Package ‘planttfhunter’

April 30, 2024

Title Identification and classification of plant transcription factors

Version 1.3.0

Date 2022-03-10

Description planttfhunter is used to identify plant transcription factors (TFs) from protein sequence data and classify them into families and subfamilies using the classification scheme implemented in PlantTFDB.
TFs are identified using pre-built hidden Markov model profiles for DNA-binding domains. Then, auxiliary and forbidden domains are used with DNA-binding domains to classify TFs into families and subfamilies (when applicable). Currently, TFs can be classified in 58 different TF families/subfamilies.

License GPL-3

URL https://github.com/almeidasilvaf/planttfhunter

BugReports https://support.bioconductor.org/t/planttfhunter

biocViews Software, Transcription, FunctionalPrediction, GenomeAnnotation, FunctionalGenomics, HiddenMarkovModel, Sequencing, Classification

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

SystemRequirements HMMER <http://hmmer.org/>

Imports Biostrings, SummarizedExperiment, utils, methods

Suggests BiocStyle, covr, sessioninfo, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 4.2.0)

LazyData false

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

git_branch devel

git_last_commit 8c49f84

git_last_commit_date 2023-10-24

Repository Bioconductor 3.20

Date/Publication 2024-04-29

Author Fabrício Almeida-Silva [aut, cre]
(<https://orcid.org/0000-0002-5314-2964>),
Yves Van de Peer [aut] (<https://orcid.org/0000-0003-4327-3730>)

Maintainer Fabrício Almeida-Silva <fabricio_almeidasilva@hotmail.com>

Contents

`planttfhunter-package` *planttfhunter: Identification and classification of plant transcription factors*

Description

planttfhunter is used to identify plant transcription factors (TFs) from protein sequence data and classify them into families and subfamilies using the classification scheme implemented in PlantTFDB. TFs are identified using pre-built hidden Markov model profiles for DNA-binding domains. Then, auxiliary and forbidden domains are used with DNA-binding domains to classify TFs into families and subfamilies (when applicable). Currently, TFs can be classified in 58 different TF families/subfamilies.

Author(s)

Maintainer: Fabrício Almeida-Silva <fabricio_almeidasilva@hotmail.com> (ORCID)

Authors:

- Yves Van de Peer <yves.vandepeer@psb.vib-ugent.be> (ORCID)

See Also

Useful links:

- https://github.com/almeidasilvaf/planttfhunter
- Report bugs at https://support.bioconductor.org/t/planttfhunter

---

annotate_pfam *Annotate proteins sequences with PFAM domains*

Description

PFAM domains are assigned to each sequence using HMMER.

Usage

```
annotate_pfam(seq = NULL, evalue = 1e-05)
```

Arguments

seq	An AAStringSet object as returned by `Biostrings::readAAStringSet()`. The sequences in this object must represent only the translated sequences of primary (or longest) transcripts.
evalue	Numeric indicating the E-value threshold for hmmsearch to be used for domains without pre-defined domain cutoffs. Only valid if parameter mode = 'local'. Default: 1e-05.

Value

A 2-column data frame with the variables **Gene** and **Domain**, which contain gene IDs and domain IDs, respectively.

Examples

```
data(gsu)
seq <- gsu[1:5]
if(hmmer_is_installed()) {
    annotate_pfam(seq)
}
```

classification_scheme *Data frame of TF family classification scheme*

Description

The classification scheme is the same as the one used by PlantTFDB.

Usage

```
data(classification_scheme)
```

Format

A data frame with the following variables:

Family TF family name.

Subfamily TF subfamily name.

DBD DNA-binding domain

Auxiliary Auxiliary domain

Forbidden Forbidden domain

References

Jin, J., Tian, F., Yang, D. C., Meng, Y. Q., Kong, L., Luo, J., & Gao, G. (2016). PlantTFDB 4.0: toward a central hub for transcription factors and regulatory interactions in plants. Nucleic acids research, gkw982.

Examples

```
data(classification_scheme)
```

classify_tfs *Identify TFs and classify them in families*

Description

Identify TFs and classify them in families

Usage

```
classify_tfs(domain_annotation = NULL)
```

Arguments

`domain_annotation`
: A 2-column data frame with the gene ID in the first column and the domain ID in the second column.

Value

A 2-column data frame with the variables **Gene** and **Family** representing gene ID and TF family, respectively.

Examples

```
data(gsu_annotation)
domain_annotation <- gsu_annotation
families <- classify_tfs(domain_annotation)
```

---

get_tf_counts — *Get TF frequencies for each species as a SummarizedExperiment object*

Description

This function identifies and classifies TFs, and returns TF counts for each family as a SummarizedExperiment object

Usage

```
get_tf_counts(proteomes, species_metadata = NULL)
```

Arguments

`proteomes` List of **AAStringSet** objects

`species_metadata` (Optional) A data frame containing species names in row names (names must match element names in the **proteomes** list), and species metadata (e.g., taxonomic information, ecological information) in columns. If NULL, the colData of the SummarizedExperiment object will be empty.

Value

A SummarizedExperiment object containing transcription factor frequencies per family in each species, as well as species metadata (if **species_metadata** is not NULL).

Examples

```
data(gsu)

set.seed(123)
# Pick random subsets of 100 genes to simulate other species
proteomes <- list(
    Gsu1 = gsu[sample(names(gsu), 50, replace = FALSE)],
    Gsu2 = gsu[sample(names(gsu), 50, replace = FALSE)],
    Gsu3 = gsu[sample(names(gsu), 50, replace = FALSE)],
    Gsu4 = gsu[sample(names(gsu), 50, replace = FALSE)]
```

```
)

# Create species metadata
species_metadata <- data.frame(
    row.names = names(proteomes),
    Division = "Rhodophyta",
    Origin = c("US", "Belgium", "China", "Brazil")
)

# Get SummarizedExperiment object
if(hmmer_is_installed()) {
    se <- get_tf_counts(proteomes, species_metadata)
}
```

---

gsu *Protein sequences of the algae species Galdieria sulphuraria*

Description

Data obtained from PLAZA Diatoms. Only genes containing domains used for TF family classification were kept for package size issues.

Usage

```
data(gsu)
```

Format

An AAStringSet object as returned by `Biostrings::readAAStringSet()`.

References

Osuna-Cruz, C. M., Bilcke, G., Vancaester, E., De Decker, S., Bones, A. M., Winge, P., ... & Vandepoele, K. (2020). The Seminavis robusta genome provides insights into the evolutionary adaptations of benthic diatoms. Nature communications, 11(1), 1-13.

Examples

```
data(gsu)
```

`gsu_annotation` *Domain annotation for the algae species Galdieria sulphuraria The data set was created using the funcion* `annotate_pfam()` *in local mode.*

Description

Domain annotation for the algae species Galdieria sulphuraria

The data set was created using the funcion `annotate_pfam()` in local mode.

Usage

```
data(gsu_annotation)
```

Format

A 2-column data frame with the following variables:

Gene Gene ID

Annotation Domain ID or domain name when ID is not available in PFAM

Examples

```
data(gsu_annotation)
```

`gsu_families` *TFs families of the algae species Galdieria sulphuraria The data set was created using the funcion* `classify_tfs()`.

Description

TFs families of the algae species Galdieria sulphuraria

The data set was created using the funcion `classify_tfs()`.

Usage

```
data(gsu_families)
```

Format

A 2-column data frame with the following variables:

Gene Gene ID

Family TF family

Examples

```
data(gsu_families)
```

hmmer_is_installed *Check if HMMER is installed*

Description

Check if HMMER is installed

Usage

```
hmmer_is_installed()
```

Value

Logical indicating whether HMMER is installed or not.

Examples

```
hmmer_is_installed()
```

tf_counts *TF counts per family in 4 simulated species*

Description

Simulated species were created by sampling 100 genes from the example data set **gsu** with after `set.seed(123)`.

Usage

```
data(tf_counts)
```

Format

A SummarizedExperiment with TF frequencies per family in each species in **assay** and species metadata in **colData**.

Examples

```
data(tf_counts)
```

Index