Package 'NestLink'

August 1, 2024

Type Package

Title NestLink an R data package to guide through Engineered Peptide Barcodes for In-Depth Analyzes of Binding Protein Ensembles

Version 1.20.0

Depends R (>= 3.6), AnnotationHub (>= 2.15), ExperimentHub (>= 1.0), Biostrings (>= 2.51), gplots (>= 3.0), protViz (>= 0.4), ShortRead (>= 1.41)

Imports grDevices, graphics, stats, utils

Description Provides next-generation sequencing (NGS) and mass spectrometry (MS) sample data, code snippets and replication material used for developing NestLink. The NestLink approach is a protein binder selection and identification technology able to biophysically characterize thousands of library members at once without handling individual clones at any stage of the process. Data were acquired on NGS and MS platforms at the Functional Genomics Center Zurich.

License GPL

VignetteBuilder knitr

Suggests BiocStyle (>= 2.2), DT, ggplot2, knitr, rmarkdown, testthat, specL, lattice, scales

NeedsCompilation no

biocViews ExperimentHub, ExperimentData, SequencingData, MassSpectrometryData, ReproducibleResearch

RoxygenNote 6.1.1

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

git_branch RELEASE_3_19

git_last_commit 2f8800a

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-08-01

Author Pascal Egloff [aut] (<https://orcid.org/0000-0001-8948-3704>),
Iwan Zimmermann [ctb] (<https://orcid.org/0000-0003-3476-4749>),
Fabian M. Arnold [ctb],
Cedric A.J. Hutter [ctb] (<https://orcid.org/0000-0002-8920-3343>),
Lennart Opitz [aut, cre] (<https://orcid.org/0000-0001-7945-6737>),
Lucy Poveda [ctb] (<https://orcid.org/0000-0002-5291-5582>),
Hans-Anton Keserue [ctb],
Christian Panse [aut, ctb] (<https://orcid.org/0000-0003-1975-3064>),
Bernd Roschitzki [aut] (<https://orcid.org/0000-0001-5756-9773>),
Markus Seeger [aut] (<https://orcid.org/0000-0003-1761-8571>)

Maintainer Lennart Opitz <lopitz@fgcz.ethz.ch>

Contents

`.ssrc.mascot` *computes the correlation of predicted and measured retention time*

Description

this helper function computes a linear model between redicted and measured retention time of the as imput set given identified peptides.

TODO(cp): consider moving this method to the protViz package.

Usage

```
.ssrc.mascot(x, scores = c(10, 20, 40, 50), ...)
```

Arguments

x	as,data.frame.mascot generated data.frame object.
scores	default is c(10, 20, 40, 50).
...	passed to the plot function.

Value

a plot and summary

Author(s)

Christian Panse <cp@fgcz.ethz.ch>, 2017,2019

Examples

```
library(ExperimentHub)
eh <- ExperimentHub();
load(query(eh, c("NestLink", "F255744.RData"))[[1]])
.ssrc.mascot(F255744, scores = 15)
```

---

compose_GPGx8cTerm *Compose a peptide with a defined AA sequence frequency*

Description

composes, out of an as input given amino acid distribution, a randomly sampled amino acid sequence. compose_GPGx8cTerm, compose_GSx7cTerm, and compose_GPx10R belong to three groups composing different flycode (peptide) construction. The construction is given in the function name. For example, GPGx8cTerm, composes a flycode having as prefix GPG followed by eight (x8) amino acids followed by a cTerm sequence. The different construction will have different detectability properties as mass range and hydrophobicity values.

Usage

```
compose_GPGx8cTerm(pool = c(rep("A", 12), rep("S", 0), rep("T", 12),
  rep("N", 12), rep("Q", 12), rep("D", 8), rep("E", 0), rep("V", 12),
  rep("L", 0), rep("F", 0), rep("Y", 8), rep("W", 0), rep("G", 12),
  rep("P", 12)), cTerm = c("VFR", "VSR", "VFGIR", "VSGER"))
```

Arguments

pool	AA distributen.
cTerm	c-Terms

Value

a AA sequence

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2015

Examples

```
set.seed(1)
compose_GPGx8cTerm()
(FlyCodes <- replicate(10, compose_GPGx8cTerm()))
plot(parentIonMass(FlyCodes) ~ssrc(FlyCodes))
```

---

compose_GPx10R *Compose a peptide with a defined AA sequence*

Description

composes, out of an as input given amino acid distribution, a randomly sampled amino acid sequence. compose_GPGx8cTerm, compose_GSx7cTerm, and compose_GPx10R belong to three groups composing different flycode (peptide) construction. The construction is given in the function name. For example, GPGx8cTerm, composes a flycode having as prefix GPG followed by eight (x8) amino acids followed by a cTerm sequence. The different construction will have different detectability properties as mass range and hydrophobicity values.

Usage

```
compose_GPx10R(aa_pool1, aa_pool2)
```

Arguments

aa_pool1	AA distributen.
aa_pool2	AA distributen.

Value

a AA sequence

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2015

Examples

```
set.seed(1)
aa_pool_1_2_9_10 <- c(rep('A', 8), rep('S', 7), rep('T', 7), rep('N', 6),
rep('Q', 6), rep('D', 8), rep('E', 8), rep('V', 9), rep('L', 6), rep('F', 5),
rep('Y', 9), rep('W', 6), rep('G', 15), rep('P', 0))

aa_pool_3_8 <- c(rep('A', 5), rep('S', 4), rep('T', 5), rep('N', 2),
rep('Q', 2), rep('D', 8), rep('E', 8), rep('V', 7), rep('L', 5), rep('F', 4),
```

```
rep('Y', 6), rep('W', 4), rep('G', 12), rep('P', 28))

compose_GPx10R(aa_pool_1_2_9_10, aa_pool_3_8)
(FlyCodes <- replicate(10, compose_GPx10R(aa_pool_1_2_9_10, aa_pool_3_8)))
plot(parentIonMass(FlyCodes) ~ssrc(FlyCodes))
```

`compose_GSx7cTerm` *Compose a FlyCode GSx7cTerm Amino Acid Sequence*

Description

composes, out of an as input given amino acid distribution, a randomly sampled amino acid sequence. `compose_GPGx8cTerm`, `compose_GSx7cTerm`, and `compose_GPx10R` belong to three groups composing different flycode (peptide) construction. The construction is given in the function name. For example, GPGx8cTerm, composes a flycode having as prefix GPG followed by eight (x8) amino acids followed by a cTerm sequence. The different construction will have different detectability properties as mass range and hydrophobicity values.

Usage

```
compose_GSx7cTerm(pool = c(rep("A", 18), rep("S", 6), rep("T", 12),
  rep("N", 1), rep("Q", 1), rep("D", 11), rep("E", 11), rep("V", 12),
  rep("L", 2), rep("F", 1), rep("Y", 4), rep("W", 1), rep("G", 8), rep("P",
  12)), cTerm = c("WR", "WLTVR", "WQEGGR", "WQSR", "WLR"))
```

Arguments

`pool`	a vector of amino acids.
`cTerm`	a vector of a sequence suffix.

Value

a amino acid sequence, e.g., GSAPTTVFGWLTVR.

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2015

Examples

```
sample.size <- 100
#
## Compose a GSXXXXXXX(WR|WLTVR|WQGGER|WQSR|WLR) peptide
set.seed(2)
FC.GSx7cTerm <- replicate(sample.size, compose_GSx7cTerm())
## Some Sanity Checks
table(FC.GSx7cTerm)
stopifnot(length(FC.GSx7cTerm) == 100)
FC.PATTERN <- "^GS[ASTNQDEFVLYWGP]{7}(WR|WLTVR|WQEGGR|WLR|WQSR)$"
```

```
stopifnot(
  length(FC.GSx7cTerm[grepl(FC.PATTERN, FC.GSx7cTerm)])
    == sample.size)
```

F255744 *F255744 Mascot Search results*

Description

F255744 Mascot Search results

Author(s)

Pascal Egloff <p.egloff@imm.uzh.ch>

See Also

F255744

Examples

```
library(ExperimentHub)
eh <- ExperimentHub(); load(query(eh, c("NestLink", "F255744.RData"))[[1]])
class(F255744)
hist(F255744$RTINSECONDS)
hist(F255744$RTINSECONDS[F255744$pep_score > 20])
```

---

getExperimentHubFilename *getExperimentHubFilename*

Description

getExperimentHubFilename

Usage

```
getExperimentHubFilename(filename)
```

Arguments

filename	of the aws s3 blob.

Value

the file name of the local ExperimentHub.

Examples

```
fl <- system.file("extdata", "metadata.csv", package="NestLink")
metadata <- read.csv(fl, stringsAsFactors=FALSE)
     metadata$Title

lapply(metadata$RDataPath, getExperimentHubFilename)
```

---

getFC *Read FlyCodes (FCs)*

Description

A wrapper function for reading the flycodes using ExperimentHub. The files are used for demonstrating the detectability of the AA sequences. The wrapper functions are extended by columns ssrc prediction and the parentIonMass. The column ESP_Prediction was generated by using the service from https://genepattern.broadinstitute.org.

Usage

```
getFC(pattern = "^GS[ASTNQDEFVLYWGP]{7}(WR|WLTVR|WQEGGR|WLR|WQSR)$",
  filename = NULL)
```

Arguments

pattern	a regular expression FlyCode pattern
filename	a two column tab separated file containing a peptide sequence and an ESP value. default is NULL which reads the data provided by the package through ExperimentHub.

Value

a data.frame object of Flycodes

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2015, 2018

Source

- https://fgcz-gstore.uzh.ch/projects/p1644/analysis_20170609_o3040/p1644o3482-4_S4.extendedFrags_uniqNB2FC.txt
- https://fgcz-gstore.uzh.ch/projects/p1644/analysis_20170609_o3040/p1644o3482-5_S5.extendedFrags_uniqNB2FC.txt

Examples

```
FC <- getFC()
dim(FC)
```

getNB *Read NanoBodies (NBs)*

Description

A wrapper function for reading the flycodes using ExperimentHub. The files are used for demonstrating the detectability of the AA sequences. The wrapper functions are extended by columns ssrc prediction and the parentIonMass. The column ESP_Prediction was generated by using the service from https://genepattern.broadinstitute.org.

Usage

```
getNB(filename = NULL)
```

Arguments

filename	a two column tab separated file containing a peptide sequence and an ESP value. default is NULL which reads the data provided by the package through ExperimentHub.

Value

a data.frame object of NBs

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2015, 2018, 2019

Source

- https://fgcz-gstore.uzh.ch/projects/p1644/analysis_20170609_o3040/p1644o3482-4_S4.extendedFrags_uniqNB2FC.txt
- https://fgcz-gstore.uzh.ch/projects/p1644/analysis_20170609_o3040/p1644o3482-5_S5.extendedFrags_uniqNB2FC.txt

Examples

```
NB <- getNB()
dim(NB)
```

nanobodyFlycodeLinking.as.fasta *Write FASTA*

Description

Write FASTA

Usage

```
nanobodyFlycodeLinking.as.fasta(x, file = NULL, ...)
```

Arguments

x	a nanobodyFlycodeLinking S3 object computed by runNGSAnalysis.
file	a filename
...	just passed

Value

sprintf stream

Author(s)

Lennart Opitz, Christian Panse 2018

Examples

```
library(ExperimentHub)
eh <- ExperimentHub()
f <- query(eh, c("NestLink", "nanobodyFlycodeLinkage.RData"))[[1]]
load(f)
summary(nanobodyFlycodeLinkage.sample)
nanobodyFlycodeLinking.as.fasta(nanobodyFlycodeLinkage.sample)
```

nanobodyFlycodeLinking.summary *Object Summaries of S3 class* nanobodyFlycodeLinking

Description

Object Summaries of S3 class nanobodyFlycodeLinking

Usage

```
nanobodyFlycodeLinking.summary(object)
```

Arguments

object	a nanobodyFlycodeLinking class computed by runNGSAnalysis.

Value

a data.frame object

Examples

```
library(ExperimentHub)
eh <- ExperimentHub()
f <- query(eh, c("NestLink", "nanobodyFlycodeLinkage.RData"))[[1]]
load(f)
summary(nanobodyFlycodeLinkage.sample)
```

NB.unambiguous *Determine unambiguous NBs*

Description

Determine unambiguous NBs

Usage

```
NB.unambiguous(x = getNB())
```

Arguments

x	a data.frame containing a column peptide

Value

a data.frame a data.frame of unambiguously assignable peptides (those, which occur only on one nanobody)

Examples

```
NB <- getNB()
dim(NB.unambiguous(NB))
```

NB.unique *make NB table unique*

Description

make NB table unique

Usage

```
NB.unique(x = getNB())
```

Arguments

x	a data.frame

Value

a data.frame

Examples

```
NB <- getNB()
dim(NB.unique(NB))
```

PGexport *PGexport results*

Description

PGexport results

Author(s)

Pascal Egloff <p.egloff@imm.uzh.ch>

Source

https://fgcz-bfabric.uzh.ch

- Workunit : 158716 - QEXACTIVEHF_1 20170919_16_62465_nl5idx1-3_6titratecoli.raw 20170919_05_62465_nl5idx1-3_6titratecoli.raw
- Workunit : 158717 - QEXACTIVEHF_1 20170919_14_62466_nl5idx1-3_7titratesmeg.raw 20170919_09_62466_nl5idx1-3_7titratesmeg.raw

Examples

```
# filename <- system.file(
#  "extdata/PGexport2_normalizedAgainstSBstandards_Peptides.csv",
#  package = "NestLink")
library(ExperimentHub)
eh <- ExperimentHub()
filename <- query(eh,
  c("NestLink", "PGexport2_normalizedAgainstSBstandards_Peptides.csv"))[[1]]
P <- read.csv(filename, header = TRUE, sep=';')
P <- P[P$Modifications == '', ]
P <- P[,c('Accession', 'Sequence',
"X20170919_05_62465_nl5idx1.3_6titratecoli",
"X20170919_16_62465_nl5idx1.3_6titratecoli",
"X20170919_09_62466_nl5idx1.3_7titratesmeg",
"X20170919_14_62466_nl5idx1.3_7titratesmeg")]
names(P)<-c('Accession','Sequence','coli1', 'coli2', 'smeg1', 'smeg2')
P<- P[grep("^P[0-9][A-Z][0-9]", P$Accession), ]

P$FCset_ng <- NA
P$FCset_ng[P$Accession %in% c('P1A4', 'P1B4', 'P1C4',
  'P1D4', 'P1E4', 'P1F4')] <- 92
P$FCset_ng[P$Accession %in% c('P1A5', 'P1B5', 'P1C5',
  'P1D5', 'P1G4', 'P1H4')] <- 295
P$FCset_ng[P$Accession %in% c('P1A6', 'P1B6', 'P1E5',
  'P1F5', 'P1G5', 'P1H5')] <- 943
P$FCset_ng[P$Accession %in% c('P1C6', 'P1D6', 'P1E6',
  'P1F6', 'P1G6', 'P1H6')] <- 3017

 P$coli1 <- (log(P$coli1,2) - mean(log(P$coli1,2))) / sd(log(P$coli1,2))
 P$coli2 <- (log(P$coli2,2) - mean(log(P$coli2,2))) / sd(log(P$coli2,2))
 P$smeg1 <- (log(P$smeg1,2) - mean(log(P$smeg1,2))) / sd(log(P$smeg1,2))
 P$smeg2 <- (log(P$smeg2,2) - mean(log(P$smeg2,2))) / sd(log(P$smeg2,2))

  O <- P
  b <- boxplot(df<-cbind(P$coli1 - P$coli2, P$coli1 - P$smeg1,
  P$coli1 - P$smeg2,P$coli2 - P$smeg1, P$coli2 - P$smeg2,
  P$smeg1 - P$smeg2),
   ylab='normalized log2ratios', ylim = c(-1,1), axes=FALSE,
    main=paste("ConcGr = all"))
axis(1, 1:6, c('coli[12]', 'coli1-smeg1', 'coli1-smeg2', 'coli2-smeg1',
'coli2- smeg2','smeg[12]'))
abline(h=0, col='red')
box()
axis(2)
axis(3, 1:6, b$n)
outliers.idx <- sapply(1:length(b$group), function(i){
  q <- df[, b$group[i]] == b$out[i];
  text(b$group[i], b$out[i], P[q, 2], pos=4, cex=0.4);
   text(b$group[i], b$out[i], P[q, 1], pos=2, cex=0.4);
   which(q)}
   )
```

plot_in_silico_LCMS_map *plot a LC-MS map of a given set of amino acid sequences*

Description

plot a LC-MS map of a given set of amino acid sequences

Usage

```
plot_in_silico_LCMS_map(peptides, ...)
```

Arguments

peptides	a vector of pepitdes.
...	pass through the plot method.

Details

TODO(cp): consider using hexbin using ggplot2 ggplot facet_wrap aes geom_point

Value

gplots::hist2d a gplot 2d histogram

Author(s)

Christian Panse

Examples

```
set.seed(1)
par(mfrow=c(2,1));
FlyCodes <- replicate(10000, compose_GPGx8cTerm())
rv <- plot_in_silico_LCMS_map(FlyCodes)
```

`runNGSAnalysis` *NGS linkage workflow*

Description

performs the NGS filtering workflow to get high quality FlyCode and Nanobody sequences linkage.

Usage

```
runNGSAnalysis(file, param)
```

Arguments

`file`	sequence file path
`param`	list of input parameters, explained in details paragraph below.

Details

The elements of the parameter list object is described as follows:

- `NB_Linker1` nucleotide sequence of the linker left to the nanobody.
- `NB_Linker2` nucleotide sequence of the linker right to the nanobody.
- `ProteaseSite` nucleotide sequence left to the flycode.
- `FC_Linker` nucleotide sequence right to the flycode.
- `knownNB` known nanobody sequences in the experiment.
- `nReads` number of Reads from the start of fastq file to process.
- `minRelBestHitFreq` minimal fraction of the dominant nanobody for a specific flycode.
- `minConsensusScore` minimal fraction per sequence position in nanabody consensus sequence calculation.
- `maxMismatch` number of accepted mismatches for all pattern search steps.
- `minNanobodyLength` minimal nanobody length in [nt].
- `minFlycodeLength` minimal flycode length in [nt].
- `FCminFreq` minimal number of subreads for a specific flycode to keep it in the analysis.

missing elements are replace by the example provided values.

Value

uniqNB2FC dataframe

Author(s)

Lennart Opitz <lopitz@fgcz.ethz.ch>, 2019

Examples

```
library(ExperimentHub)
eh <- ExperimentHub()
expFile <- query(eh, c("NestLink", "NL42_100K.fastq.gz"))[[1]]
knownNB_File <- query(eh, c("NestLink", "knownNB.txt"))[[1]]
knownNB_data <- read.table(knownNB_File, sep='\t', header = TRUE,
    row.names = 1, stringsAsFactors = FALSE)
knownNB <- Biostrings::translate(DNAStringSet(knownNB_data$Sequence))
names(knownNB) <- rownames(knownNB_data)
knownNB <- sapply(knownNB, toString)
param <- list()
param[['NB_Linker1']] <- "GGCCggcggGGCC"
param[['NB_Linker2']] <- "GCAGGAGGA"
param[['ProteaseSite']] <- "TTAGTCCCAAGA"
param[['FC_Linker']] <- "GGCCaaggaggcCGG"
param[['knownNB']] <- knownNB
param[['nReads']] <- 10000
param[['minRelBestHitFreq']] <- 0.8
param[['minConsensusScore']] <- 0.9
param[['maxMismatch']] <- 1
param[['minNanobodyLength']] <- 348
param[['minFlycodeLength']] <- 33
param[['FCminFreq']] <- 1
runNGSAnalysis(file = expFile[1], param)
```

---

twoPatternReadFilter *Filter input sequences for two patterns*

Description

Filter input sequences for two patterns

Usage

```
twoPatternReadFilter(reads, leftPattern, rightPattern, maxMismatch,
  prevPatternPos = NULL)
```

Arguments

reads	input sequences
leftPattern	left pattern motive.
rightPattern	right pattern motive.
maxMismatch	maximal number of miss matches.
prevPatternPos	prev pattern posiotion; default is set to NULL.

Value

list object

Examples

```
reads <- DNAStringSet(c('ACTGGGTTT','ACCCTGGGTTT'))
leftPattern <- 'CT'
rightPattern <- 'TTT'
maxMismatch <- 0
twoPatternReadFilter(reads, leftPattern, rightPattern, maxMismatch)
```

WU160118 *WU160118 Mascot Search results*

Description

WU160118 Mascot Search results

Author(s)

Christian Panse

References

https://fgcz-bfabric.uzh.ch/bfabric/userlab/show-workunit.html?id=160118

See Also

please read the vignette summaryFASTA.Rmd.

Examples

```
library(ExperimentHub)
eh <- ExperimentHub();
load(query(eh, c("NestLink", "WU160118.RData"))[[1]])
class(WU160118)
PATTERN <- "^GS[ASTNQDEFVLYWGP]{7}(WR|WLTVR|WQEGGR|WLR|WQSR)$"
idx <- grepl(PATTERN, WU160118$pep_seq)
WU <- WU160118[idx & WU160118$pep_score > 25,]

library(lattice)
histogram(~RTINSECONDS| datfilename, data = WU, type='count')
```

Index