

# Package ‘pureseqtmr’

July 23, 2025

**Title** Predict Transmembrane Protein Topology

**Version** 1.4

**Description** Proteins reside in either the cell plasma or in the cell membrane. A membrane protein goes through the membrane at least once. Given the amino acid sequence of a membrane protein, the tool 'PureseqTM' (<[https://github.com/PureseqTM/pureseqTM\\_package](https://github.com/PureseqTM/pureseqTM_package)>, as described in ``Efficient And Accurate Prediction Of Transmembrane Topology From Amino acid sequence only." , Wang, Qing, et al (2019), <[doi:10.1101/627307](https://doi.org/10.1101/627307)>), can predict the topology of a membrane protein. This package allows one to use 'PureseqTM' from R.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 3.5.0)

**Imports** data.table, devtools, dplyr, ggplot2, Peptides, plyr, rappdirs, readr, stringr, tibble, Rcpp

**Suggests** testthat, knitr, markdown, rmarkdown, profvis

**URL** <https://github.com/richelbilderbeek/pureseqtmr/>

**BugReports** <https://github.com/richelbilderbeek/pureseqtmr/>

**VignetteBuilder** knitr

**SystemRequirements** PureseqTM  
([https://github.com/PureseqTM/pureseqTM\\_package](https://github.com/PureseqTM/pureseqTM_package))

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Author** Richèl J.C. Bilderbeek [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-1107-7049>>)

**Maintainer** Richèl J.C. Bilderbeek <[richel@richelbilderbeek.nl](mailto:richel@richelbilderbeek.nl)>

**Repository** CRAN

**Date/Publication** 2023-04-06 13:40:02 UTC

## Contents

|                                                                 |    |
|-----------------------------------------------------------------|----|
| are_tmhs . . . . .                                              | 3  |
| are_valid_protein_sequences . . . . .                           | 4  |
| calc_distance_to_tmh_center_from_topology . . . . .             | 4  |
| calc_distance_to_tmh_center_from_topology_str . . . . .         | 5  |
| calc_distance_to_tmh_center_from_topology_str_cpp_stl . . . . . | 5  |
| check_protein_sequence . . . . .                                | 6  |
| check_protein_sequences . . . . .                               | 6  |
| check_pureseqtm_installation . . . . .                          | 7  |
| check_topology . . . . .                                        | 8  |
| check_topology_str . . . . .                                    | 9  |
| convert_tmhmm_to_pureseqtm_topology . . . . .                   | 9  |
| count_n_tmhs . . . . .                                          | 10 |
| create_pureseqtm_files . . . . .                                | 10 |
| create_pureseqtm_proteome_file . . . . .                        | 11 |
| default_params_doc . . . . .                                    | 12 |
| get_default_pureseqtm_folder . . . . .                          | 14 |
| get_example_filename . . . . .                                  | 14 |
| get_example_filenames . . . . .                                 | 15 |
| get_pureseqtm_url . . . . .                                     | 16 |
| get_pureseqtm_version . . . . .                                 | 16 |
| install_pureseqtm . . . . .                                     | 17 |
| is_on_appveyor . . . . .                                        | 18 |
| is_on_ci . . . . .                                              | 18 |
| is_on_github_actions . . . . .                                  | 19 |
| is_on_travis . . . . .                                          | 19 |
| is_protein_name_line . . . . .                                  | 20 |
| is_pureseqtm_installed . . . . .                                | 20 |
| is_tmh . . . . .                                                | 21 |
| is_topology_line . . . . .                                      | 22 |
| is_valid_protein_sequence . . . . .                             | 22 |
| load_fasta_file_as_tibble . . . . .                             | 23 |
| load_fasta_file_as_tibble_cpp . . . . .                         | 24 |
| load_fasta_file_as_tibble_cpp_raw . . . . .                     | 24 |
| load_fasta_file_as_tibble_r . . . . .                           | 25 |
| load_topology_file_as_tibble . . . . .                          | 25 |
| mock_predict_topologies_from_sequences . . . . .                | 26 |
| mock_predict_topology . . . . .                                 | 27 |
| parse_pureseqtm_proteome_text . . . . .                         | 27 |
| plot_topology . . . . .                                         | 28 |
| predict_topologies_from_sequences . . . . .                     | 29 |
| predict_topology . . . . .                                      | 30 |
| predict_topology_from_sequence . . . . .                        | 31 |
| pureseqtmr . . . . .                                            | 32 |
| pureseqtmr_report . . . . .                                     | 32 |
| run_pureseqtm_proteome . . . . .                                | 33 |
| save_tibble_as_fasta_file . . . . .                             | 34 |

|                                            |           |
|--------------------------------------------|-----------|
| <code>are_tmhs</code>                      | 3         |
| <code>tally_tmhs</code> . . . . .          | 34        |
| <code>uninstall_pureseqtm</code> . . . . . | 35        |
| <b>Index</b>                               | <b>36</b> |

---

|                       |                                                 |
|-----------------------|-------------------------------------------------|
| <code>are_tmhs</code> | <i>Are the sequences transmembrane helices?</i> |
|-----------------------|-------------------------------------------------|

---

**Description**

Are the sequences transmembrane helices?

**Usage**

```
are_tmhs(protein_sequences, folder_name = get_default_pureseqtm_folder())
```

**Arguments**

- `protein_sequences`  
one ore more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL
- `folder_name`  
superfolder of PureseqTM. The superfolder's name is /home/[user\_name]/.local/share by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

**Value**

a vector of booleans of the same length as the number of sequences. The ith element is **TRUE** if the ith protein sequence is a transmembrane helix

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_pureseqtm_installed()) {  
  sequences <- c(  
    "QKNWSALLTAVVIILTIAGNILVIMAVSLEKKLQATNYFLM",  
    "VVIILTIRGNILVIMAVSLE"  
  )  
  are_tmhs(sequences)  
}
```

---

```
are_valid_protein_sequences
```

*Determine if these are all valid protein sequences*

---

### Description

Determine if these are all valid protein sequences, as can be used in topology prediction

### Usage

```
are_valid_protein_sequences(protein_sequences, verbose = FALSE)
```

### Arguments

protein\_sequences

one ore more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL

verbose

set to TRUE for more output

### Value

TRUE if the protein sequence is valid

---

```
calc_distance_to_tmh_center_from_topology
```

*Calculate the the distance for each amino acid to the center of the TMH*

---

### Description

Calculate the the distance for each amino acid to the center of the TMH

### Usage

```
calc_distance_to_tmh_center_from_topology(topology)
```

### Arguments

topology

the topology as a [tibble](#) with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains the respective topologies as strings.

### Value

a [tibble](#) with the columns 'name' and 'position' and 'distance\_to\_tmh\_center'

**Author(s)**

Richèl J.C. Bilderbeek

---

calc\_distance\_to\_tmh\_center\_from\_topology\_str

*Calculate the the distance for each amino acid to the center of the TMH*

---

**Description**

Calculate the the distance for each amino acid to the center of the TMH

**Usage**

calc\_distance\_to\_tmh\_center\_from\_topology\_str(topology\_str)

**Arguments**

topology\_str    the topology as a string, for example 000000111100000

**Value**

a [tibble](#) with the columns 'position' and 'distance\_to\_tmh\_center'

**Author(s)**

Richèl J.C. Bilderbeek

---

calc\_distance\_to\_tmh\_center\_from\_topology\_str\_cpp\_stl

*Use Rcpp to calculate the distance to a TMH center*

---

**Description**

Use Rcpp to calculate the distance to a TMH center

**Usage**

calc\_distance\_to\_tmh\_center\_from\_topology\_str\_cpp\_stl(topology\_str)

**Arguments**

topology\_str    a topology as a string

**Value**

a vector with distances

check\_protein\_sequence

*Check one protein sequence*

---

### Description

Will [stop](#) if the protein sequence is invalid, with a helpful error message.

### Usage

```
check_protein_sequence(protein_sequence)
```

### Arguments

protein\_sequence

a protein sequence, with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL.  
Use [check\\_protein\\_sequence](#) to check if a protein sequence is valid.

### Details

A protein sequence is invalid if:

- it has zero, two or more sequences
- the sequence contains zero, 1 or 2 amino acids
- the sequence contains characters that are not in the amino acid uppercase alphabet, that is ACDEFGHIKLMNPQRSTVWY

### Value

nothing. Will [stop](#) if the protein sequence is invalid, with a helpful error message.

### Examples

```
check_protein_sequence("FAMILYVW")
```

---

check\_protein\_sequences

*Check one or more protein sequences*

---

### Description

Will [stop](#) if the protein sequence is invalid, with a helpful error message.

### Usage

```
check_protein_sequences(protein_sequences)
```

**Arguments**

protein\_sequences

one or more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL

**Details**

A protein sequence is invalid if:

- it has zero, two or more sequences
- the sequence contains zero, 1 or 2 amino acids
- the sequence contains characters that are not in the amino acid uppercase alphabet, that is ACDEFGHIKLMNPQRSTVWY

**Value**

nothing. Will [stop](#) at the first invalid protein sequence, with a helpful error message.

**Examples**

```
check_protein_sequences(c("FAMILYVW", "FAMILYVW"))
```

---

check\_pureseqtm\_installation

*Checks the installation of PureseqTM. Throws a helpful error message if incomplete, else does nothing*

---

**Description**

Checks the installation of PureseqTM. Throws a helpful error message if incomplete, else does nothing

**Usage**

```
check_pureseqtm_installation(folder_name = get_default_pureseqtm_folder())
```

**Arguments**

folder\_name      superfolder of PureseqTM. The superfolder's name is /home/[user\_name]/.local/share by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

**Value**

Nothing. Will [stop](#) with a helpful error message if PureseqTM is not installed.

**Author(s)**

Richèl J.C. Bilderbeek

## Examples

```
if (is_pureseqtm_installed()) {  
  check_pureseqtm_installation()  
}
```

---

|                |                                        |
|----------------|----------------------------------------|
| check_topology | <i>Check if the topology is valid.</i> |
|----------------|----------------------------------------|

---

## Description

Check if the argument is of the same type as a predicted topology, as can be created with [predict\\_topology](#). Will **stop** if not.

## Usage

```
check_topology(topology)
```

## Arguments

|          |                                                                                                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| topology | the topology as a <a href="#">tibble</a> with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains the respective topologies as strings. |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Value

Nothing. Will **stop** with a helpful error message if the topology is invalid.

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
if (is_pureseqtm_installed()) {  
  fasta_filename <- get_example_filename("1bhaA.fasta")  
  topology <- predict_topology(fasta_filename)  
  check_topology(topology)  
}
```

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| check_topology_str | Check if the topology string is valid. Will <i>stop</i> if not. |
|--------------------|-----------------------------------------------------------------|

### Description

Check if the topology string is valid. Will **stop** if not.

## Usage

```
check_topology_str(topology_str)
```

## Arguments

|              |                                                       |
|--------------|-------------------------------------------------------|
| topology_str | the topology as a string, for example 000000111100000 |
|--------------|-------------------------------------------------------|

## Value

Nothing. Will **stop** with a helpful error message if the topology is invalid.

**Author(s)**

Richèl J.C. Bilderbeek

## Examples

```
check_topology_str("0000000000000000000000000000111111111111111100000")
```

|                                     |                                                         |
|-------------------------------------|---------------------------------------------------------|
| convert_tmhmm_to_pureseqtm_topology | <i>Convert a TMHMM topology to a PureseqTM topology</i> |
|-------------------------------------|---------------------------------------------------------|

### Description

## Convert a TMHMM topology to a PureseqTM topology

## Usage

```
convert_tmhmm_to_pureseqtm_topology(tmhmm_topology)
```

## Arguments

tmhmm\_topology topology as used by TMHMM

## Value

a **tibble** with column names `name` and `topology`, as can be checked by `check_topology`

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
tmhmm_topo_filename <- system.file(
  "extdata", "UP000005640_9606_no_u.tmhmm", package = "pureseqtmr"
)
tmhmm_topology <- load_topology_file_as_tibble(tmhmm_topo_filename)
convert_tmhmm_to_pureseqtm_topology(tmhmm_topology)
```

---

count\_n\_tmhs

*Count the number of TMHs in a topology*

---

**Description**

Count the number of TMHs in a topology

**Usage**

```
count_n_tmhs(topology_strs)
```

**Arguments**

topology\_strs    the topologies as zero, one or more strings, for example `c("0", "1")`

**Examples**

```
count_n_tmhs("000000000000000000000000000000")
count_n_tmhs("000000000001111100000000000000")
count_n_tmhs(c("0", "1"))
```

---

create\_pureseqtm\_files

*Create the five PureseqTM output files, by running PureseqTM.*

---

**Description**

Create the five PureseqTM output files, by running PureseqTM.

**Usage**

```
create_pureseqtm_files(
  fasta_filename,
  folder_name = get_default_pureseqtm_folder(),
  temp_folder_name = tempfile(pattern = "pureseqt_")
)
```

**Arguments**

fasta\_filename path to a FASTA file

folder\_name superfolder of PureseqTM. The superfolder's name is /home/[user\_name]/.local/share by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

temp\_folder\_name path of a temporary folder. The folder does not need to exist. Files that are out in this folder are not automatically deleted, which is not a problem, as the default path given by [tempdir](#) is automatically cleaned by the operating system

**Value**

full path to the files created

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_pureseqtm_installed()) {
  fasta_filename <- get_example_filename("1bhaA.fasta")
  create_pureseqtm_files(fasta_filename)
}
```

---

```
create_pureseqtm_proteome_file
```

*Create the output file of a PureseqTM proteome run*

---

**Description**

Create the output file of a PureseqTM proteome run

**Usage**

```
create_pureseqtm_proteome_file(
  fasta_filename,
  topology_filename = tempfile(fileext = ".top"),
  folder_name = get_default_pureseqtm_folder()
)
```

**Arguments**

fasta\_filename path to a FASTA file

topology\_filename name of the file to save a protein's topology to

folder\_name superfolder of PureseqTM. The superfolder's name is /home/[user\_name]/.local/share by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

**Value**

the filename

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_pureseqtm_installed()) {  
  fasta_filename <- get_example_filename("1bhaA.fasta")  
  create_pureseqtm_proteome_file(fasta_filename)  
}
```

---

|                    |                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------|
| default_params_doc | <i>This function does nothing. It is intended to inherit its parameters' documentation.</i> |
|--------------------|---------------------------------------------------------------------------------------------|

---

**Description**

This function does nothing. It is intended to inherit its parameters' documentation.

**Usage**

```
default_params_doc(  
  download_url,  
  fasta_filename,  
  fasta_file_text,  
  folder_name,  
  protein_sequence,  
  protein_sequences,  
  pureseqtm_filename,  
  pureseqtm_proteome_text,  
  pureseqtm_result,  
  pureseqtm_url,  
  temp_fasta_filename,  
  temp_folder_name,  
  tmhmm_topology,  
  topology,  
  topology_filename,  
  topology_str,  
  topology_strs,  
  verbose  
)
```

**Arguments**

|                         |                                                                                                                                                                                                                                                               |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| download_url            | the URL to download PureseqTM from                                                                                                                                                                                                                            |
| fasta_filename          | path to a FASTA file                                                                                                                                                                                                                                          |
| fasta_file_text         | text of a FASTA file                                                                                                                                                                                                                                          |
| folder_name             | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a>                                                                                             |
| protein_sequence        | a protein sequence, with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL. Use <a href="#">check_protein_sequence</a> to check if a protein sequence is valid.                                                                                     |
| protein_sequences       | one or more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL                                                                                                                                                  |
| pureseqtm_filename      | filename to write the PureseqTM results to                                                                                                                                                                                                                    |
| pureseqtm_proteome_text | the output of a call to PureseqTM_proteome.sh                                                                                                                                                                                                                 |
| pureseqtm_result        | the result of a PureseqTM run                                                                                                                                                                                                                                 |
| pureseqtm_url           | URL of the PureseqTM git repository                                                                                                                                                                                                                           |
| temp_fasta_filename     | temporary FASTA filename, which will be deleted after usage                                                                                                                                                                                                   |
| temp_folder_name        | path of a temporary folder. The folder does not need to exist. Files that are out in this folder are not automatically deleted, which is not a problem, as the default path given by <a href="#">tempdir</a> is automatically cleaned by the operating system |
| tmhmm_topology          | topology as used by TMHMM                                                                                                                                                                                                                                     |
| topology                | the topology as a <a href="#">tibble</a> with the columns 'name' and 'topology', where the 'name' column holds all the proteins' names, and 'topology' contains the respective topologies as strings.                                                         |
| topology_filename       | name of the file to save a protein's topology to                                                                                                                                                                                                              |
| topology_str            | the topology as a string, for example 000000111100000                                                                                                                                                                                                         |
| topology_strs           | the topologies as zero, one or more strings, for example c("0", "1")                                                                                                                                                                                          |
| verbose                 | set to TRUE for more output                                                                                                                                                                                                                                   |

**Note**

This is an internal function, so it should be marked with @noRd. This is not done, as this will disallow all functions to find the documentation parameters

**Author(s)**

Richèl J.C. Bilderbeek

get\_default\_pureseqtm\_folder

*Get the path to the folder where this package installs PureseqTM by default*

---

### Description

Get the path to the folder where this package installs PureseqTM by default

### Usage

```
get_default_pureseqtm_folder()
```

### Value

the path to the folder where this package installs PureseqTM by default

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
get_default_pureseqtm_folder()
```

---

get\_example\_filename    *Get the full path to a PureseqTM example file.*

---

### Description

Get the full path to a PureseqTM example file. If the filename specified is not a PureseqTM example file, this function will [stop](#)

### Usage

```
get_example_filename(filename, folder_name = get_default_pureseqtm_folder())
```

### Arguments

|             |                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename    | name of the example file, without the path                                                                                                                        |
| folder_name | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |

### Value

the full path to a PureseqTM example file

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

use [get\\_example\\_filenames](#) to get all PureseqTM example filenames

**Examples**

```
if (is_pureseqtm_installed()) {  
  get_example_filename("1bhaA.fasta")  
}
```

---

get\_example\_filenames *Get the full path to all PureseqTM example files*

---

**Description**

Get the full path to all PureseqTM example files

**Usage**

```
get_example_filenames(folder_name = get_default_pureseqtm_folder())
```

**Arguments**

folder\_name      superfolder of PureseqTM. The superfolder's name is /home/[user\_name]/.local/share by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

**Value**

a character vector with all PureseqTM example files

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

use [get\\_example\\_filename](#) to get the full path to a PureseqTM example file

**Examples**

```
if (is_pureseqtm_installed()) {  
  get_example_filenames()  
}
```

---

|                   |                                                 |
|-------------------|-------------------------------------------------|
| get_pureseqtm_url | <i>Get the URL of the PureseqTM source code</i> |
|-------------------|-------------------------------------------------|

---

**Description**

Get the URL of the PureseqTM source code

**Usage**

```
get_pureseqtm_url()
```

**Value**

a URL as a character vector of one element

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_pureseqtm_url()
```

---

|                       |                                  |
|-----------------------|----------------------------------|
| get_pureseqtm_version | <i>Get the PureseqTM version</i> |
|-----------------------|----------------------------------|

---

**Description**

Get the PureseqTM version

**Usage**

```
get_pureseqtm_version(folder_name = get_default_pureseqtm_folder())
```

**Arguments**

|             |                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| folder_name | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

a version number as a character vector of one element, for example v0.10

**Author(s)**

Richèl J.C. Bilderbeek

## Examples

```
if (is_pureseqtm_installed()) {  
  get_pureseqtm_version()  
}
```

---

|                   |                                            |
|-------------------|--------------------------------------------|
| install_pureseqtm | <i>Install PureseqTM to a local folder</i> |
|-------------------|--------------------------------------------|

---

## Description

Install PureseqTM to a local folder

## Usage

```
install_pureseqtm(  
  folder_name = get_default_pureseqtm_folder(),  
  pureseqtm_url = get_pureseqtm_url()  
)
```

## Arguments

|               |                                                                                                                                                                   |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| folder_name   | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
| pureseqtm_url | URL of the PureseqTM git repository                                                                                                                               |

## Value

Nothing.

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
## Not run:  
  install_pureseqtm()  
  
## End(Not run)
```

---

|                |                                                  |
|----------------|--------------------------------------------------|
| is_on_appveyor | <i>Determines if the environment is AppVeyor</i> |
|----------------|--------------------------------------------------|

---

**Description**

Determines if the environment is AppVeyor

**Usage**

```
is_on_appveyor()
```

**Value**

**TRUE** if run on AppVeyor, **FALSE** otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_on_appveyor()) {  
  message("Running on AppVeyor")  
}
```

---

|          |                                                                          |
|----------|--------------------------------------------------------------------------|
| is_on_ci | <i>Determines if the environment is a continuous integration service</i> |
|----------|--------------------------------------------------------------------------|

---

**Description**

Determines if the environment is a continuous integration service

**Usage**

```
is_on_ci()
```

**Value**

**TRUE** if run on AppVeyor or Travis CI, **FALSE** otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_on_ci()) {  
  message("Running on a continuous integration service")  
}
```

---

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| is_on_github_actions | <i>Determines if the environment is GitHub Actions</i> |
|----------------------|--------------------------------------------------------|

---

**Description**

Determines if the environment is GitHub Actions

**Usage**

```
is_on_github_actions()
```

**Value**

**TRUE** if run on GitHub Actions, **FALSE** otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_on_github_actions()) {  
  message("Running on GitHub Actions")  
}
```

---

|              |                                                   |
|--------------|---------------------------------------------------|
| is_on_travis | <i>Determines if the environment is Travis CI</i> |
|--------------|---------------------------------------------------|

---

**Description**

Determines if the environment is Travis CI

**Usage**

```
is_on_travis()
```

**Value**

**TRUE** if run on Travis CI, **FALSE** otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_on_ci()) {  
  message("Running on Travis CI")  
}
```

---

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| is_protein_name_line | <i>Is the line of text the name of a protein, as used within a FASTA filename?</i> |
|----------------------|------------------------------------------------------------------------------------|

---

**Description**

Is the line of text the name of a protein, as used within a FASTA filename?

**Usage**

```
is_protein_name_line(line)
```

**Arguments**

|      |                                    |
|------|------------------------------------|
| line | line of text from a FASTA filename |
|------|------------------------------------|

**Value**

**TRUE** if the line can be the name of a protein in a FASTA file

**Author(s)**

Rich  l J.C. Bilderbeek

**Examples**

```
is_protein_name_line(">5H2A_CRIGR")
```

---

|                        |                                                  |
|------------------------|--------------------------------------------------|
| is_pureseqtm_installed | <i>Measure if PureseqTM is installed locally</i> |
|------------------------|--------------------------------------------------|

---

**Description**

Measure if PureseqTM is installed locally

**Usage**

```
is_pureseqtm_installed(folder_name = get_default_pureseqtm_folder())
```

**Arguments**

|             |                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| folder_name | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

[TRUE](#) if PureseqTM is installed locally, [FALSE](#) otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
is_pureseqtm_installed()
```

---

|        |                                                                                     |
|--------|-------------------------------------------------------------------------------------|
| is_tmh | <i>Determine if the protein sequence contains at least one transmembrane helix.</i> |
|--------|-------------------------------------------------------------------------------------|

---

**Description**

Determine if the protein sequence contains at least one transmembrane helix.

**Usage**

```
is_tmh(protein_sequence, folder_name = get_default_pureseqtm_folder())
```

**Arguments**

|                  |                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| protein_sequence | a protein sequence, with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL. Use <a href="#">check_protein_sequence</a> to check if a protein sequence is valid. |
| folder_name      | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a>         |

**Value**

[TRUE](#) if the protein sequence contains at least one transmembrane helix

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_pureseqtm_installed()) {
  # This sequence is a TMH
  is_tmh("QKNWSALLTAVVIILTIAGNILVIMAVSLEKKLQATNYFLM")

  # This sequence is not a TMH
  is_tmh("VVIILTIRGNILVIMAVSLE")
}
```

---

|                  |                                                                           |
|------------------|---------------------------------------------------------------------------|
| is_topology_line | <i>Is the line of text the topology, as used within a FASTA filename?</i> |
|------------------|---------------------------------------------------------------------------|

---

**Description**

Is the line of text the topology, as used within a FASTA filename? In this context, a topology is a string of zeroes and ones, in which a one denotes that that amino acid is within the membrane.

**Usage**

```
is_topology_line(line)
```

**Arguments**

|      |                                    |
|------|------------------------------------|
| line | line of text from a FASTA filename |
|------|------------------------------------|

**Value**

**TRUE** if the line can be the text of a topology in a FASTA file.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
# This is a valid topology
is_topology_line("000010101011")

# This is an invalid topology
is_topology_line("invalid")
```

---

|                           |                                                   |
|---------------------------|---------------------------------------------------|
| is_valid_protein_sequence | <i>Determine if this a valid protein sequence</i> |
|---------------------------|---------------------------------------------------|

---

**Description**

Determine if this is a valid protein sequence, as can be used in topology prediction

**Usage**

```
is_valid_protein_sequence(protein_sequence, verbose = FALSE)
```

**Arguments**

- protein\_sequence      a protein sequence, with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL. Use [check\\_protein\\_sequence](#) to check if a protein sequence is valid.
- verbose                set to TRUE for more output

**Value**

TRUE if the protein sequence is valid

---

load\_fasta\_file\_as\_tibble

*Parse a FASTA file to a table with a name and sequence column*

---

**Description**

Parse a FASTA file to a table with a name and sequence column

**Usage**

```
load_fasta_file_as_tibble(fasta_filename)
```

**Arguments**

fasta\_filename    path to a FASTA file

**Value**

a [tibble](#) with a name and sequence column

**See Also**

use [load\\_fasta\\_file\\_as\\_tibble\\_cpp](#) to directly call the C++ function that does the actual work. Use [load\\_fasta\\_file\\_as\\_tibble\\_r](#) to call the (approx ten thousand times slower) R function

---

`load_fasta_file_as_tibble_cpp`*Parse a FASTA file to a table with a name and sequence column*

---

**Description**

Parse a FASTA file to a table with a name and sequence column

**Usage**

```
load_fasta_file_as_tibble_cpp(fasta_filename)
```

**Arguments**

`fasta_filename` path to a FASTA file

**Value**

a [tibble](#) with a name and sequence column

---

`load_fasta_file_as_tibble_cpp_raw`*Use Rcpp to load a FASTA file*

---

**Description**

Use Rcpp to load a FASTA file

**Usage**

```
load_fasta_file_as_tibble_cpp_raw(fasta_filename)
```

**Arguments**

`fasta_filename` FASTA filename

**Value**

a list with two character vectors, named 'name' and 'sequence'

---

`load_fasta_file_as_tibble_r`*Parse a FASTA file to a table with a name and sequence column*

---

**Description**

Parse a FASTA file to a table with a name and sequence column

**Usage**

```
load_fasta_file_as_tibble_r(fasta_filename)
```

**Arguments**

`fasta_filename` path to a FASTA file

**Value**

a [tibble](#) with a name and sequence column

---

`load_topology_file_as_tibble`*Parse a topology (.topo) file to a table with a name and topology column*

---

**Description**

Parse a topology (.topo) file to a table with a name and topology column

**Usage**

```
load_topology_file_as_tibble(topology_filename)
```

**Arguments**

`topology_filename`  
name of the file to save a protein's topology to

**Value**

a [tibble](#) with a name and topology column, as can be checked by [check\\_topology](#)

**Examples**

```
topology_filename <- system.file(  
  "extdata", "100507436.topo", package = "pureseqtmr"  
)  
load_topology_file_as_tibble(topology_filename)
```

---

mock\_predict\_topologies\_from\_sequences

*Do a mock prediction directly on a protein sequence, as can be useful in testing Use [predict\\_topologies\\_from\\_sequences](#) for doing a real prediction.*

---

## Description

Do a mock prediction directly on a protein sequence, as can be useful in testing Use [predict\\_topologies\\_from\\_sequences](#) for doing a real prediction.

## Usage

```
mock_predict_topologies_from_sequences(protein_sequences)
```

## Arguments

protein\_sequences

one or more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL

## Value

a topology as a string of zeroes and ones, where a one denotes that the corresponding amino acid is located within the membrane.

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
protein_sequence <- paste0(
  "QEKNWSALLTAVVIILTIAGNILVIMAVSLEKKLQATNYFLM",
  "SLAIADMLLGFLVMPVSMILTILYGYRWP"
)
mock_predict_topologies_from_sequences(protein_sequence)
```

---

mock\_predict\_topology *Do a mock prediction of the topology of proteins*

---

**Description**

Uses [predict\\_topology](#) for doing a real prediction

**Usage**

```
mock_predict_topology(fasta_filename)
```

**Arguments**

fasta\_filename path to a FASTA file

**Value**

a [tibble](#) with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains all respective topologies.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
fasta_filename <- tempfile()
save_tibble_as_fasta_file(
  t = tibble::tibble(
    name = c("A", "B"),
    sequence = c("FAMILYVW", "VWFAMILY")
  ),
  fasta_filename = fasta_filename
)
mock_predict_topology(fasta_filename)
```

---

parse\_pureseqtm\_proteome\_text

*Parse the output of a call to PureseqTM\_proteome.sh*

---

**Description**

Parse the output of a call to PureseqTM\_proteome.sh

**Usage**

```
parse_pureseqtm_proteome_text(pureseqtm_proteome_text)
```

**Arguments**

pureseqtm\_proteome\_text  
the output of a call to PureseqTM\_proteome.sh

---

|               |                          |
|---------------|--------------------------|
| plot_topology | <i>Plot the topology</i> |
|---------------|--------------------------|

---

**Description**

Plot the topology

**Usage**

```
plot_topology(topology)
```

**Arguments**

topology      the topology as a [tibble](#) with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains the respective topologies as strings.

**Value**

a [ggplot](#) that displays the topology of one or more proteins

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_pureseqtm_installed() && is_on_ci()) {  
  fasta_filename <- get_example_filename("test_proteome.fasta")  
  topology <- predict_topology(fasta_filename)  
  plot_topology(topology)  
}
```

---

`predict_topologies_from_sequences`*Run PureseqTM directly on a protein sequence*

---

**Description**

Run PureseqTM directly on a protein sequence

**Usage**

```
predict_topologies_from_sequences(  
  protein_sequences,  
  folder_name = get_default_pureseqtm_folder(),  
  temp_fasta_filename = tempfile(fileext = ".fasta")  
)
```

**Arguments**

|                                  |                                                                                                                                                                   |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>protein_sequences</code>   | one or more protein sequence, each sequence with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL                                                      |
| <code>folder_name</code>         | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
| <code>temp_fasta_filename</code> | temporary FASTA filename, which will be deleted after usage                                                                                                       |

**Value**

a topology as a string of zeroes and ones, where a one denotes that the corresponding amino acid is located within the membrane.

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

use [mock\\_predict\\_topologies\\_from\\_sequences](#) to mock the prediction of protein sequences, as can be useful in testing

**Examples**

```
if (is_pureseqtm_installed()) {  
  protein_sequence <- paste0(  
    "QKKNWSALLTAVVIILTIAGNILVIMAVSLEKKLQATNYFLM",  
    "SLAIADMLLGFLVMPVSMILTILYGYRWP"  
  )  
  predict_topology_from_sequence(protein_sequence)  
}
```

---

|                  |                                                   |
|------------------|---------------------------------------------------|
| predict_topology | <i>Predict the topology of proteins from file</i> |
|------------------|---------------------------------------------------|

---

## Description

Predict the topology of zero, one or more proteins, of which the names and sequences are stored in the FASTA format

## Usage

```
predict_topology(
  fasta_filename,
  folder_name = get_default_pureseqtm_folder(),
  topology_filename = tempfile(fileext = ".top")
)
```

## Arguments

|                   |                                                                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fasta_filename    | path to a FASTA file                                                                                                                                              |
| folder_name       | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
| topology_filename | name of the file to save a protein's topology to                                                                                                                  |

## Value

a [tibble](#) with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains all respective topologies.

## Note

unlike PureseqTM, the topologies predicted are returned in the same order as the original sequences. A bugreport is posted at the PureseqTM GitHub repository at [https://github.com/PureseqTM/PureseqTM\\_Package/issues/11](https://github.com/PureseqTM/PureseqTM_Package/issues/11)

## Author(s)

Richèl J.C. Bilderbeek

## See Also

use [mock\\_predict\\_topology](#) to do a mock prediction, as can be useful in testing

## Examples

```
if (is_pureseqtm_installed()) {
  fasta_filename <- get_example_filename("1bhaA.fasta")
  predict_topology(fasta_filename)
}
```

---

`predict_topology_from_sequence`*Run PureseqTM directly on a protein sequence*

---

## Description

Will [stop](#) if the protein sequence is shorter than three amino acids.

## Usage

```
predict_topology_from_sequence(  
  protein_sequence,  
  folder_name = get_default_pureseqtm_folder(),  
  temp_fasta_filename = tempfile(fileext = ".fasta")  
)
```

## Arguments

|                                  |                                                                                                                                                                   |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>protein_sequence</code>    | a protein sequence, with the amino acids as capitals, for example MEILCEDNTSLSSIPNSL                                                                              |
| <code>folder_name</code>         | superfolder of PureseqTM. The superfolder's name is /home/[user_name]/.local/share by default, as can be obtained by <a href="#">get_default_pureseqtm_folder</a> |
| <code>temp_fasta_filename</code> | temporary FASTA filename, which will be deleted after usage                                                                                                       |

## Value

a topology as a string of zeroes and ones, where a one denotes that the corresponding amino acid is located within the membrane.

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
if (is_pureseqtm_installed()) {  
  protein_sequence <- paste0(  
    "QKKNWSALLTAVVIILTIAGNILVIMAVSLEKKLQNATNYFLM",  
    "SLAIADMLLGFLVMPVSMILTILYGYRWP"  
  )  
  predict_topology_from_sequence(protein_sequence)  
}
```

---

`pureseqtmr`

*`pureseqtmr`: estimate the topology of membrane proteins*

---

## Description

Proteins reside in either the cell plasma or in the cell membrane. A membrane protein goes through the membrane at least once. There are multiple ways to span this hydrophobic layer. One common structure is the transmembrane (alpha) helix (TMH). Given the amino acid sequence of a membrane protein, this package predicts which parts of the protein are TMHs

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
if (is_pureseqtm_installed()) {
  # Obtain an example filename
  fasta_filename <- get_example_filename("1bhaA.fasta")

  # Get the topology as a tibble
  topology <- predict_topology(fasta_filename)

  # show the topology
  plot_topology(topology)
}
```

---

`pureseqtmr_report`

*Create a [pureseqtmr](#) report, to be used when reporting bugs*

---

## Description

Create a [pureseqtmr](#) report, to be used when reporting bugs

## Usage

```
pureseqtmr_report(folder_name = get_default_pureseqtm_folder())
```

## Arguments

`folder_name`      superfolder of PureseqTM. The superfolder's name is `/home/[user_name]/.local/share` by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

## Value

Nothing.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
pureseqtmr_report()
```

---

```
run_pureseqtm_proteome
```

*Run PureseqTM on a proteome*

---

**Description**

Run PureseqTM on a proteome

**Usage**

```
run_pureseqtm_proteome(  
  fasta_filename,  
  folder_name = get_default_pureseqtm_folder(),  
  topology_filename = tempfile(fileext = ".top")  
)
```

**Arguments**

`fasta_filename` path to a FASTA file

`folder_name` superfolder of PureseqTM. The superfolder's name is `/home/[user_name]/.local/share` by default, as can be obtained by [get\\_default\\_pureseqtm\\_folder](#)

`topology_filename` name of the file to save a protein's topology to

**Value**

the topology of the proteome, using the same output as PureseqTM. Use [predict\\_topology](#) to get the topology as a [tibble](#)

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

- Use [predict\\_topology](#) to predict the topology of a proteome
- Use [create\\_pureseqtm\\_files](#) to only create the PureseqTM output files

**Examples**

```
if (is_pureseqtm_installed()) {
  fasta_filename <- get_example_filename("1bhaA.fasta")
  run_pureseqtm_proteome(fasta_filename)
}
```

---

```
save_tibble_as_fasta_file
```

*Save the first two columns of a tibble as a FASTA file*

---

**Description**

Save the first two columns of a tibble as a FASTA file

**Usage**

```
save_tibble_as_fasta_file(t, fasta_filename)
```

**Arguments**

t                      a [tibble](#)  
 fasta\_filename    path to a FASTA file

**Author(s)**

Richèl J.C. Bilderbeek

---

```
tally_tmhs
```

*Count the number of transmembrane helices in a topology*

---

**Description**

Count the number of transmembrane helices in a topology

**Usage**

```
tally_tmhs(topology)
```

**Arguments**

topology            the topology as a [tibble](#) with the columns 'name' and 'topology', where the 'name' column hold all the proteins' names, and 'topology' contains the respective topologies as strings.

**Value**

a [tibble](#) with the number of TMHs per protein

**Examples**

```
if (is_pureseqtm_installed()) {  
  tally_tmhs(  
    predict_topology(  
      get_example_filename("1bhaA.fasta")  
    )  
  )  
}
```

---

|                     |                            |
|---------------------|----------------------------|
| uninstall_pureseqtm | <i>Uninstall PureseqTM</i> |
|---------------------|----------------------------|

---

**Description**

Uninstall PureseqTM

**Usage**

```
uninstall_pureseqtm(folder_name = get_default_pureseqtm_folder())
```

**Arguments**

|             |                                                                                                                                              |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| folder_name | name of the folder where the PureseqTM files are installed. The name of the PureseqTM binary file will be at [folder_name]/PureseqTM_Package |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

Nothing.

**Author(s)**

Richèl J.C. Bilderbeek

# Index

are\_tmhs, [3](#)  
are\_valid\_protein\_sequences, [4](#)  
  
calc\_distance\_to\_tmh\_center\_from\_topology, [4](#)  
calc\_distance\_to\_tmh\_center\_from\_topology\_str, [5](#)  
calc\_distance\_to\_tmh\_center\_from\_topology\_str\_cpp\_stl, [26](#), [29](#)  
[5](#)  
check\_protein\_sequence, [6](#), [6](#), [13](#), [21](#), [23](#)  
check\_protein\_sequences, [6](#)  
check\_pureseqtm\_installation, [7](#)  
check\_topology, [8](#), [9](#), [25](#)  
check\_topology\_str, [9](#)  
convert\_tmhmm\_to\_pureseqtm\_topology, [9](#)  
count\_n\_tmhs, [10](#)  
create\_pureseqtm\_files, [10](#), [33](#)  
create\_pureseqtm\_proteome\_file, [11](#)  
  
default\_params\_doc, [12](#)  
  
FALSE, [18](#), [19](#), [21](#)  
  
get\_default\_pureseqtm\_folder, [3](#), [7](#), [11](#),  
[13](#), [14](#), [14](#), [15–17](#), [20](#), [21](#), [29–33](#)  
get\_example\_filename, [14](#), [15](#)  
get\_example\_filenames, [15](#), [15](#)  
get\_pureseqtm\_url, [16](#)  
get\_pureseqtm\_version, [16](#)  
ggplot, [28](#)  
  
install\_pureseqtm, [17](#)  
is\_on\_appveyor, [18](#)  
is\_on\_ci, [18](#)  
is\_on\_github\_actions, [19](#)  
is\_on\_travis, [19](#)  
is\_protein\_name\_line, [20](#)  
is\_pureseqtm\_installed, [20](#)  
is\_tmh, [21](#)  
is\_topology\_line, [22](#)  
is\_valid\_protein\_sequence, [22](#)  
  
load\_fasta\_file\_as\_tibble, [23](#)  
load\_fasta\_file\_as\_tibble\_cpp, [23](#), [24](#)  
load\_fasta\_file\_as\_tibble\_cpp\_raw, [24](#)  
load\_fasta\_file\_as\_tibble\_r, [23](#), [25](#)  
load\_topology\_file\_as\_tibble, [25](#)  
  
mock\_predict\_topologies\_from\_sequences,  
mock\_predict\_topology, [27](#), [30](#)  
  
parse\_pureseqtm\_proteome\_text, [27](#)  
plot\_topology, [28](#)  
predict\_topologies\_from\_sequences, [26](#),  
[29](#)  
predict\_topology, [8](#), [27](#), [30](#), [33](#)  
predict\_topology\_from\_sequence, [31](#)  
pureseqtmr, [32](#), [32](#)  
pureseqtmr\_report, [32](#)  
  
run\_pureseqtm\_proteome, [33](#)  
  
save\_tibble\_as\_fasta\_file, [34](#)  
stop, [6–9](#), [14](#), [31](#)  
  
tally\_tmhs, [34](#)  
tempdir, [11](#), [13](#)  
tibble, [4](#), [5](#), [8](#), [9](#), [13](#), [23–25](#), [27](#), [28](#), [30](#), [33–35](#)  
TRUE, [3](#), [18–22](#)  
  
uninstall\_pureseqtm, [35](#)