Package ‘powerHaDeX’

January 24, 2022

Type Package

Title Efficient Simulation of HDX-MS Data and Tools for the Statistical Analysis

Version 1.0

Date 2022-01-21

Maintainer Krystyna Grzesiak <krygrz11@gmail.com>

Description Facilitates simulating and analyzing data coming from HDX-MS experiments along with the possibility of comparing the power of the tests verifying differences in the levels of deuterium uptake. The simulation of mass spectra is a fast version of <https://github.com/kanzy/HX-MS-Simulations>.

License GPL (>= 3)

Encoding UTF-8

Imports checkmate, data.table, expm, ggplot2, glmnet, lme4, lmerTest, methods, nlme, plyr, Rcpp (>= 1.0.3), signal,

Depends R (>= 3.5.0)

LinkingTo Rcpp

RoxygenNote 7.1.2

Suggests testthat, spelling, knitr, rmarkdown, covr

Language en-US

VignetteBuilder knitr

NeedsCompilation yes

Author Krystyna Grzesiak [cre, aut],
Mateusz Staniak [aut],
Michal Burdukiewicz [ctb] (<https://orcid.org/0000-0001-8926-582X>)

Repository CRAN

Date/Publication 2022-01-24 19:32:46 UTC

R topics documented:

`powerHDX-package` *A short title line describing what the package does*

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
## Not run:
   ## Optional simple examples of the most important functions
   ## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
```

---

add_column *Complete data frame with columns*

Description

This function adds column if does not exist and fill it with provided value.

Usage

```
add_column(data, col_name, value = NULL)
```

Arguments

data	a data frame of interest.
col_name	a character. Name of column that should be created if it does not exist
value	optional. A value to fill with.

---

calculate_hdx_power *Calculate power of statistical tests for HDX experiments*

Description

This function estimates power of statistical tests for HDX experiments.

Usage

```
calculate_hdx_power(
  deuteration_curves,
  tests,
  significance_level = 0.05,
  summarized = TRUE
)
```

Arguments

deuteration_curves
: list returned by the get_noisy_deuteration_curves

tests
: lists of tests to perform. Each test function should take two parameters - data (data_table containing replicated curves) and significance_level, and have particular output - data frame of variables: Test (name of a test which should be displayed in the final result), State_1, State_2 (biological states of interest), Test_statistic, P_value, Significant_difference (the same as p_value <= significance_level), Time (character, "continuous" or "categorical"), Transformation (character, transformation that is used for exposure), AIC, logLik. For example see test_houde.

significance_level
: significance level that will be used for testing. See tests

summarized
: logical. Indicates whether the power should be calculated. Default TRUE.

Value

list of data.tables with test result, optionally summarized with power.

See Also

test_houde, test_semiparametric, test_hdx_analyzer, test_memhdx_model

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 100,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 200,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                     n_replicates = 4,
```

```
                                            n_experiments = 2,
                                            compare_pairs = TRUE,
                                            reference = "all")
calculate_hdx_power(deut_curves_p_states,
                    tests = list(test_houde),
                    summarized = TRUE)
```

---

calculate_peptide_mass *Peptide mass*

Description

Calculate mass of undeuterated peptide

Usage

```
calculate_peptide_mass(sequence)
```

Arguments

sequence	character vector of amino acid sequence of a peptide

Details

Calculates peptide mass as a sum of amino acids' from sequence masses and H2O mass (1.007825 * 2 + 15.994915 = 18.01056).

Value

a single number denoting the mass of the undeuterated peptide.

---

create_experimental_file *Create experimental file*

Description

This function generates replications of mass spectra that are consistent with common experimental data files

Usage

```
create_experimental_file(
  peptides,
  times = c(0.167, 1, 5, 25, 1440),
  charge,
  n_replicates = 3,
  mass_deviations = 50,
  intensity_deviations = NULL,
  file_type = "DynamX"
)
```

Arguments

`peptides`
: a data frame of sequences (`sequence`), `Protein`, and `Start`, `End` and parameters except `times` that can be used for simulating mass spectra. See `simulate_theoretical_spectra` for more details about the additional parameters.

`times`
: a vector of times at which deuteration levels will be measured (seconds)

`charge`
: vector of charges of the peptide ion. If NULL, one value is sampled from vector 2:6. Default NULL.

`n_replicates`
: number of technical replicates to create

`mass_deviations`
: mass deviation in parts per million. Either a single number (then the error at each time point will be the same) or a vector of the same length as number of unique time points in the experiment. The error will be sampled from normal distribution with standard deviation equal to

$$mass_deviations * undeuterated_mass/1e6$$

Default to 50.

`intensity_deviations`
: optional, standard deviations of random noise that will be added to intensities. Either a single number (then the error at each time point will be the same) or a vector of the same length as number of unique time points in the experiment. The error will be sampled from normal distribution with these standard deviations.Default `NULL`.

`file_type`
: the type of file. Default to "DynamX".

Value

data table. The table of HDX-MS results consistent with ‘file_type‘ format.

Examples

```
peptides <- data.frame(sequence = c("FPTTKTY", "LVRKDLQN"),
                       protection_factor = c(10, 100))
create_experimental_file(peptides, charge = 1:3)
```

`fix_columns_names_types` *Standardize column names and types*

Description

Standardize column names and types

Usage

```
fix_columns_names_types(curves)
```

Arguments

`curves`	list of lists of data.tables

`get_deuteration_single_timepoint` *Calculates deuteration for given timepoint*

Description

Calculates deuteration for given timepoint

Usage

```
get_deuteration_single_timepoint(
  initial_matrix,
  time_sequence,
  hd_probs,
  dh_probs
)
```

Arguments

`initial_matrix`	A matrix
`time_sequence`	vector of exchange times
`hd_probs`	probabilities of transition HD
`dh_probs`	probabilities of transition DH

Value

a matrix denoting hydrogen-deuterium exchange for given timepoint.

`get_noisy_deuteration_curves`
Replicated deuterium uptake curves

Description

This function creates a list of lists of noisy deuteration curves based on theoretical spectra in order to imitate the data from the HDX experiments.

Usage

```
get_noisy_deuteration_curves(
  theoretical_spectra,
  compare_pairs = TRUE,
  reference = NA,
  n_replicates = 4,
  n_experiments = 100,
  mass_deviations = 50,
  intensity_deviations = NULL,
  per_run_deviations = NULL,
  relative = TRUE
)
```

Arguments

`theoretical_spectra`
a data table or a list of data tables of theoretical spectra created by the function `simulate_theoretical_spectra`.

`compare_pairs` if FALSE, all groups (defined by the protection factor) will be considered jointly. If TRUE (default), each protection factor will be considered together with the protection factor given by the 'reference' parameter.

`reference` protection factor that will be used for comparison to other protection factors in. The function accepts either `NA` (for comparing all protection factors), a number (for comparing with reference value of protection factor) or "all" (for pairwise comparisons of all the possible combinations). Default `NA`.

`n_replicates` number of technical replicates to create

`n_experiments` number of replicates of an experiment for power calculation.

`mass_deviations`
mass deviation in parts per million. Either a single number (then the error at each time point will be the same) or a vector of the same length as number of unique time points in the experiment. The error will be sampled from normal distribution with standard deviation equal to

$$mass_deviations * undeuterated_mass/1e6$$

Default to 50.

intensity_deviations
: optional, standard deviations of random noise that will be added to intensities. Either a single number (then the error at each time point will be the same) or a vector of the same length as number of unique time points in the experiment. The error will be sampled from normal distribution with these standard deviations.Default NULL.

per_run_deviations
: optional, standard deviations of random noise that will be added to deuteration curves. Either a single number (then the error at each time point will be the same) or a vector of the same length as number of unique time points in the experiment. The error will be sampled from normal distribution with these standard deviations. Default NULL.

relative
: logical, if TRUE (default), each deuteration curve will start at 0 (relative mass will be returned). Default TRUE.

Value

a list (for paired states when compare_pairs is TRUE) of lists (repetitions of experiment for power calculations) of data tables of the variables:

- Sequence - provided amino acid sequence

- Rep - technical replication

- State - provided protection factor (the theoretical - in practice unknown - state of the protein)

- Exposure - exposure time

- Mass - mass or deuterium uptake when relative is TRUE.

- Charge - charge

- Experimental_state - the biological state (from the viewpoint of the experimenter) provided in the case when compare_pairs is TRUE.

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 100,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 200,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
```

```
                                                      use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                      n_replicates = 4,
                                                      n_experiments = 2,
                                                      compare_pairs = TRUE,
                                                      reference = "all")
```

get_spectra_list   *Get a list of spectra*

Description

Create a list of data tables of spectra for all states jointly or paired states.

Usage

```
get_spectra_list(theoretical_spectra, compare_pairs = FALSE, reference = NA)
```

Arguments

`theoretical_spectra`
: a data table or a list of data tables of theoretical spectra created by the function `simulate_theoretical_spectra`.

`compare_pairs`
: if FALSE, all groups (defined by the protection factor) will be considered jointly. If TRUE (default), each protection factor will be considered together with the protection factor given by the ‘reference‘ parameter.

`reference`
: protection factor that will be used for comparison to other protection factors in. The function accepts either `NA` (for comparing all protection factors), a number (for comparing with reference value of protection factor) or "all" (for pairwise comparisons of all the possible combinations). Default `NA`.

Details

If the parameter `compare_pairs` is `FALSE` then all the provided protection factors will be considered jointly. If `compare_pairs` is `TRUE`, then the parameter `reference` is necessary (a single number or `"all"`). Then the data is split via the supplementary function `get_paired_spectra` into data tables of spectra with paired biological states (the reference protection factor and the protection factor of interest if provided, or all the possible pairs if `reference` equals "all").

Value

list of data.tables containing spectra - for paired states or all states.

plot_spectra *Draw mass spectra*

Description

Graphical visualization of mass spectra obtained using the function simulate_theoretical_spectra.

Usage

```
plot_spectra(
  spectra,
  time_points = unique(spectra[["Exposure"]]),
  charges = unique(spectra[["Charge"]]),
  control_time = FALSE,
  ...
)
```

Arguments

spectra	data table. Result of simulate_theoretical_spectra.
time_points	vector of values of exposure times to be displayed on the plot. Default unique(spectra[["Exposure"]])
charges	vector of charges to be displayed on the plot. Default unique(spectra[["Charge"]]).
control_time	logical. Indicates whether the spectrum at the control time (conventionally equal to 0) should be drawn.
...	additional arguments passing to the theme.

Details

This function draws mass spectra from data obtained via simulate_theoretical_spectra.

Value

ggplot object

Examples

```
theo_spectra <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                             charge = c(3, 5),
                                             protection_factor = 100,
                                             times = c(0.167, 5),
                                             pH = 7.5,
                                             temperature = 15,
                                             n_molecules = 500,
                                             time_step_const = 1,
                                             use_markov = TRUE)
```

```
plot_spectra(theo_spectra)
```

---

powerHaDeX *powerHaDeX*

Description

The R-package powerHaDeX for simulating and analyzing data coming from HDX-MS experiments along with the possibility of comparing the power of the tests verifying differences in deuteration levels.

Author(s)

Michal Burdukiewicz, Krystyna Grzesiak, Mateusz Staniak

---

prepare_input_peptides *Prepare input for* `create_experimental_file`

Description

Supplementary function providing appropriate input.

Usage

```
prepare_input_peptides(peptides)
```

Arguments

`peptides`	a data frame of parameters for which `simulate_theoretical_spectra` will be executed.

Value

a data frame being a proper input for `create_experimental_file`.

simulate_theoretical_spectra

Simulate theoretical spectra of a deuterated peptide over time

Description

Simulate theoretical spectra of a deuterated peptide over time

Usage

```
simulate_theoretical_spectra(
  sequence,
  charge = NULL,
  protection_factor = 1,
  times = c(60, 600),
  pH = 7.5,
  temperature = 15,
  n_molecules = 100,
  time_step_const = 1,
  if_corr = FALSE,
  min_probability = 1e-04,
  use_markov = TRUE
)
```

Arguments

`sequence` amino acid sequence of a peptide as a single string

`charge` vector of charges of the peptide ion. If NULL, one value is sampled from vector 2:6. Default NULL.

`protection_factor` protection factor. If a single number of provided, same protection factor will be assumed for each amide. Default value: 1 (indicates that the exchange rate is equal to the intristic exchange rate)

`times` a vector of times at which deuteration levels will be measured (seconds)

`pH` pH of the reaction. Default to 7.5.

`temperature` temperature of the reaction (Celsius)

`n_molecules` number of peptide molecules. Default to 100.

`time_step_const` time step constant. Default value: 1. Value that indicates the length of the time step of the simulation. The bigger the time step, the fewer time points are simulated (the fewer iterations in case of Zhong-Yuan Kan's approach).

`if_corr` logical. PH correction indicator. Default value `FALSE`. The value of pH is equal to pD. If there is correction, the pD = pH + 0.4. (Conelly et al 1993)

`min_probability` smallest isotopic probability to consider use_markov — logical. If TRUE algorithm basing on Markov chain will be used. If FALSE simulation provided by Zhong-Yuan Kan will be executed. Default to TRUE, as it fastens the calculation

Details

To the results calculated by get_iso_probs_deut is added a minimal exchange control - for time point 0 (directly after adding a buffer). The m/z values are obtained as a ratio of the peptide_mass magnified by proton mass and the peptide charge. The distribution of undeuterated peptide is the intensities vector.

Value

a data table of variables:

- Exposure - time point of a measurement,

- Mz - mass-to-charge ratio,

- Intensity - isotopic probabilities larger than min_probability(the smaller ones are zeroes)

and the variables provided by user

- Sequence,

- PF,

- Charge,

- PH.

See Also

The algorithm that is used to simulate theoretical spectra is based on Zhong-Yuan Kan's implementation in Matlab. The original version of codes is located in the repository https://github.com/kanzy/HX-MS-Simulations (as at 29.06.2020). In the powerHaDeX package can be found the Kan's algorithm re-implemented in R (using Rcpp) and the accelerated implementation (that uses Markov chains' properties). Moreover, the package powerHaDeX allows the user to simulate spectra for more than one exposure time for both (Rcpp and Markov) approaches.

Examples

```
simulate_theoretical_spectra(sequence = "LVRKDLQN",
                             charge = c(3, 5),
                             protection_factor = 100,
                             times = c(0.167, 5),
                             pH = 7.5,
                             temperature = 15,
                             n_molecules = 500,
                             time_step_const = 1,
                             use_markov = TRUE)
```

test_hadex_data — *Apply tests for HaDeX data*

Description

This function converts the data from HaDeX in order to make it compatible with the input of test functions and perform the testing procedures of provided tests.

Usage

```
test_hadex_data(
  dat,
  states = unique(dat[["State"]])[1:2],
  tests = list(test_houde)
)
```

Arguments

`dat`	data.table. The data of hdx_data class from the HaDeX package.
`states`	a character vector containing two states from provided ‘dat‘ that should be tested. By default the first two states (if exist) from ‘dat‘ are chosen.
`tests`	a list of testing functions. In the ‘powerHaDeX‘ package the following tests are implemented: - `test_houde`, - `test_hdx_analyzer`, - `test_memhdx_model`, -`test_semiparametric`.

Value

This function returns a data table of variables:

- `Test` - name of test,

- `State_1`, `State_2` - tested states from `states`,

- `Significant_difference` - `TRUE` or `FALSE`, indicating whether the null hypothesis is rejected

- `Sequence` - amino acid sequence that was tested test_hdx_analyzer *HDX-Analyzer model*

Description

This function performs the test based on the simplest linear models for deuteration curves containing time, state of the protein and the interaction term. Its input and output are compatible with the function calculate_hdx_power.

Usage

```
test_hdx_analyzer(data, significance_level = 0.05)
```

Arguments

data	data.table with deuteration curves
significance_level	significance level for tests

Value

This function returns a data table compatible with the function calculate_hdx_power.

References

Liu, Sanmin et al. (2011). “HDX-analyzer: a novel package for statistical analysis of protein structure dynamics”. In:BMC bioinformatics12.1, pp. 1–10.

See Also

Other tests:

- test_houde

- test_memhdx_model

-test_semiparametric

Or calculate_hdx_power for estimation of power of tests for differences in deuteration levels.

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                     charge = c(3, 5),
                                                     protection_factor = 100,
                                                     times = c(0.167, 5),
                                                     pH = 7.5,
                                                     temperature = 15,
                                                     n_molecules = 500,
                                                     time_step_const = 1,
                                                     use_markov = TRUE)
```

```
theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 200,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                     n_replicates = 4,
                                                     n_experiments = 1,
                                                     reference = 100)[[1]][[1]]
test_hdx_analyzer(deut_curves_p_states)
```

`test_houde` *Houde's test for deuteration curves*

Description

This function performs Damian Houde's confidence intervals test for differences in deuteration levels. Its input and output are compatible with the function `calculate_hdx_power`.

Usage

```
test_houde(data, significance_level = 0.05)
```

Arguments

`data`	data.table with deuteration curves
`significance_level`	significance level for tests

Value

This function returns a data table compatible with the function `calculate_hdx_power`.

References

Houde, Damian, Steven A Berkowitz, and John R Engen (2011). "The utility of hydrogen/deuterium exchange mass spectrometry in biopharmaceutical comparabilitystudies". In:Journal of pharmaceutical sciences100.6, pp. 2071–2086.

See Also

Other tests:

- test_hdx_analyzer
- test_memhdx_model
- test_semiparametric

Or calculate_hdx_power for estimation of power of tests for differences in deuteration levels.

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 100,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)
theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 200,
                                                    times = c(0.167, 5),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                     n_replicates = 4,
                                                     n_experiments = 1,
                                                     reference = 100)[[1]][[1]]
test_houde(deut_curves_p_states)
```

---

test_memhdx_model *MEMHDX model*

Description

This function performs the test based on a linear mixed effects model used in MEMHDX tools. Its input and output are compatible with the function calculate_hdx_power.

Usage

```
test_memhdx_model(data, significance_level = 0.05)
```

Arguments

`data` data.table with deuteration curves

`significance_level` significance level for tests

Value

This function returns a data table compatible with the function `calculate_hdx_power`.

References

Hourdel, Véronique et al. (July 2016). "MEMHDX: an interactive tool to expedite the statistical validation and visualization of large HDX-MS data sets". In:Bioinformatics32.22, pp. 3413–3419.issn: 1367-4803.

See Also

Other tests:

- `test_houde`
- `test_hdx_analyzer`
- `test_semiparametric`

Or `calculate_hdx_power` for estimation of power of tests for differences in deuteration levels.

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                     charge = c(3, 5),
                                                     protection_factor = 100,
                                                     times = c(0.167, 5),
                                                     pH = 7.5,
                                                     temperature = 15,
                                                     n_molecules = 500,
                                                     time_step_const = 1,
                                                     use_markov = TRUE)
theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                     charge = c(3, 5),
                                                     protection_factor = 200,
                                                     times = c(0.167, 5),
                                                     pH = 7.5,
                                                     temperature = 15,
                                                     n_molecules = 500,
                                                     time_step_const = 1,
                                                     use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                      n_replicates = 4,
                                                      n_experiments = 1,
                                                      reference = 100)[[1]][[1]]
```

```
test_memhdx_model(deut_curves_p_states)
```

test_semiparametric — *Semiparametric test for differences in deuteration levels*

Description

This function performs the semiparametric test for differences in deuteration levels. Its input and output are compatible with the function calculate_hdx_power.

Usage

```
test_semiparametric(data, significance_level = 0.05)
```

Arguments

data	data.table with deuteration curves
significance_level	significance level for tests

Details

This function uses truncated_lines. The knots considered in the testing procedure are chosen using ridge regression.

Value

This function returns a data table compatible with the function calculate_hdx_power.

See Also

Other tests:

- test_houde

- test_hdx_analyzer

-test_memhdx_model

Or calculate_hdx_power for estimation of power of tests for differences in deuteration levels.

Examples

```
theo_spectra_pf_100 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 100,
                                                    times = c(0.167, 5, 10, 30),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
```

```
                                                    time_step_const = 1,
                                                    use_markov = TRUE)
theo_spectra_pf_200 <- simulate_theoretical_spectra(sequence = "LVRKDLQN",
                                                    charge = c(3, 5),
                                                    protection_factor = 200,
                                                    times = c(0.167, 5, 10, 30),
                                                    pH = 7.5,
                                                    temperature = 15,
                                                    n_molecules = 500,
                                                    time_step_const = 1,
                                                    use_markov = TRUE)

theo_spectra_two_states <- rbind(theo_spectra_pf_100, theo_spectra_pf_200)

deut_curves_p_states <- get_noisy_deuteration_curves(theo_spectra_two_states,
                                                     n_replicates = 4,
                                                     n_experiments = 1,
                                                     reference = 100)[[1]][[1]]
test_semiparametric(deut_curves_p_states)
```

Index