Package ‘BioPlex’

May 21, 2026

Title R-side access to BioPlex protein-protein interaction data

Version 1.19.0

Description The BioPlex package implements access to the BioPlex protein-protein interaction networks and related resources from within R.
Besides protein-protein interaction networks for HEK293 and HCT116 cells, this includes access to CORUM protein complex data, and transcriptome and proteome data for the two cell lines.
Functionality focuses on importing the various data resources and storing them in dedicated Bioconductor data structures, as a foundation for integrative downstream analysis of the data.

URL `https://github.com/ccb-hms/BioPlex`

BugReports `https://github.com/ccb-hms/BioPlex/issues`

Encoding UTF-8

License Artistic-2.0

VignetteBuilder knitr

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0), SummarizedExperiment

Imports BiocFileCache, GEOquery, graph, methods, utils

Suggests AnnotationDbi, AnnotationHub, BiocStyle, DEXSeq, ExperimentHub, depmap, knitr, rmarkdown

biocViews CellCulture, ColonCancerData, ExperimentHub, ExpressionData, GEO, Genome, Homo_sapiens_Data, MassSpectrometryData, Proteome, ReproducibleResearch, RNASeqData

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

git_branch devel

git_last_commit 74eaec1

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-21

Author Ludwig Geistlinger [aut, cre],
Robert Gentleman [aut]

Maintainer Ludwig Geistlinger <`ludwig_geistlinger@hms.harvard.edu`>

Contents

`annotatePFAM` *Annotate PFAM domains to BioPlex PPI graph*

Description

This function adds PFAM domain annotations to the node metadata of the BioPlex PPI graph.

Usage

```
annotatePFAM(bp.gr, orgdb)
```

Arguments

`bp.gr`	an object of class `graph` storing the BioPlex PPIs. Typically obtained via `bioplex2graph`.
`orgdb`	an `orgdb` object storing annotation data for human.

Value

An object of class `graphNEL` containing PFAM domain annotations in the `nodeData`.

References

BioPlex: `https://bioplex.hms.harvard.edu/interactions.php`

PFAM: `http://pfam.xfam.org`

See Also

`nodeData`

Examples

```
# (1) Obtain the latest version of the 293T PPI network
bp.293t <- getBioPlex(cell.line = "293T", version = "3.0")

# (2) Turn the data into a graph
bp.gr <- bioplex2graph(bp.293t)

# (3) Obtain orgdb package from AnnotationHub
ah <- AnnotationHub::AnnotationHub()
orgdb <- AnnotationHub::query(ah, c("orgDb", "Homo sapiens"))
orgdb <- orgdb[[1]]

# (4) Annotate PFAM domains
bp.gr <- annotatePFAM(bp.gr, orgdb)
```

`bioplex2graph` *Representation of BioPlex PPIs in a graph data structure*

Description

Representation of BioPlex PPIs in a `graphNEL` object from the `graph` package.

Usage

```
bioplex2graph(bioplex.df)
```

Arguments

`bioplex.df`	a `data.frame` storing the Bioplex PPIs in a flat from-to format. Typically obtained via `getBioPlex`.

Value

An object of class `graphNEL`.

References

BioPlex: `https://bioplex.hms.harvard.edu/interactions.php`

See Also

`getBioPlex`, `ftM2graphNEL`

Examples

```
# (1) Obtain the latest version of the 293T PPI network
bp.293t <- getBioPlex(cell.line = "293T", version = "3.0")

# (2) Turn the data into a graph
bp.gr <- bioplex2graph(bp.293t)
```

`ccleProteome2SummarizedExperiment`

Convenient access to the CCLE proteome data

Description

Functionality for storing the protein expression data from the Cancer Cell Line Encyclopedia (CCLE) in a `SummarizedExperiment`.

Usage

```
ccleProteome2SummarizedExperiment(df, cell.line = "HCT116")
```

Arguments

`df`	a `data.frame` storing the CCLE protein expression data with one measurement in each row. Typically obtained from `ExperimentHub`. See examples.
`cell.line`	character. One or more cell line IDs such as `"HCT116"` (human colon cancer cell line 116). Use `NULL` to not subset by cell line. Defaults to `"HCT116"`, which will then subset the df to measurements for HCT116 only.

Value

A `SummarizedExperiment` storing protein expression data for the specified cell line(s).

References

CCLE proteomics: `https://gygi.hms.harvard.edu/publications/ccle.html`

Examples

```
# Connect to ExperimentHub
eh <- ExperimentHub::ExperimentHub()

# Obtain CCLE proteome data frame
AnnotationHub::query(eh, c("gygi", "depmap"))
ccle.prot <- eh[["EH3459"]]
ccle.prot <- as.data.frame(ccle.prot)

# Turn into a SummarizedExperiment
se <- ccleProteome2SummarizedExperiment(ccle.prot)
```

`corum2graphlist` *Represent CORUM protein complex data as a list of graph instances*

Description

Functionality for storing CORUM protein complex data in a `list` of `graph` instances.

Usage

```
corum2graphlist(corum.df, subunit.id.type = c("UNIPROT", "ENTREZID"))
```

Arguments

`corum.df`	A `data.frame` storing the CORUM protein complex data. Typically obtained via `getCorum`.
`subunit.id.type`	character. Supported options include `"UNIPROT"` (default) and `"ENTREZID"`.

Value

A `list` with an entry for each complex. Each entry is an object of class `graphNEL` connecting all subunit IDs with each other by undirected edges.

References

CORUM: `http://mips.helmholtz-muenchen.de/corum/#download`

Examples

```
# (1) Obtain the core set of CORUM complexes ...
core <- getCorum(set = "core")

# (2) ... turn into a list of graphs
core.glist <- corum2graphlist(core)
```

`corum2list` *Represent CORUM protein complex data as a simple list*

Description

Functionality for storing CORUM protein complex data in a `list`.

Usage

```
corum2list(corum.df, subunit.id.type = c("UNIPROT", "ENTREZID"))
```

Arguments

`corum.df`	A `data.frame` storing the CORUM protein complex data. Typically obtained via `getCorum`.
`subunit.id.type`	character. Supported options include `"UNIPROT"` (default) and `"ENTREZID"`.

Value

A `list` with an entry for each complex. Each entry is a character vector of subunit IDs.

References

CORUM: `http://mips.helmholtz-muenchen.de/corum/#download`

Examples

```
# (1) Obtain the core set of CORUM complexes ...
core <- getCorum(set = "core")

# (2) ... turn into a list
core.list <- corum2list(core)
```

`getBioPlex` *Obtain BioPlex protein-protein interaction data*

Description

Functionality for retrieving the BioPlex protein-protein interaction data. Available networks include:

- BioPlex 293T cells (versions 1.0, 2.0, and 3.0),
- BioPlex HCT116 cells (version 1.0).

See references.

Usage

```
getBioPlex(
  cell.line = c("293T", "HCT116"),
  version = c("3.0", "1.0", "2.0"),
  remap.uniprot.ids = FALSE,
  cache = TRUE
)
```

Arguments

`cell.line` character. Valid options include:

- `"293T"`: derivative of human embryonic kidney 293 cell line,
- `"HCT116"`: human colon cancer cell line 116.

Defaults to `"293T"`.

`version` character. Valid options include `"1.0"`, `"2.0"`, and `"3.0"` for 293T cells. For HCT116 cells, only `"1.0"` is available. Defaults to `"3.0"`.

`remap.uniprot.ids` logical. Should the protein-to-gene mappings from BioPlex (i.e. UNIPROT-to-SYMBOL and UNIPROT-to-ENTREZID) be updated using Bioc annotation functionality? Defaults to `FALSE` which will then keep the mappings provided by BioPlex.

`cache` logical. Should a locally cached version used if available? Defaults to `TRUE`.

Value

A `data.frame`.

References

BioPlex: https://bioplex.hms.harvard.edu/interactions.php

Examples

```
# (1) Obtain the latest version of the 293T PPI network
bp.293t <- getBioPlex(cell.line = "293T", version = "3.0")

# (2) Obtain the latest version of the HCT116 PPI network
bp.hct116 <- getBioPlex(cell.line = "HCT116", version = "1.0")
```

`getBioplexProteome` *Obtain BioPlex3 proteome data*

Description

Functionality for retrieving the BioPlex3 protein expression data comparing expression in the HCT116 and the 293T cell lines.

Usage

```
getBioplexProteome(cache = TRUE)
```

Arguments

`cache`	logical. Should a locally cached version used if available? Defaults to `TRUE`.

Value

A `SummarizedExperiment` storing protein expression data for the both cell line(s) with 5 replicates each.

References

BioPlex: https://bioplex.hms.harvard.edu

Examples

```
  se <- getBioplexProteome()
```

`getCorum` *Obtain CORUM protein complex data*

Description

Functionality for retrieving the CORUM protein complex data. Available complex collections include:

- complete set of complexes,
- core set of complexes,
- complexes with splice variants.

See references.

Usage

```
getCorum(
  set = c("all", "core", "splice"),
  organism = "Human",
  remap.uniprot.ids = FALSE,
  cache = TRUE,
  mode = c("ehub", "web")
)
```

Arguments

`set`	character. Valid options include: • `"all"`: complete set of complexes, • `"core"`: core set of complexes, • `"splice"`: complexes with splice variants. Defaults to `"all"`.
`organism`	character. Use `NULL` to not subset by organism. Defaults to `"Human"` which restricts the data to human protein complexes only.
`remap.uniprot.ids`	logical. Should the protein-to-gene mappings from CORUM (i.e. UNIPROT-to-SYMBOL and UNIPROT-to-ENTREZID) be updated using Bioc annotation functionality? Currently only supported in combination with `organism = "Human"`. Defaults to `FALSE` which will then keep the mappings provided by CORUM.
`cache`	logical. Should a locally cached version used if available? Defaults to `TRUE`.
`mode`	character. Should CORUM complexes be obtained from ExperimentHub or via a web download from the CORUM homepage? Defaults to `"ehub"`, which will obtain the chosen complex set from ExperimentHub.

Value

A `data.frame`.

References

CORUM: `http://mips.helmholtz-muenchen.de/corum/#download`

Examples

```
# Obtain the core set of CORUM complexes
core <- getCorum(set = "core")
```

`getGSE122425` *Convenient access to 293T transcriptome data from GEO*

Description

Functionality for storing the 293T RNA-seq data from GSE122425 in a `SummarizedExperiment`. The dataset includes three wild type samples and three NSUN2 knockout samples.

Usage

```
getGSE122425(cache = TRUE)
```

Arguments

`cache`	logical. Should a locally cached version used if available? Defaults to `TRUE`.

Value

A `SummarizedExperiment` storing RNA-seq data for the 293T cell line.

References

GSE122425: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE122425`

Examples

```
# Obtain the data as a SummarizedExperiment
se <- getGSE122425()
```

`hasSubunit` *Identify CORUM complexes that have a subunit of interest*

Description

Screens a `list` of `graph` instances storing CORUM protein complex data for a subunit of choice.

Usage

```
hasSubunit(glist, subunit, id.type = "SYMBOL")
```

Arguments

`glist`	A `list` of `graph`s storing CORUM complexes. Typically obtained via `corum2graphlist`.
`subunit`	character. A gene ID corresponding to the subunit of interest.
`id.type`	character. Gene ID type of the given subunit. Defaults to `"SYMBOL"`.

Value

A logical vector indicating which graphs have a node with the given subunit.

Examples

```
# (1) Obtain the core set of CORUM complexes ...
core <- getCorum(set = "core")

# (2) ... turn into a list of graphs ...
core.glist <- corum2graphlist(core)

# (3) .. check for a particular subunit of interest
has.cdk2 <- hasSubunit(core.glist, subunit = "CDK2")
```

`mapSummarizedExperimentOntoGraph`

Map experimental data onto a graph

Description

Functionality for mapping experimental data stored in a `SummarizedExperiment` onto a `graph` object.

Usage

```
mapSummarizedExperimentOntoGraph(
  gr,
  se,
  col.names = NULL,
  rowdata.cols = NULL,
  prefix = ""
)
```

Arguments

`gr`	an object of class `graph`.
`se`	an object of class `SummarizedExperiment`.
`col.names`	character. Column names of se for which assay data should be mapped onto the nodes of `gr`. Defaults to `NULL` which will then use all column names of `se`.
`rowdata.cols`	character. Column names of `rowData(se)` which should be mapped onto the nodes of `gr`. Defaults to `NULL` which will then use all column names of `rowData(se)`.
`prefix`	character. Informative prefix that should be pasted together with the selected `col.names` and `rowdata.cols` to allow easy identification of columns of interest when mapping from multiple experimental datasets.

Value

An object of class `graph`.

Examples

```
# (1) Obtain the latest version of the 293T PPI network ...
bp.293t <- getBioPlex(cell.line = "293T", version = "3.0")

# (2) ... and turn into a graph
bp.gr <- bioplex2graph(bp.293t)

# (3) Obtain the BioPlex3 proteome data ...
se <- getBioplexProteome()

# (4) ... and map onto the graph
bp.gr <- mapSummarizedExperimentOntoGraph(bp.gr, se)
```

Index