Display sequence filter attributes*

### Description

Display all regular expressions used in a sequence filter object.

### Usage

```
selex.getSeqfilter(regex)
```

### Arguments

| | |
|---|---|
| `regex` | A sequence filter object. |

## Value

A character string containing the regular expressions used in a sequence filter object.

## See Also

selex.seqfilter

## Examples

```
selex.getSeqfilter(regex)
```

---

selex.infogain *Compute or retrieve information gain between rounds*

---

## Description

A function used to compute and store the information gain for various K-mer lengths on sample using markovModel to predict prior probabilities.

## Usage

```
selex.infogain(sample, k=NULL, markovModel, seqfilter=NULL,
  checkBarcode=TRUE)
```

## Arguments

| | |
|---|---|
| sample | A sample handle to the dataset on which to perform the analysis. |
| k | The range of K-mer lengths for which the information gain should be calculated. If NA, the range is automatically set to start from the optimal Markov model order + 1 to the length of the variable region. |
| markovModel | A Markov model handle. |
| seqfilter | A sequence filter object to include/exclude sequences that are read in from the FASTQ file. |
| checkBarcode | Checks to see if the sample barcodes used to construct the Markov model match those in the current sample, and prevents computation if they do not match. |

## Details

selex.infogainSummary is required to view the computed information gain values. When a new seqfilter object is provided, the information gain analysis is redone. See selex.seqfilter for more details.

See 'References' for more on the computation of information gain values.

## Value

selex.infogain returns the highest information gain value.

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

selex.infogainSummary, selex.mm, selex.run

## Examples

```
# Calculate information gain for a fixed range
ig1 =  selex.infogain(sample=r2, k=c(8:10), markovModel=mm)

# View the results
selex.infogainSummary()[,c(1,2,3,4,5)]

# Now calculate for the default range
ig2 = selex.infogain(sample=r2, markovModel=mm)

# View the results again
selex.infogainSummary()[,c(1,2,3,4,5)]
```

---

# selex.infogainSummary *Summarize available information gain values*

---

## Description

This function returns the sample, Kmer length, information gain value, Markov model/type, and applied filters for either all computed information gain values or a specified sample in the current working directory.

## Usage

```
selex.infogainSummary(sample=NULL, displayFilter=FALSE)
```

## Arguments

| | |
|---|---|
| sample | A sample handle to the sample for which information gain statistics should be returned. |
| displayFilter | Display all applied sequence filter attributes. |

## Value

selex.infogainSummary returns a data frame containing the sample, Kmer length, information gain value, Markov model/type, and applied filters for all computed information gain values in the current working directory. If sample is provided, the above information is given for the specific sample only.

## See Also

selex.infogain, selex.summary

## Examples

```
selex.infogainSummary()
```

---

# selex.jvmStatus *Display current JVM memory usage*

---

## Description

A function that displays the current JVM memory usage.

## Usage

```
selex.jvmStatus()
```

## Details

selex.jvmStatus is useful for verifying the proper installaztion and initialization of rJava. Setting verbose=FALSE in selex.config will disable terminal ouput and a call to selex.jvmStatus will display nothing. If the current JVM memory allocation is suboptimal, settings can be changed using the memSize option in selex.config.

## Value

selex.jvmStatus does not return a value.

## See Also

selex.config

## Examples

```
selex.jvmStatus()
```

`selex.kmax` *Calculate kmax for a dataset*

## Description

This function returns the longest oligonucleotide length k such that all DNA sequences of length k ('K-mers') are found at least a minimum count number of times for the given sample.

## Usage

```
selex.kmax(sample, threshold=100, seqfilter=NULL)
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle. |
| `threshold` | The minimum count to be used. |
| `seqfilter` | A sequence filter object to include/exclude sequences that are read in from the FASTQ file. |

## Details

The kmax value is used to build the Markov model training and cross-validation datasets. While `selex.mm` contains a default kmax constructor, running `selex.kmax` can be useful in building analysis-specific Markov models.

`selex.kmax` discovers the kmax value by building K-mer count tables; after completion, the K-mer count tables can be viewed using `selex.counts`. When a new `seqfilter` object is provided, the kmax value is recomputed. See `selex.seqfilter` for more details.

## Value

`selex.kmax` returns the kmax value.

## See Also

`selex.counts`, `selex.mm`, `selex.sample`, `selex.seqfilter`

## Examples

```
kmax = selex.kmax(sample=r0, threshold=50)
```

`selex.kmerPSFM` *Construct a PSFM from a K-mer table*

## Description

A function used to calculate and return the PFSM (Position Specific Frequency Matrix) for a K-mer table of length `k` from `sample`. If an `offset` value is provided, K-mer counting takes place at a fixed position within the variable region.

## Usage

```
selex.kmerPSFM(sample, k, offset=NULL)
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle to the dataset on which K-mer counting should be perfomed. |
| `k` | K-mer length(s) to be counted. |
| `offset` | Location of window for which K-mers should be counted for. If not provided, K-mers are counted across all windows. `offset` starts from 0. |

## Details

A K-mer table will be constructed for the specified `sample`, length `k`, and `offset` if it does not already exist.

The output can be used by the `seqLogo` package to create a sequence logo.

## Value

`selex.kmerPSFM` returns a matrix containing the frequences for each base at every position.

## See Also

`selex.counts`, `selex.fastqPSFM`

## Examples

```
# Build the PSFM
psfm1 = selex.kmerPSFM(sample=r0, k=8, 0)

# Can make sequence logos using the seqLogo package:
#library(seqLogo)
#seqLogo(makePWM(t(psfm1)))
```

`selex.loadAnnotation` *Load a sample annotation file*

## Description

A function used to load sample metadata contained within a sample annotation file and make it visible to the current SELEX session. These samples can then be used to create sample handles (see `selex.sample`).

## Usage

```
selex.loadAnnotation(config_path, data_folder=NULL)
```

## Arguments

| | |
|---|---|
| `config_path` | Location on disk to the sample annotation file. |
| `data_folder` | Location on disk where FASTQ sample files are stored. This is either an absolute path, or relative to the location of the annotation file. If unspecified, it uses the parent folder of the annotation file. |

## Details

A sample annotation file is an XML file that acts as a database storing metadata for different SELEX experiments. Here, a SELEX experiment refers to a single SELEX round that has been sequenced. Such a database allows the user to explicitly store all relevant information in a structured manner for easy future access.

A sample annotation file is provided below. Every annotation file can contain multiple `SequencingRunInfo` instances; every instance within an annotation file must contain a unique name. If multiple annotation files are used in a given SELEX session, all such names must be unique. For example, the following annotation files A and B

File A
`<SequencingRunInfo name="exdUbx.Run1">` and
`<SequencingRunInfo name="exdUbx.Run2">`

File B
`<SequencingRunInfo name="exdUbx.Run1">`

have a legal naming system if **either** File A or File B is used in a single SELEX session, but have an invalid naming system if **both** are used. In general, it is a good idea to ensure that every `SequencingRunInfo` name is unique. Every `SequencingRunInfo` instance references a single FASTQ file. The user has the option of providing additional metadata regarding the FASTQ file.

A `SequencingRunInfo` instance can contain multiple `Samples`. Every `Sample` name within a `SequencingRunInfo` instance must contain a unique name **and** round combination. For example,

`<Sample name="exdLab", round="0">` and
`<Sample name="exdLab", round="1">`

is a valid name `Sample` name combinations while

`<Sample name="exdLab", round="0">` and
`<Sample name="exdLab", round="0">`

is not. Non Round 0 `Samples` have the option of referencing a Round 0 file, working as a checking mechanism to prevent the wrong Round 0 sample from being used to analyze a later round sample.

Once samples have been loaded into the current SELEX session, a sample handle can be generated using the `SequencingRunInfo` name, `Sample` name, and `Round` number. Sample handles make it easier to reference individual `Samples` while running an analysis. See `selex.sample` for more information.

## Value

Not applicable

## Note

Sample annotation files are structured as follows:

<?xml version="1.0" encoding="UTF-8"?>

<SELEXSequencingConfig xmlns:xsi="http://www.columbia.edu/" xsi:noNamespaceSchemaLocation="selex.xsd">

<SequencingRunInfo name="exdUbx.exdScr.0"> <!– information needed for differentiating multiple sequencing info instances –>

<DataFile>/Users/Documents/Data/Run1.fastq.gz</DataFile> <!– absolute or relative path –>
<SequencingPlatform>Illumina</SequencingPlatform> <!– #optional –>
<ResearcherName>John Smith</ResearcherName> <!– #optional –>
<ResearcherEmail>jsmith@columbia.edu</ResearcherEmail> <!– #optional –>
<SequencingFacilityName>Columbia University Genome Center</SequencingFacilityName> <!– #optional –>
<SequencingFacilityEmail>cugc@columbia.edu</SequencingFacilityEmail> <!– #optional –>
<Description>Ubx/Scr Round 0 Probes</Description> <!– #optional –>
<Notes>Our first SELEX Run</Notes> <!– #optional –>

<Sample name="barcodeCCAGCTG.v1" round="0">
<Protein>Probes</Protein>
<Concentration></Concentration> <!– #optional –>
<VariableRegionLength>16</VariableRegionLength>
<LeftFlank>GTTCAGAGTTCTACAGTCCGACGATCTGG</LeftFlank>
<RightFlank>CCAGCTGTCGTATGCCGTCTTCTGCTTG</RightFlank>
<LeftBarcode>TGG</LeftBarcode>

```
<RightBarcode>CCAGCTG</RightBarcode>
<Round0></Round0>
<Notes></Notes> <!– #optional –>
</Sample>

<Sample name="barcodeCCACGTC.v1" round="0">
<Protein>Probes</Protein>
<Concentration></Concentration>
<VariableRegionLength>16</VariableRegionLength>
<LeftFlank>GTTCAGAGTTCTACAGTCCGACGATCTGG</LeftFlank>
<RightFlank>CCACGTCTCGTATGCCGTCTTCTGCTTG</RightFlank>
<LeftBarcode>TGG</LeftBarcode>
<RightBarcode>CCACGTC</RightBarcode>
<Round0></Round0>
<Notes></Notes>
</Sample>

</SequencingRunInfo>

<!– #New FASTQ file below –>

<SequencingRunInfo name="exdUbx.exdScr.L.2">

<DataFile>/Users/Documents/Data/Run2.fastq.gz</DataFile>
<SequencingPlatform>Illumina</SequencingPlatform>
<ResearcherName>John Smith</ResearcherName>
<ResearcherEmail>jsmith@columbia.edu</ResearcherEmail>
<SequencingFacilityName>Columbia University Genome Center</SequencingFacilityName>
<SequencingFacilityEmail>cugc@columbia.edu</SequencingFacilityEmail>
<Description>Ubx/Scr Round 2</Description>
<Notes>Our first SELEX Run</Notes>

<Sample name="barcodeCCAGCTG.v1.low" round="2">
<Protein>hmExdUbx</Protein>
<Concentration>low</Concentration>
<VariableRegionLength>16</VariableRegionLength>
<LeftFlank>GTTCAGAGTTCTACAGTCCGACGATCTGG</LeftFlank>
<RightFlank>CCAGCTGTCGTATGCCGTCTTCTGCTTG</RightFlank>
<LeftBarcode>TGG</LeftBarcode>
<RightBarcode>CCAGCTG</RightBarcode>
<Round0 sequencingName="exdUbx.exdScr.0" sampleName="barcodeCCAGCTG.v1"/>
<Notes></Notes>
</Sample>

</SequencingRunInfo>

</SELEXSequencingConfig>
```

## See Also

selex.defineSample, selex.getAttributes, selex.sample, selex.sampleSummary, selex.saveAnnotation

## Examples

```
#Initialize the SELEX package
#options(java.parameters="-Xmx1500M")
#library(SELEX)

# Configure the current session
workDir = file.path(".", "SELEX_workspace")
selex.config(workingDir=workDir,verbose=FALSE, maxThreadNumber= 4)

# Extract sample data from package, including XML database
sampleFiles = selex.exampledata(workDir)

# Load & display all sample files using XML database
selex.loadAnnotation(sampleFiles[3])
selex.sampleSummary()

# Create a sample handle
r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)

# Use the sample handle to display sample properties
selex.getAttributes(r0)
```

---

# selex.mm *Build or retrieve a Markov model*

---

## Description

A function used to compute and store a Markov model built by training and cross-validating on specified files. It returns a Markov model handle to conveniently reference the calculated model in other SELEX functions.

## Usage

```
selex.mm(sample, order=NA, crossValidationSample=NULL, Kmax= NULL,
  seqfilter=NULL, mmMethod="DIVISION", mmWithLeftFlank=FALSE)
```

## Arguments

| | |
|---|---|
| sample | A sample handle to the training dataset. |
| order | The order of the Markov model to be built. If NA, builds Markov models of all orders from 0 to Kmax-1, selecting the one with highest R^2. |
| crossValidationSample | A sample handle to the cross-validation dataset. If NULL, no R^2 value will be provided. |

| | |
|---|---|
| Kmax | The K-mer length to determine model fit on. If NA, automatically finds kmax with a minimum count set to 100. See selex.kmax and 'Details'. |
| seqfilter | A sequence filter object to include/exclude sequences that are read in from the FASTQ file. |
| mmMethod | A character string indicating the algorithm used to evaluate the Markov model conditional probabilities. Can be either "DIVISION" (default) or "TRANSITION". See 'Details'. |
| mmWithLeftFlank | Predict expected counts by considering the sequences in the left flank of the variable region. See 'Details'. |

## Details

selex.mm builds an N-th order Markov model by training the model on the sample dataset and tests its accuracy by attempting to predict the K-mer counts of the cross-validation dataset. This K-mer length is determined either by setting Kmax or by an internal call to selex.kmax. If a cross-validation dataset does not exist, selex.split can be used to partition a dataset into testing and training datasets.

The Markov model uses conditional probabilities to predict the probability of observing a given K-mer. For example, let us consider the following 6-mer sequence:

AGGCTA

If a fourth-order Markov model were to predict the prior observation probability of the above sequence, the following would have to be evaluated:

$$p(AGGCTA) = p(AGG)\ p(C|AGG)\ p(T|GGC)\ p(A|GCT)$$

These conditional probabilities can be evaluated using one of two algorithms. The TRANSITION algorithm relies on K-mer counts of length equal to the order of the Markov model. For the example above,

$$p(C|AGG) = \frac{count(AGGC)}{count(AGGA) + count(AGGC) + count(AGGG) + count(AGGT)}$$

The DIVISION algorithm relies on K-mer frequencies for lengths equal to Markov model order and order-1. For the example above,

$$p(C|AGG) = \frac{P(AGGC)}{P(AGG)} = \frac{\frac{count(AGGC)}{total4-mercounts}}{\frac{(count(AGG)}{total3-mercounts}}$$

The the flanking sequences in the left flank of the variable region can be taken into consideration when predicting prior observation probabilities by setting mmWithLeftFlank to TRUE. If the left barcode of the sequence in the example above is TGG, P(AGG) in the prior probability becomes

$$p(AGG) = p(A|TGG)\ p(G|GGA)\ p(G|GAG).$$

After computation, the Markov model is stored in the working directory. When a new seqfilter object is provided, the Markov model is reconstructed. See selex.seqfilter for more details. See

'References' for more details regarding the model construction process.

`selex.mmSummary` can be used to view the R^2 values for all orders that have been computed for the Markov model. If `crossValidationSample` is `NULL`, the resulting Markov model will not be displayed by `selex.mmSummary`.

## Value

`selex.mm` returns a Markov model handle.

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

`selex.counts`, `selex.infogain`, `selex.kmax`, `selex.mmSummary`, `selex.run`, `selex.split`

## Examples

```
mm = selex.mm(sample=r0.split$train, order=NA, crossValidationSample=r0.split$test)

# View Markov model R^2 values
selex.mmSummary()
```

---

# selex.mmProb *Compute prior probability of sequence using Markov model*

---

## Description

A function used to calculate and return the prior observation probability of a DNA sequence `seqStr` as predicted by Markov model `markovModel`.

## Usage

```
selex.mmProb(seqStr, markovModel)
```

## Arguments

| | |
|---|---|
| `seqStr` | A character string representing the DNA sequence to be evaluated. |
| `markovModel` | A Markov model handle. |

## Value

`selex.mm` returns the prior observation probability of `seqStr`.

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

`selex.mmProb`

## Examples

```
# Build the Markov model on the training dataset
mm = selex.mm(sample=r0.split$train, order=NA, crossValidationSample=r0.split$test)

# Compute expected Markov model probability value for a random 12-mer
selex.mmProb(seqStr="ATTGCAGACTTG", markovModel=mm)
```

---

# selex.mmSummary *Summarize Markov model properties*

---

## Description

This function returns sample, order, Markov model type, R^2 value, cross validation sample/length, and applied filters for either all computed Markov models or a specified sample in the current working directory.

## Usage

```
selex.mmSummary(sample=NULL, displayFilter=FALSE)
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle to the sample for which Markov model information should be returned. |
| `displayFilter` | Display all applied sequence filter attributes. |

### Value

`selex.mmSummary` returns a data frame containing the sample, order, Markov model type, R^2 value, cross validation sample/length, and applied filters for all computed Markov models in the current working directory. If `sample` is provided, the above information is given for the specific sample only.

### See Also

`selex.mm`, `selex.summary`

### Examples

```
selex.mmSummary()
```

---

## selex.revcomp *Create forward-reverse complement data pairs*

---

### Description

A function used to find and return the reverse complement of K-mers and the values associated with them. It is useful for calculating forward/reverse complement symmetrized values.

### Usage

```
selex.revcomp(kmer,value)
```

### Arguments

| | |
|---|---|
| `kmer` | A string array representing K-mers. |
| `value` | An array of associated values. |

### Details

`selex.revcomp` finds and returns the reverse complement and associated value of every input K-mer, if it exists. If a reverse complement does not exist for a given K-mer, it is removed from the output. For example, consider the following K-mer and value arrays:

| | |
|---|---|
| ACGT | .34 |
| GCTA | .22 |
| CGAC | .98 |
| TAGC | .19 |

The output of `selex.revcomp` will be:

| | | | |
|---|---|---|---|
| ACGT | .34 | ACGT | .34 |
| GCTA | .22 | TAGC | .19 |

TAGC .19 GCTA .22

## Value

selex.revcomp returns a data frame containing the original K-mers and values, along with their reverse complements and associated values.

## See Also

selex.affinities, selex.counts

## Examples

```
# Find round 2 affinities
r2Aff = selex.affinities(sample=r2, k=10, markovModel=mm)

# Find the reverse complement affinities and standard errors
Aff = selex.revcomp(kmer=r2Aff$Kmer, value=r2Aff$Affinity)
SE = selex.revcomp(kmer=r2Aff$Kmer, value=r2Aff$SE)

# Find the forward/reverse complement symmetrized Affinity and SE values
symAff = (Aff$Value+Aff$Reverse.Complement.Values)/2
symSE = sqrt((SE$Value^2+SE$Reverse.Complement.Values^2)/2)

# Final Result
final = data.frame(Kmer=Aff$Kmer, Affinity=Aff$Value,
  SymmetrizedAffinity=symAff/max(symAff), SE=SE$Value,
  SymmetrizedSE=symSE/max(symAff))
final = final[order(-final$SymmetrizedAffinity),]
```

---

# selex.run *Run a standard SELEX analysis*

---

## Description

A function used to, in one shot,

- Determine kmax on the crossValidationSample with the minimum count determined by minCount
- Build a Markov model on the trainingSample and test it on the crossValidationSample with kmax length K-mers used to determine model fit, and constructed using mmMethod
- Calculate information gain for infoRange K-mer lengths on the infoGainSample, using the Markov model order with the highest R^2 to predict previous round values.

## Usage

```
selex.run(trainingSample, crossValidationSample, minCount=100, infoGainSample,
  infoRange=NULL, mmMethod="DIVISION", mmWithLeftFlank=FALSE)
```

## Arguments

| | |
|---|---|
| trainingSample | A sample handle to the training dataset. |
| crossValidationSample | A sample handle to the cross-validation dataset. |
| minCount | The minimum count to be used. |
| infoGainSample | A sample handle to the dataset on which to perform the information gain analysis. |
| infoRange | The range of K-mer lengths for which the information gain should be calculated. If NULL, the range is automatically set to start from the optimal Markov model order + 1 to the length of the variable region. This is the same as k in selex.infogain. |
| mmMethod | A character string indicating the algorithm used to evaluate the Markov model conditional probabilities. Can be either "DIVISION" (default) or "TRANSITION". |
| mmWithLeftFlank | Predict expected counts by considering the sequences in the left flank of the variable region. |

## Details

Please see the individual functions or 'References' for more details.

## Value

Not applicable

## References

Slattery, M., Riley, T.R., Liu, P., Abe, N., Gomez-Alcala, P., Dror, I., Zhou, T., Rohs, R., Honig, B., Bussemaker, H.J.,and Mann, R.S. (2011) Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell 147:1270–1282.

Riley, T.R., Slattery, M., Abe, N., Rastogi, C., Liu, D., Mann, R.S., and Bussemaker, H.J. (2014) SELEX-seq: a method for characterizing the complete repertoire of binding site preferences for transcription factor complexes. Methods Mol. Biol. 1196:255–278.

## See Also

selex.counts, selex.countSummary, selex.infogain, selex.infogainSummary, selex.mm, selex.mmSummary

## Examples

```
#Initialize the SELEX package
#options(java.parameters="-Xmx1500M")
#library(SELEX)

# Configure the current session
workDir = file.path(".", "SELEX_workspace")
selex.config(workingDir=workDir,verbose=FALSE, maxThreadNumber= 4)
```

```
# Extract sample data from package, including XML database
sampleFiles = selex.exampledata(workDir)

# Load all sample files using XML database
selex.loadAnnotation(sampleFiles[3])

# Create sample handles
r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)
r2 = selex.sample(seqName='R2.libraries', sampleName='ExdHox.R2', round=2)

# Split the r0 sample into testing and training datasets
r0.split = selex.split(sample=r0)

# Run entire analysis
selex.run(trainingSample=r0.split$train, crossValidationSample=r0.split$test,
  infoGainSample=r2)

# Display results
selex.mmSummary()[,c(1,2,3,4,5,6)]
selex.infogainSummary()[,c(1,2,3,4,5)]
```

---

`selex.sample` *Create a sample handle*

---

## Description

A function used to create a sample handle for conveniently referencing visible samples in other SELEX functions.

## Usage

```
selex.sample(seqName, sampleName, round, index=NULL)
```

## Arguments

| | |
|---|---|
| `seqName` | Sequencing run name |
| `sampleName` | Sample name |
| `round` | Sample round |
| `index` | Row number of desired sample when using `selex.sampleSummary`. When used, overrides other variables. |

## Details

A sample is considered 'visible' to the current SELEX session when its metadata has been loaded or defined and the sample's sequencing run info name, sample name, and round is displayed when `selex.sampleSummary` is called. Alternatively, a row number from `selex.sampleSummary` can be used to reference a sample.

A sample handle is an object that contains a reference to one visible sample. This object can be passed to SELEX functions instead of explicitly passing all sample information (such as variable region length, barcodes, round, FASTQ file location, etc).

## Value

`selex.sample` returns a sample handle object.

## See Also

`selex.defineSample`, `selex.getAttributes`, `selex.loadAnnotation`, `selex.sampleSummary`

## Examples

```
r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)
r2 = selex.sample(index=3)
```

---

# selex.samplePSFM *Construct a diagnostic PSFM for a FASTQ file*

---

## Description

A function used to calculate and return the PFSM (Position Specific Frequency Matrix) for a sample, regardless of sequence filtering.

## Usage

```
selex.samplePSFM(sample)
```

## Arguments

| | |
|---|---|
| `sample` | A sample handle to the dataset on which the PSFM is calculated. |

## Details

The output can be used by the `seqLogo` package to create a sequence logo.

## Value

`selex.samplePSFM` returns a matrix containing the frequences for each base at every position.

## See Also

`selex.kmerPSFM`

## Examples

```
# Display all currently loaded samples
selex.sampleSummary()

r0 = selex.sample(seqName="R0.libraries", sampleName="R0.barcodeGC", round=0)

# Make PSFMs for the two visible FASTQ files:
psfm1 = selex.samplePSFM(r0)

# Can make sequence logos using the seqLogo package:
#library(seqLogo)
#seqLogo(makePWM(t(psfm1)))
```

---

`selex.sampleSummary` *Show samples visible to the current SELEX session*

---

## Description

A function used to output the sequencing run names, sample names, rounds, left and right barcodes, and file paths of all samples that have been currently loaded from sequencing run info files or defined using `selex.defineSample`.

## Usage

```
selex.sampleSummary()
```

## Details

`selex.sampleSummary` displays all the samples currently visible to the SELEX session. A sample is 'visible' to the current SELEX session when its metadata has been loaded or defined and can be used to create a sample handle from its sequencing run info name, sample name, and round (see `selex.sample`). Alternatively, a sample handle can also be specified using its row number as returned by `selex.sampleSummary`.

## Value

`selex.sampleSummary` returns a data frame containing sequencing run names, sample names, rounds, left and right barcodes, and file paths.

## See Also

`selex.defineSample`, `selex.getAttributes`, `selex.loadAnnotation`, `selex.sample`, `selex.saveAnnotation`

## Examples

```
selex.sampleSummary()
```

`selex.saveAnnotation` *Save sample annotations to file*

## Description

A function used to save sample metadata visible to the current SELEX session in a sample annotation file.

## Usage

```
selex.saveAnnotation(filePath)
```

## Arguments

| | |
|---|---|
| `filePath` | Location on disk to create the sample annotation file. **The full system path must be specified.** |

## Details

A sample annotation file is an XML file that acts as a database storing metadata for different SELEX experiments. `selex.saveAnnotation` provides a convenient way to permanently store manually entered sample information using `selex.defineSample`. For more information on the XML format used to store the information, see `selex.loadAnnotation`.

## Value

Not applicable

## See Also

`selex.defineSample`, `selex.loadAnnotation`, `selex.sampleSummary`

## Examples

```
selex.defineSample(seqName='R0.libraries', seqFile=sampleFiles[1],
                sampleName='R0.barcodeGC', round=0, varLength=16,
                leftBarcode='TGG', rightBarcode='CCAGCTG')

selex.saveAnnotation(paste0(workDir, "sample_annotations.xml"))
```

`selex.seqfilter` *Create a sequence filter*

## Description

A function used to create a sequence filter object to conveniently and precisely include or exclude sequences from being counted or displayed. The filters are formed using Java regular expressions and can be used by a variety of functions within the package.

## Usage

```
selex.seqfilter(variableRegionIncludeRegex=NULL,
  variableRegionExcludeRegex=NULL, variableRegionGroupRegex=NULL,
  kmerIncludeRegex=NULL, kmerExcludeRegex=NULL, kmerIncludeOnly=NULL,
  viewIncludeRegex=NULL, viewExcludeRegex=NULL, viewIncludeOnly=NULL)
```

## Arguments

`variableRegionIncludeRegex`
: Include reads with variable regions containing this regular expression.

`variableRegionExcludeRegex`
: Exclude reads with variable regions containing this regular expression.

`variableRegionGroupRegex`
: Select subsequences of variable regions matching this regular expression.

`kmerIncludeRegex`
: Perform K-mer counting on variable regions containing this regular expression.

`kmerExcludeRegex`
: Perform K-mer counting on variable regions **not** containing this regular expression.

`kmerIncludeOnly`
: Perform K-mer counting on variable regions **exactly** matching this regular expression.

`viewIncludeRegex`
: Display K-mers containing this reguar expression.

`viewExcludeRegex`
: Display K-mers **not** containing this regular expression.

`viewIncludeOnly`
: Display K-mers **exactly** matching this regular expression.

## Details

The filters described by `selex.seqfilter` are used to filter sequences in the different stages of the K-mer counting process: read filtering, variable region filtering, and K-mer filtering.

Read Filtering Variable Region Filtering K-mer Filtering

| | | |
|---|---|---|
| variableRegionIncludeRegex | kmerIncludeRegex | viewIncludeRegex |
| variableRegionExcludeRegex | kmerExcludeRegex | viewExcludeRegex |
| variableRegionGroupRegex | kmerIncludeOnly | viewIncludeOnly |

Read filtering includes or excludes reads from the FASTQ file, acting as additional filters to those used to extract the variable regions. For example, consider an experimental design where the left barcode is TGG, right right barcode is TTAGC, and the variable region length is 10. FASTQ reads will be rejected unless they have the correct format; the sequences below represent hypothetical FASTQ reads:

| | |
|---|---|
| 5' TGG NNNNNNNNN TTAGC 3' | template |
| 5' TCG ATCAGTGGAC TTAGC 3' | **fails** (left match failed) |
| 5' TGG NAGGTCAGAC TTAGC 3' | **fails** (indeterminate base in variable region) |
| 5' TGG ATCAGTGGAC TTAGC 3' | **passes** |

The read filter options then act as additional filters on the 10-bp variable region. `variableRegionGroupRegex` has the added functionality of selecting substrings from the variable region itself. Using the same example, `variableRegionGroupRegex` could be used to select 5-mers regions flanked by AA on the left and the right (or AA NNNNN AA):

| | |
|---|---|
| 5' TCG ATCAGTGGAC TTAGC 3' | **fails** template (left match failed) |
| 5' TGG ATCAGTGGAC TTAGC 3' | **passes** template, **fails** filter (no match) |
| 5' TGG TAAGTGCCAA TTAGC 3' | **passes** template and filter |

When the `variableRegionGroupRegex` filter matches, *only the subsequence* will be used in future counting. In the above example, this would be GTGCC.

After the variable regions have been extracted from the FASTQ file, the next step involves K-mer counting. Variable region filtering comes into play here, allowing or preventing K-mer counting on these sequences. Lastly, K-mer filtering determines what K-mers are returned or displayed in tables.

Any function utilizing sequence filters will recompute results if, for a given sample, new values for the read filtering or variable region filter options are provided.

## Value

`selex.seqfilter` returns a sequence filter object.

## See Also

`selex.affinities`, `selex.counts`, `selex.getSeqfilter`, `selex.infogain`, `selex.kmax`, `selex.mm`

## Examples

```
# Raw K-mer counts
my.counts1 = selex.counts(sample=r0, k=16, top=100)

# Include reads whose variable regions begin with TGTA
```

```
regex = selex.seqfilter(variableRegionIncludeRegex="^TGTA")
my.counts2 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Exclude reads whose variables regions begin with TGT
regex = selex.seqfilter(variableRegionExcludeRegex="^TGT")
my.counts3 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Extract 13-bp substring from reads whose variable regions begin with TGT
regex = selex.seqfilter(variableRegionGroupRegex="^TGT([ACGT]{13})")
my.counts4 = selex.counts(sample=r0, k=13, top=100, seqfilter=regex)

# Extract 5-bp substring from reads whose variable regions begin with TGT
regex = selex.seqfilter(variableRegionGroupRegex="^TGT([ACGT]{5})")
my.counts5 = selex.counts(sample=r0, k=5, top=100, seqfilter=regex)

# Select variable regions beginning with A and ending with G
regex = selex.seqfilter(kmerIncludeRegex="^A.{14}G")
my.counts6 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Exclude variable regions beginning with A and ending with G
regex = selex.seqfilter(kmerExcludeRegex="^A.{14}G")
my.counts7 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Exclude variable regions beginning with A and ending with G, and display
# 16-mers that start and end with T
regex = selex.seqfilter(kmerExcludeRegex="^A.{14}G",
  viewIncludeRegex="^T[ACTG]{14}T")
my.counts8 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Exclude variable regions beginning with A and ending with G, and display
# 16-mers that do not start and end with T
regex = selex.seqfilter(kmerExcludeRegex="^A.{14}G",
  viewExcludeRegex="^T[ACTG]{14}T")
my.counts9 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Only count variable regions containing TGTAAAATCAGTGCTG or TGTAAGTGGACTCTCG
regex = selex.seqfilter(kmerIncludeOnly=c('TGTAAAATCAGTGCTG',
  'TGTAAGTGGACTCTCG'))
my.counts10 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)

# Only display results for the K-mers TGTAAAATCAGTGCTG and TGTAAGTGGACTCTCG
regex = selex.seqfilter(viewIncludeOnly=c('TGTAAAATCAGTGCTG',
  'TGTAAGTGGACTCTCG'))
my.counts11 = selex.counts(sample=r0, k=16, top=100, seqfilter=regex)
```

---

`selex.setwd` *Set or change the working directory*

---

## Description

A function that changes the current working directory.

### Usage

```
selex.setwd(path)
```

### Arguments

| | |
|---|---|
| path | Physical location on disk for the current working directory. **The full system path must be specified.** |

### Details

The working directory must be specified for every selex.run. This directory is used to store the output of all analyses performed by the selex package. If a certain calculation has already been performed in the given working directory, the previously computed values are returned. If the specified path does not exist, `selex.config` will attempt to create it.

### Value

Not applicable

### See Also

selex.config

### Examples

```
workDir = file.path(".", "SELEX_workspace")
# Change the current working directory
selex.setwd(path=workDir)
```

---

## selex.split *Randomly split a dataset*

---

### Description

A function used to randomly split and store a dataset into smaller fractions as defined by `ratios`. This is useful in generating cross-validation datasets for Markov Model testing when none are available.

### Usage

```
selex.split(sample, ratios=NA)
```

### Arguments

| | |
|---|---|
| sample | A sample handle to the dataset to be split. |
| ratios | The fractions with with new datasets should be generated. If `NA`, ratios is set to `c(.5,.5)` and `sample` is evenly split into two datasets. |

## Details

selex.split saves the newly generated datasets in the current working directory and makes them visible to the current SELEX session. Naming convention of new samples: New sequence name = <seqName>.split.x ; New sample name = <sampleName>.split.x .

## Value

Not applicable

## See Also

selex.defineSample, selex.loadAnnotation, selex.mm, selex.sampleSummary

## Examples

```
# Split the r0 sample into testing and training datasets
r0.split = selex.split(sample=r0)

# show information about subsamples
names(r0.split)
r0.split$info
selex.getAttributes(r0.split$train)

# Display all currently loaded samples
selex.sampleSummary()
```

---

# selex.summary *Display all count table, Markov model, and information gain summaries*

---

## Description

This function returns the summaries of either all the calculated count tables, Markov models, and information gain values in the current working directory or those for a specific sample.

## Usage

```
selex.summary(sample=NULL, displayFilter=FALSE)
```

## Arguments

| | |
|---|---|
| sample | A sample handle to the sample for which summaries should be returned. |
| displayFilter | Display all applied sequence filter attributes. |

## Value

`selex.summary` returns a list containing the following components:

`countSummary` a data frame containing the sample, kmer length, offset, minimum/maximum count, total count, and applied filters for either all count tables or a specified sample in the current working directory.

`markovModel` a data frame containing the sample, order, Markov model type, R^2 value, cross validation sample/length, and applied filters for either all computed Markov models or a specified sample in the current working directory.

`informationGain` a data frame containing the sample, Kmer length, information gain value, Markov model/type, and applied filters for either all computed information gain values or a specified sample in the current working directory.

## See Also

`selex.counts`, `selex.countSummary`, `selex.infogain`, `selex.infogainSummary`, `selex.mm`, `selex.mmSummary`

## Examples

```
selex.summary()
```

# Index