Package: ksrlive (via r-universe)

September 1, 2024

Type Package

Title Identify Kinase Substrate Relationships Using Dynamic Data

Version 1.0

Date 2015-10-13

Author Westa Domanova

Maintainer Westa Domanova <w.domanova@gmail.com>

Description Using this package you can combine known kinase substrate relationships with experimental data and determine active kinases and their substrates.

License GPL-2 | GPL-3

Depends R (>= 3.0.0)

Imports tightClust, stats

RoxygenNote 4.1.1.9001

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-19 08:41:56

Contents

clust.expand *Find clusters containing core substrates*

Description

clust.expand returns a list of kinase substrate relationships

Usage

```
clust.expand(clust, clust_all, diff = NULL)
```

Arguments

clust	named list containing named vectors of cluster assignments, names correspond to rownames in data and names of list are kinase identifiers (result of clustering performed using exclusive substrates)
clust_all	named list containing named vectors of cluster assignments, names correspond to rownames in data and names of list are kinase identifiers (result of clustering performed using all substrates)
diff	character vector of substrate identifiers that are differentially regulated

Details

The function clust.expand takes the resulting core substrates from the exclusive clustering and finds the corresponding substrate clusters in the clustering using all substrates.

Value

named list containing named vectors of cluster assignments, names correspond to rownames in data and names of list are kinase identifiers

Examples

```
data(phosphonetworkdf)
data(datakin)
# only need what is present in data
phosphonetwork_data <- phosphonetwork_df[
phosphonetwork_df[,"SUB_IDENT"] %in% data_kin[,"SUB_IDENT"]
,]
fam <- list(akt = c("P31749", "P31751"))
kin_data_fam_exc <- KSR.list(phosphonetwork_data[, c("SUB_IDENT", "KIN_ACC_ID")],
                             kinasefamilies = fam,
                             exclusive = TRUE)

# only do for Akt and Mtor (P31749, P42345)
substrate_profiles <- lapply(kin_data_fam_exc[c("P31749", "P42345")],
function(x){data_kin[match(x, data_kin[,"SUB_IDENT"]),1:9]})
```

```
substrate_profiles_random <- lapply(substrate_profiles,
function(x){rbind(x, random.data(x, random.seed = 123))})

target <- 3
substrate_profiles_tight <- lapply(substrate_profiles_random, function(x){
tightClust::tight.clust(x, target = target, k.min = 7, resamp.num = 100, random.seed = 12345)
})

kin_clust<- mapply(function(x,y){clustering(x, y)},
                        substrate_profiles_tight, substrate_profiles, SIMPLIFY = FALSE)

# do clustering using all available substrates
kin_data_fam <- KSR.list(phosphonetwork_data[, c("SUB_IDENT", "KIN_ACC_ID")],
                         kinasefamilies = fam)

substrate_profiles_all <- lapply(kin_data_fam[c("P31749", "P42345")],
function(x){data_kin[match(x, data_kin[,"SUB_IDENT"]),1:9]})

substrate_profiles_random_all <- lapply(substrate_profiles_all,
                       function(x){rbind(x, random.data(x, random.seed = 123))})

target <- 3
substrate_profiles_tight_all <- lapply(substrate_profiles_random_all, function(x){
tightClust::tight.clust(x, target = target, k.min = 7, resamp.num = 100, random.seed = 12345)
})

kin_clust_all <- mapply(function(x,y){clustering(x, y)},
                        substrate_profiles_tight_all, substrate_profiles_all,
                        SIMPLIFY = FALSE)

expand_all <- mapply(function(x,y){clust.expand(x, y)},
                        kin_clust, kin_clust_all, SIMPLIFY = FALSE)
```

`clustering` *Return clustering assignments produced by tight.clust*

Description

`clustering` returns vectors of clustering assignments

Usage

```
clustering(tightclust, data)
```

Arguments

`tightclust`	list of objects returned by the tight.clust function
`data`	data frame of time course of substrates, each substrate is a row

Details

The function clustering creates a named list of cluster assignments for substrates.

Value

named list containing named vectors of cluster assignments, names correspond to rownames in data and names of list are kinase identifiers

Examples

```
data(phosphonetworkdf)
data(datakin)
# only need what is present in data
phosphonetwork_data <- phosphonetwork_df[
phosphonetwork_df[,"SUB_IDENT"] %in% data_kin[,"SUB_IDENT"]
,]
fam <- list(akt = c("P31749", "P31751"))
kin_data_fam_exc <- KSR.list(phosphonetwork_data[, c("SUB_IDENT", "KIN_ACC_ID")],
                             kinasefamilies = fam,
                             exclusive = TRUE)
# only do for Akt and Mtor (P31749, P42345)
substrate_profiles <- lapply(kin_data_fam_exc[c("P31749", "P42345")],
function(x){data_kin[match(x, data_kin[,"SUB_IDENT"]),1:9]})

substrate_profiles_random <- lapply(substrate_profiles,
function(x){rbind(x, random.data(x, random.seed = 123))})

target <- 3
substrate_profiles_tight <- lapply(substrate_profiles_random, function(x){
tightClust::tight.clust(x, target = target, k.min = 7, resamp.num = 100, random.seed = 12345)
})

kin_clust<- mapply(function(x,y){clustering(x, y)},
                        substrate_profiles_tight, substrate_profiles, SIMPLIFY = FALSE)
```

---

data_kin *Time course data for phosphorylation sites*

Description

This dataset contains time course data of phosphorylation sites after insulin stimulation.

Usage

```
data_kin
```

Format

```
'data.frame': 84 obs. of  10 variables:
 $ 0_scaled    : num  0.4481 0 0 0.1618 0.0909 ...
 $ 15s_scaled  : num  0.224 0.517 0.357 0 0 ...
 $ 30s_scaled  : num  0.266 0.655 0.636 0.785 0.136 ...
 $ 1min_scaled : num  0.0332 1 0.8149 0.7188 0.0909 ...
 $ 2min_scaled : num  0 0.918 0.756 0.912 0 ...
 $ 5min_scaled : num  0.6017 0.6897 0.8571 0.9523 0.0455 ...
 $ 10min_scaled: num  0.759 0.74 0.964 0.79 1 ...
 $ 20min_scaled: num  1 0.483 0.974 1 0.5 ...
 $ 60min_scaled: num  0.598 0.724 1 0.78 0.545 ...
 $ SUB_IDENT   : chr  "O43521_FIFMRRSSLLSRSSS" "O60343_QFRRRAHTFSHPPSS" "O60825_IRRPRNYSVGSRPLK" "O6082
```

Source

Humphrey et al., Cell Metabolism, 2013

KSR.list *Create a kinase substrate relationship list from a data frame*

Description

`KSR.list` returns a list of kinase substrate relationships

Usage

```
KSR.list(df, kinasefamilies = NULL, exclusive = FALSE)
```

Arguments

`df`	data frame of kinase substrate relationships with substrate identifier in the first column and kinase identifier in the second column.
`kinasefamilies`	named list of kinase identifiers that have to be combined, one list per kinase family, list will be named after first family member
`exclusive`	logical, if TRUE only substrates exclusive to the kinase will be included in the list (substrates with multiple kinases will be excluded)

Details

The function KSR.list creates a list of kinase substrate relationships from a data frame and can combine kinase families into one list. Substrates occuring in multiple lists can be excluded.

Value

named list of substrate identifiers, with the corresponding kinase identifiers as the list names

Examples

```
data(phosphonetworkdf)
data(datakin)

# first column has to be substrate id, second kinase id
kin_data <- KSR.list(phosphonetwork_df[, c("SUB_IDENT", "KIN_ACC_ID")])
# Akt1 and Akt2 belong to the same kinase family, combine their substrates
# into one list and name the list after the first family member
fam <- list(akt = c("P31749", "P31751"))
kin_data_fam <- KSR.list(phosphonetwork_df[, c("SUB_IDENT", "KIN_ACC_ID")],
kinasefamilies = fam)

# only include phosphosites appearing once
kin_data_fam_exc <- KSR.list(phosphonetwork_df[, c("SUB_IDENT", "KIN_ACC_ID")],
                             kinasefamilies = fam,
                             exclusive = TRUE)
```

---

ksrlive *Identify site specific kinase substrate relationships using dynamic data.*

Description

Using this package you can combine known site specific kinase substrate relationships with dynamic experimental data and determine active kinases and their substrates.

Author(s)

Westa Domanova

---

phosphonetwork_df *site specific kinase substrate relationship dataset*

Description

This dataset contains all site specific kinase relationships from PhosphoSitePlus, PhosphoElm, HPRD and PhosphoPoint.

Usage

```
phosphonetwork_df
```

Format

```
'data.frame': 13505 obs. of  34 variables:
 $ SUB_ACC_ID         : chr  "A1KXE4" "A1X283" "A2A9C3" "A2APB8" ...
 $ MODSITE_SEQ      : chr  "QTGYTPGTPYKVSCS" "DMSASAGYEEISDPD" "TPGSLVGSPREASGM" "KIARDPQTPILQTKY" ...
 $ KIN_ACC_ID         : chr  "P24941" "P12931" "Q9JLN9" "P63085" ...
 $ ORG           : Factor w/ 17 levels "chicken","cow",..: 8 8 10 10 10 8 8 10 8 8 ...
 $ KINASE             : chr  "CDK2" "SRC" "MTOR" "ERK2" ...
 $ KIN_GENE_SYMB      : chr  "CDK2" "SRC" "MTOR" "MAPK1" ...
 $ HU_CHR_LOC       : Factor w/ 274 levels "","10p11.1","10p11.23",..: 27 132 1 1 1 7 215 1 173 13 ...
 $ SUBSTRATE          : chr  "FAM168B" "SH3PXD2B" "SZT2" "TPX2" ...
 $ SUB_GENE_ID        : chr  "130074" "285590" "230676" "72119" ...
 $ SUB_GENE_SYMB      : chr  "FAM168B" "SH3PXD2B" "Szt2" "Tpx2" ...
 $ SUB_MOD_RSD        : chr  "T57" "Y508" "S3230" "T369" ...
 $ SITE_GRP_ID      : int  9831677 17303901 14575118 455432 3202029 3963101 975498 468668 451197 454238 .
 $ IN_VIVO_RXN        : Factor w/ 2 levels " ","X": 1 2 2 1 1 1 2 1 2 2 ...
 $ IN_VITRO_RXN       : Factor w/ 2 levels " ","X": 2 1 1 2 2 2 1 2 2 2 ...
 $ CST_CAT.         : Factor w/ 563 levels "","11817","11834",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ PhosphositePLUS    : num  1 1 1 1 1 1 1 1 1 1 ...
 $ SEQ           : chr  "MNPVYSPGSSGVPYANAKGIGYPAGFPMGYAAAAPAYSPNMYPGANPTFQTGYTPGTPYKVSCSPTSGAVPPYSSS
 $ PhosphoELM         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ SwissProt          : chr  NA NA NA NA ...
 $ PubMed          : Factor w/ 2842 levels "",";","10023679;",..: NA NA NA NA NA NA NA NA NA NA ...
 $ KIN_GENE_ID        : chr  NA NA NA NA ...
 $ HPRD               : num  NA NA NA NA NA NA NA NA NA NA ...
 $ PhosphoPoint       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ SUB_HPRD_ID        : int  NA NA NA NA NA NA NA NA NA NA ...
 $ SUB_HPRDISO_ID    : Factor w/ 13183 levels "00001_1","00002_1",..: NA NA NA NA NA NA NA NA NA NA ...
 $ KIN_HPRD_ID      : Factor w/ 517 levels "-","00021","00084",..: NA NA NA NA NA NA NA NA NA NA ...
 $ SUB_ACC_ID.human   : chr  "A1KXE4" "A1X283" "Q5T011" "Q9ULW0" ...
 $ Position           : chr  "57" "508" "3230" "369" ...
 $ MODSITE_SEQ.human  : chr  "QTGYTPGTPYKVSCS" "DMSASAGYEEISDPD" "APGSSAGSPGEASGL" "KICRDPQTPVLQTKH" ..
 $ MODSITE_SEQ.mouse  : chr  "QTGYTPGTPYKVSCS" "DLSASTGYEEISDPT" "TPGSLVGSPREASGM" "KIARDPQTPILQTKY" ..
 $ SUB_ACC_ID.mouse   : chr  "Q80XQ8" "A2AAY5" "A2A9C3" "A2APB8" ...
 $ KIN_ACC_ID.human   : chr  "P24941" "P12931" "P42345" "P28482" ...
 $ KIN_GENE_SYMB.human: chr  "CDK2" "SRC" "MTOR" "MAPK1" ...
 $ SUB_IDENT        : chr  "A1KXE4_QTGYTPGTPYKVSCS" "A1X283_DMSASAGYEEISDPD" "Q5T011_APGSSAGSPGEASGL" "
```

`random.data` *Create random data*

Description

`random.data` returns a data frame of random numeric values

Usage

```
random.data(data, back_data = NULL, n = 50, random.seed = NULL)
```

Arguments

data	data frame of time course of substrates, each substrate is a row
back_data	data frame of numeric values that can to be used as background data, if not provided a values are drawn from a uniform distribution between minimum and maximum of input data
n	numeric specifying how many rows should be contained in the resulting data frame
random.seed	numeric used as seed

Details

The function random.data returns a data frame of random numeric values with the same number of columns as the input data and with n-nrow(data) rows. By default the values are drawn from a uniform distribution of values between the minimum and the maximum of the input data. Values can be drawn from background data instead if included.

Value

data frame of random numeric values with n-nrow(data) rows and same number of columns as input data

Examples

```
data(phosphonetworkdf)
data(datakin)
# only need what is present in data
phosphonetwork_data <- phosphonetwork_df[
phosphonetwork_df[,"SUB_IDENT"] %in% data_kin[,"SUB_IDENT"]
,]
fam <- list(akt = c("P31749", "P31751"))
kin_data_fam_exc <- KSR.list(phosphonetwork_data[, c("SUB_IDENT", "KIN_ACC_ID")],
                             kinasefamilies = fam,
                             exclusive = TRUE)
# only do for Akt and Mtor (P31749, P42345)
substrate_profiles <- lapply(kin_data_fam_exc[c("P31749", "P42345")],
function(x){data_kin[match(x, data_kin[,"SUB_IDENT"]),1:9]})

substrate_profiles_random <- lapply(substrate_profiles,
function(x){rbind(x, random.data(x, random.seed = 123))})
```

Index