# Package: planttfhunter (via r-universe)

October 7, 2026

**Title** Identification and classification of plant transcription factors

**Version** 1.13.1

**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 schemes implemented in PlantTFDB and TAPscan. TFs are also classified into structure-based families, classes, and superclasses as in Plant-TFClass. 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). TFs can be classified in 58 different TF families/subfamilies with PlantTFDB's scheme, and in over 130 families of transcription-associated proteins with TAPscan's scheme.

**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

**Config/pak/sysreqs** zlib1g-dev

**Repository** https://bioc.r-universe.dev

**Date/Publication** 2026-10-06 08:23:03 UTC

**RemoteUrl** https://github.com/bioc/planttfhunter

**RemoteRef** HEAD

**RemoteSha** e32b42f93a007723467da1a361c894c5e24c4b52

# Contents

annotate_domains . . . 2
classify_tfs . . . 3
get_tf_counts . . . 5
gsu . . . 7
gsu_annotation . . . 7
hmmer_is_installed . . . 8
planttfclass_scheme . . . 8
planttfdb_scheme . . . 9
tap_class . . . 10
tapscan_scheme . . . 10

**Index** **12**

---

`annotate_domains` *Annotate protein sequences with domains used for TF classification*

---

## Description

Protein sequences are scanned with HMMER for domains used by the classification schemes of **PlantTFDB** and **TAPscan**. For PlantTFDB, PFAM and self-built profile HMMs are used with the domain-specific score cutoffs defined by PlantTFDB (or an E-value threshold, for domains without cutoffs). For TAPscan, TAPscan v4 profile HMMs are used with their gathering (GA) thresholds, as in the original TAPscan implementation.

## Usage

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

### Arguments

| | |
|---|---|
| seq | An AAStringSet object. The sequences in this object must represent only the translated sequences of primary (or longest) transcripts. To create such an object from a FASTA file, use Biostrings::readAAStringSet(). |
| evalue | Numeric indicating the E-value threshold for hmmsearch to be used for Plant-TFDB domains without pre-defined domain cutoffs. Default: 1e-05. |
| threads | Numeric indicating the number of threads to be used by hmmsearch. Default: 1. |

### Value

A list of 2 data frames named **PlantTFDB** and **TAPscan**, with domain annotation for each classification scheme. Both data frames have one row per domain hit and the following variables:

**Gene** Character, gene ID.

**Domain** Character, domain ID or name.

**qlen** Numeric, length of the profile HMM.

**c_evalue** Numeric, conditional E-value of the domain hit.

**hmm_from, hmm_to** Numeric, start and end coordinates of the alignment in the profile HMM.

Rows are kept in the same order as in the HMMER output, as this order is used by the TAPscan classification algorithm.

### Examples

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

---

## classify_tfs *Identify TFs and classify them in families*

---

### Description

TFs are classified using the classification schemes of PlantTFDB, TAPscan, and Plant-TFClass.

### Usage

```
classify_tfs(
  domain_annotation = NULL,
  scheme = c("all", "PlantTFDB", "TAPscan", "PlantTFClass")
)
```

## Arguments

| | |
|---|---|
| `domain_annotation` | A list of data frames with domain annotation, as returned by `annotate_domains()`. |
| `scheme` | Character indicating which classification scheme(s) to return. One of 'all' (all schemes), 'PlantTFDB', 'TAPscan', or 'PlantTFClass'. Default: 'all'. |

## Details

PlantTFDB and TAPscan classifications are based on the domains found in each gene. Plant-TFClass is a structure-based classification of TFs into superclasses, classes, and families, and Plant-TFClass classifications are inferred from TAPscan classifications using a correspondence table between the two classification systems. When a TAPscan subfamily corresponds to more than one Plant-TFClass family (e.g., TAPscan's AP2 includes Plant-TFClass's AP2, ERF/DREB, and RAV), PlantTFDB subfamilies are used to resolve the ambiguity. If the ambiguity cannot be resolved (e.g., the gene was not classified by PlantTFDB), all possible classifications are reported, separated by ';'.

In all schemes, classification levels without a more specific level are filled with the name of the more general level. For example, PlantTFDB family bHLH has no subfamilies, so its subfamily is bHLH.

The classification schemes are available in the data sets `planttfdb_scheme` (PlantTFDB), `tapscan_scheme` (TAPscan), and `planttfclass_scheme` (correspondence between TAPscan and Plant-TFClass).

## Value

A data frame with one row per gene and the following variables (if **scheme = "all"**):

**Gene** Gene ID.

**PlantTFDB_family** TF family according to PlantTFDB.

**PlantTFDB_subfamily** TF subfamily according to PlantTFDB.

**TAPscan_family** TF family according to TAPscan.

**TAPscan_subfamily** TF subfamily according to TAPscan.

**TAP_class** Class of transcription-associated protein (TAP) according to TAPscan: transcription factor (TF), transcriptional regulator (TR), or putative TAP (PT). See `data(tap_class)`.

**PlantTFClass_superclass** TF superclass according to Plant-TFClass.

**PlantTFClass_class** TF class according to Plant-TFClass.

**PlantTFClass_family** TF family according to Plant-TFClass.

Only genes classified by at least one scheme are included. Genes not classified by a scheme have NA in the respective variables. Genes assigned to more than one PlantTFDB family have families (and subfamilies, in the same order) separated by ';'.

If **scheme** is 'PlantTFDB', 'TAPscan', or 'PlantTFClass', only variable **Gene** and the variables of the chosen scheme are returned, and only genes classified by that scheme are included. Note that TAPscan classifies not only TFs, but also transcriptional regulators and putative TAPs, which can be distinguished using the variable **TAP_class**.

## References

Jin, J., Tian, F., Yang, D. C., Meng, Y. Q., Kong, L., Luo, J., & Gao, G. (2017). PlantTFDB 4.0: toward a central hub for transcription factors and regulatory interactions in plants. Nucleic Acids Research, 45(D1), D1040-D1045.

Petroll, R., Varshney, D., Hiltemann, S., Finke, H., Schreiber, M., de Vries, J., & Rensing, S. A. (2025). Enhanced sensitivity of TAPscan v4 enables comprehensive analysis of streptophyte transcription factor evolution. The Plant Journal, 121(1), e17184.

Blanc-Mathieu, R., Dumas, R., Turchi, L., Lucas, J., & Parcy, F. (2024). Plant-TFClass: a structural classification for plant transcription factors. Trends in Plant Science, 29(1), 40-51.

## Examples

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

# Only TAPscan classifications, with TAP classes (TF, TR, or PT)
tapscan_families <- classify_tfs(gsu_annotation, scheme = "TAPscan")
table(tapscan_families$TAP_class)
```

---

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

---

## Description

This function counts TFs per family (or other classification level) in each species from TF classifications obtained with `classify_tfs()`, and returns the counts as a `SummarizedExperiment` object.

## Usage

```
get_tf_counts(
  families,
  species_metadata = NULL,
  scheme = c("PlantTFDB", "TAPscan", "PlantTFClass"),
  level = "family"
)
```

## Arguments

| | |
|---|---|
| `families` | A named list of data frames with TF classifications for each species, as returned by `classify_tfs()`. List names represent species names. |
| `species_metadata` | (Optional) A data frame containing species names in row names (names must match element names in the **families** list), and species metadata (e.g., taxonomic information, ecological information) in columns. If NULL, the colData of the `SummarizedExperiment` object will be empty. |

| | |
|---|---|
| scheme | Character indicating which classification scheme to use. One of 'PlantTFDB', 'TAPscan', or 'PlantTFClass'. Default: 'PlantTFDB'. |
| level | Character indicating the classification level at which TFs will be counted. Possible levels depend on the classification scheme: 'family' or 'subfamily' for PlantTFDB and TAPscan, and 'superclass', 'class', or 'family' for Plant-TFClass. Default: 'family'. |

## Value

A SummarizedExperiment object containing TF frequencies per family (or other level) in each species, as well as species metadata (if **species_metadata** is not NULL). The rowData of the object contains the more general classification levels of each row (if any), which can be used to aggregate counts, as well as the TAP class of each row for TAPscan. For PlantTFDB, genes assigned to more than one family are counted once for each family. For Plant-TFClass, genes with ambiguous classifications (i.e., multiple possible families separated by ';') are counted in a separate row with all possible families.

## Examples

```
data(gsu_annotation)
gsu_families <- classify_tfs(gsu_annotation)

# Simulate TF classifications for 4 species by sampling 100 genes
set.seed(123)
families <- list(
    Gsu1 = gsu_families[sample(nrow(gsu_families), 100), ],
    Gsu2 = gsu_families[sample(nrow(gsu_families), 100), ],
    Gsu3 = gsu_families[sample(nrow(gsu_families), 100), ],
    Gsu4 = gsu_families[sample(nrow(gsu_families), 100), ]
)

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

# Count TFs per PlantTFDB family
se <- get_tf_counts(families, species_metadata)

# Count TAPscan TFs (excluding TRs and PTs) per subfamily
tapscan_tfs <- lapply(families, function(x) x[x$TAP_class %in% "TF", ])
se_tapscan <- get_tf_counts(
    tapscan_tfs, scheme = "TAPscan", level = "subfamily"
)
se_tapscan

# Count TFs per Plant-TFClass superclass
se_superclass <- get_tf_counts(
    families, scheme = "PlantTFClass", level = "superclass"
```

```
)
```

---

`gsu` *Protein sequences of the algae species Galdieria sulphuraria*

---

## Description

Data obtained from PLAZA Diatoms. Only genes containing domains used for TF family classification (with PlantTFDB's or TAPscan's schemes) 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 function* `annotate_domains()`.

---

## Description

Domain annotation for the algae species Galdieria sulphuraria

The data set was created using the function `annotate_domains()`.

## Usage

```
data(gsu_annotation)
```

## Format

A list of 2 data frames (**PlantTFDB** and **TAPscan**) with domain annotation, as returned by `annotate_domains()`.

## Examples

```
data(gsu_annotation)
```

## 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 (i.e., whether the program `hmmsearch` is in the PATH) or not.

### Examples

```
hmmer_is_installed()
```

## planttfclass_scheme *Correspondence table between TAPscan and Plant-TFClass*

### Description

Table used by `classify_tfs()` to infer Plant-TFClass superclasses, classes, and families from TAPscan families and subfamilies. TAPscan families/subfamilies with multiple rows correspond to more than one Plant-TFClass family, and the PlantTFDB subfamily of each gene (variable **PlantTFDB_subfamily**) is used to choose among them.

### Usage

```
data(planttfclass_scheme)
```

### Format

A data frame with the following variables:

**TAPscan_family** TAPscan family.

**TAPscan_subfamily** TAPscan subfamily (or family, for families without subfamilies).

**PlantTFClass_superclass** Plant-TFClass superclass.

**PlantTFClass_class** Plant-TFClass class.

**PlantTFClass_family** Plant-TFClass family (or class, for classes without families).

**PlantTFDB_subfamily** PlantTFDB subfamily used to choose among multiple Plant-TFClass families for the same TAPscan family/subfamily. NA for TAPscan families/subfamilies with a single Plant-TFClass family.

## References

Blanc-Mathieu, R., Dumas, R., Turchi, L., Lucas, J., & Parcy, F. (2024). Plant-TFClass: a structural classification for plant transcription factors. Trends in Plant Science, 29(1), 40-51.

## Examples

```
data(planttfclass_scheme)
```

---

`planttfdb_scheme` *Data frame of PlantTFDB's TF family classification scheme*

---

## Description

The classification scheme is the same as the one used by PlantTFDB, except for three family/subfamily names that were changed to match the output of `classify_tfs()` ('M_type' to 'M-type', 'MYB_related' to 'MYB-related', and 'LBD (AS2/LOB)' to 'LBD'). As in PlantTFDB, families without subfamilies have the family name as subfamily.

## Usage

```
data(planttfdb_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(planttfdb_scheme)
```

## tap_class *Classes of transcription-associated proteins (TAPs) in TAPscan*

### Description

Families of transcription-associated proteins (TAPs) in TAPscan v4, their classes, descriptions, and references. Data were obtained from TAPscan's website (file *_data/import-tapinfo/tapinfo_v4.csv* in the GitHub repository Rensing-Lab/TAPscan-v4-website). HTML tags in descriptions and references were removed. Family MADS_MIKC, which is not included in the original file, was manually added as a TF family (as family MADS), without description and references.

### Usage

```
data(tap_class)
```

### Format

A data frame with the following variables:

**TAP_family** TAP family or subfamily, as in the variable **TAPscan_subfamily** of the output of `classify_tfs()`.

**TAP_class** TAP class. One of 'TF' (transcription factor), 'TR' (transcriptional regulator), or 'PT' (putative TAP).

**Description** Description of the TAP family.

**References** References for the TAP family, separated by ' | '.

### References

Petroll, R., Varshney, D., Hiltemann, S., Finke, H., Schreiber, M., de Vries, J., & Rensing, S. A. (2025). Enhanced sensitivity of TAPscan v4 enables comprehensive analysis of streptophyte transcription factor evolution. The Plant Journal, 121(1), e17184.

### Examples

```
data(tap_class)
```

## tapscan_scheme *Data frame of TAPscan's classification scheme*

### Description

TAPscan v4 classification rules (v82), with one row per rule. Genes are assigned to a family (and subfamily, if any) if they have all domains in variable **Should** and none of the domains in variable **Should_not**. For families with more than one rule (e.g., bZIP), genes are assigned to the family if they satisfy any of the rules.

## Usage

```
data(tapscan_scheme)
```

## Format

A data frame with the following variables:

**Family** TAPscan family, as in variable **TAPscan_family** of the output of `classify_tfs()`.

**Subfamily** TAPscan subfamily, as in variable **TAPscan_subfamily** of the output of `classify_tfs()`. Families without subfamilies have the family name as subfamily.

**Rule** Name of the rule in TAPscan's rules file.

**Should** Domains that genes should have, separated by ' and '. Numbers in parentheses (e.g., 'Myb_DNA-binding (2)') indicate the required number of copies of a domain.

**Should_not** Domains that genes should not have, separated by ' or '.

**TAP_class** TAP class. One of 'TF' (transcription factor), 'TR' (transcriptional regulator), or 'PT' (putative TAP). NA for families without a TAP class in TAPscan (see `tap_class`).

## References

Petroll, R., Varshney, D., Hiltemann, S., Finke, H., Schreiber, M., de Vries, J., & Rensing, S. A. (2025). Enhanced sensitivity of TAPscan v4 enables comprehensive analysis of streptophyte transcription factor evolution. The Plant Journal, 121(1), e17184.

## Examples

```
data(tapscan_scheme)
```

# Index

∗ **datasets**
gsu, 7
gsu_annotation, 7
planttfclass_scheme, 8
planttfdb_scheme, 9
tap_class, 10
tapscan_scheme, 10

annotate_domains, 2

classify_tfs, 3

get_tf_counts, 5
gsu, 7
gsu_annotation, 7

hmmer_is_installed, 8

planttfclass_scheme, 8
planttfdb_scheme, 9

tap_class, 10
tapscan_scheme, 10