Introduction to the TPP package for analyzing Thermal Proteome Profiling data: 2D-TPP experiments

Dorothee Childs, Nils Kurzawa
European Molecular Biology Laboratory (EMBL), Heidelberg, Germany
dorothee.childs@embl.de

TPP version 3.4.0 (Last revision 2016-10-24)

Abstract

Thermal Proteome Profiling (TPP) combines the cellular thermal shift assay concept [1] with mass spectrometry based proteome-wide protein quantitation [2]. Thereby, drug-target interactions can be inferred from changes in the thermal stability of a protein upon drug binding, or upon downstream cellular regulatory events, in an unbiased manner.

The package TPP facilitates this process by providing executable workflows that conduct all necessary data analysis steps. Recent advances in the field have lead to the development of so called 2D Thermal Proteome Profiling (2D-TPP) experiments [3]. Recent advances in the field have lead to the development of so called 2D Thermal Proteome Profiling (2D-TPP) experiments [3]. Similar as for the TPP-TR and the TPP-CCR analysis, the function analyze2DTPP executes the whole workflow from data import through normalization and curve fitting to statistical analysis. Nevertheless, all of these steps can also be invoked separately by the user. The corresponding functions can be recognized by their suffix tpp2d.

Here, we first show how to start the whole analysis using analyze2DTPP. Afterwards, we demonstrate how to carry out single steps individually.

For details about the analysis of 1D TR- or CCR experiments [2, 4], please refer to the vignette TPP_introduction_1D.

Contents

1 Installation .................................................. 1
   1.1 Special note for Windows users ......................... 2

2 Analyzing 2D-TPP experiments .......................... 3
   2.1 Overview .................................................. 3
   2.2 Performing the analysis .................................. 3
   2.3 Quality control analyses ................................. 13
   2.4 Spline fits of treatment effects over temperature .... 14

1 Installation

To install the package, type the following commands into the R console

source("http://bioconductor.org/biocLite.R")
biocLite("TPP")

The installed package can be loaded by

library("TPP")
1.1 Special note for Windows users

The TPP package uses the openxlsx package to produce Excel output [5]. openxlsx requires a zip application to be installed on your system and to be included in the path. On Windows, such a zip application is not installed by default, but is available, for example, via Rtools. Without the zip application, you can still use the ‘TPP’ package and access its results via the dataframes produced by the main functions.
2 Analyzing 2D-TPP experiments

2.1 Overview

Before you can start your analysis, you need to specify information about your experiments:

The mandatory information comprises a unique experiment name, as well as the isobaric labels and corresponding temperature values for each experiment. The package retrieves this information from a configuration table that you need to specify before starting the analysis. This table can either be a data frame that you define in your R session, or a spreadsheet in .xlsx or .csv format. In a similar manner, the measurements themselves can either be provided as a list of data frames, or imported directly from files during runtime.

We demonstrate the functionality of the package using the dataset Panobinostat_2DTPP_smallExampleData. It contains an illustrative subset of a larger dataset which was obtained by 2D-TPP experiments on HepG2 cells treated with the histone deacetylase (HDAC) inhibitor panobinostat in the treatment groups and with vehicle in the control groups. The experiments were performed for different temperatures. The raw MS data were processed with the Python package isobarQuant, which provides protein fold changes relative to the protein abundance at the lowest temperature as input for the TPP package [3].

2.2 Performing the analysis

First of all, we load an example data set:

```r
data(panobinostat_2DTPP_smallExample, package = "TPP")
```

Using this command we load two objects:

1. **Panobinostat_2DTPP_smallExampleData**: a list of data frames that contain the measurements to be analyzed.
2. **hdac2D_config**: a configuration table with details about each experiment.

```r
cfg_tpp2d <- panobinostat_2DTPP_config
data_tpp2d <- panobinostat_2DTPP_data
```

```r
cfg_tpp2d
## Compound Experiment Temperature 126 127L 127H 128L 128H 129L 129H 130L 130H 131L
## 1 Panobinostat X020466 42.0 5 1 0.143 0.02 0 - - - - -
## 2 Panobinostat X020466 44.1 - - - - - 5 1 0.143 0.02 0
## 3 Panobinostat X020467 46.2 5 1 0.143 0.02 0 - - - - -
## 4 Panobinostat X020467 48.1 - - - - - 5 1 0.143 0.02 0
## 5 Panobinostat X020468 50.4 5 1 0.143 0.02 0 - - - - -
## 6 Panobinostat X020468 51.9 - - - - - 5 1 0.143 0.02 0
## 7 Panobinostat X020469 54.0 5 1 0.143 0.02 0 - - - - -
## 8 Panobinostat X020469 56.1 - - - - - 5 1 0.143 0.02 0
## 9 Panobinostat X020470 58.2 5 1 0.143 0.02 0 - - - - -
## 10 Panobinostat X020470 60.1 - - - - - 5 1 0.143 0.02 0
## 11 Panobinostat X020471 62.4 5 1 0.143 0.02 0 - - - - -
## 12 Panobinostat X020471 63.9 - - - - - 5 1 0.143 0.02 0
## `RefCol` `Path`
## 1 128H
## 2 131L
## 3 128H
## 4 131L
## 5 128H
## 6 131L
## 7 128H
## 8 131L
## 9 128H
## 10 131L
```
The data object `Panobinostat_2DTPP_smallExampleData` is organized as a list of data frames which contain the experimental raw data of a 2D-TPP experiment. The names of the list elements correspond to the different multiplexed experiments. Each experimental dataset contains the following columns:

```r
data_tpp2d %>% str()

## List of 6
## $ X020466:'data.frame': 484 obs. of 15 variables:
## $ X020467:'data.frame': 478 obs. of 15 variables:
## $ X020468:'data.frame': 448 obs. of 15 variables:
## $ X020469:'data.frame': 372 obs. of 15 variables:
## $ X020470:'data.frame': 306 obs. of 15 variables:
## $ X020471:'data.frame': 261 obs. of 15 variables:
```

In order to perform the complete workflow we can now simply use:

```r
tpp2dResults <- analyze2DTPP(configTable = config_tpp2d,
data = data_tpp2d,
compFc = TRUE,
idVar = "representative",
intensityStr = "sumionarea_protein_",
nonZeroCols = "qusm",
addCol = "clustername",
methods = "doseResponse",
createReport = "none")
```

## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
## Warning in function list[[k]](value): NAs introduced by coercion
tpp2dResults %>% mutate_if(is.character, factor) %>% summary

```
### Protein_ID norm_rel_fc_protein_0_unmodified
### X020466_42_IPI00000001.2: 1 Min. :1
### X020466_42_IPI00000005.1: 1 1st Qu.:1
### X020466_42_IPI00000690.1: 1 Median :1
### X020466_42_IPI00000811.2: 1 Mean :1
### X020466_42_IPI00000875.7: 1 3rd Qu.:1
### X020466_42_IPI00001466.2: 1 Max. :1
### (Other) :4650
### norm_rel_fc_protein_0.02_unmodified norm_rel_fc_protein_0.143_unmodified
### Min. :0.1767 Min. :0.2612
### 1st Qu.:0.9192 1st Qu.:0.9364
### Median :1.0000 Median :1.0000
### Mean :1.0035 Mean :1.0105
### 3rd Qu.:1.0727 3rd Qu.:1.0632
### Max. :10.0240 Max. :17.0405
### norm_rel_fc_protein_1_unmodified norm_rel_fc_protein_5_unmodified
### Min. : 0.2422 Min. : 0.2512
### 1st Qu.: 0.9344 1st Qu.: 0.9337
### Median : 1.0000 Median : 1.0000
### Mean : 1.0163 Mean : 1.0259
### 3rd Qu.: 1.0654 3rd Qu.: 1.0589
### Max. :10.0240 Max. :17.0405
### norm_rel_fc_protein_0_normalized_to_lowest_conc
### Min. :1
### 1st Qu.:1
### Median :1
### 3rd Qu.:1
### Max. :1
### norm_rel_fc_protein_0.02_normalized_to_lowest_conc
### Min. :0.1767
### 1st Qu.:0.9192
### Median :1.0000
### Mean :1.0035
### 3rd Qu.:1.0727
### Max. :4.6565
### norm_rel_fc_protein_0.143_normalized_to_lowest_conc
### Min. :0.2612
### 1st Qu.:0.9364
### Median :1.0000
### Mean :1.0105
### 3rd Qu.:1.0632
### Max. :5.8855
### norm_rel_fc_protein_1_normalized_to_lowest_conc
```
<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>norm_rel_fc_protein_5_normalized_to_lowest_conc</strong></td>
<td>0.2512</td>
<td>0.9337</td>
<td>1.0000</td>
<td>1.0259</td>
<td>1.0589</td>
<td>2.542</td>
<td>4421</td>
</tr>
<tr>
<td><strong>norm_rel_fc_protein_0_transformed</strong></td>
<td>0.000</td>
<td>0.000</td>
<td>1.0000</td>
<td>0.621</td>
<td>1.000</td>
<td>3.294</td>
<td>4421</td>
</tr>
<tr>
<td><strong>norm_rel_fc_protein_0.02_transformed</strong></td>
<td>-0.884</td>
<td>-0.154</td>
<td>0.297</td>
<td>0.302</td>
<td>0.614</td>
<td>2.542</td>
<td>4421</td>
</tr>
<tr>
<td><strong>norm_rel_fc_protein_0.143_transformed</strong></td>
<td>-1.201</td>
<td>0.086</td>
<td>0.376</td>
<td>0.400</td>
<td>1.000</td>
<td>0.621</td>
<td>4421</td>
</tr>
<tr>
<td><strong>norm_rel_fc_protein_1_transformed</strong></td>
<td>-0.961</td>
<td>-0.154</td>
<td>0.297</td>
<td>0.302</td>
<td>0.614</td>
<td>2.542</td>
<td>4421</td>
</tr>
<tr>
<td><strong>norm_rel.fc_protein_5_transformed</strong></td>
<td>-0.961</td>
<td>-0.154</td>
<td>0.297</td>
<td>0.302</td>
<td>0.614</td>
<td>2.542</td>
<td>4421</td>
</tr>
<tr>
<td><strong>pEC50</strong></td>
<td>0.000</td>
<td>0.000</td>
<td>1.0000</td>
<td>0.621</td>
<td>1.000</td>
<td>3.294</td>
<td>4421</td>
</tr>
<tr>
<td><strong>slope</strong></td>
<td>-50.000</td>
<td>-10.804</td>
<td>-1.000</td>
<td>-8.302</td>
<td>1.159</td>
<td>5.000</td>
<td>4421</td>
</tr>
<tr>
<td><strong>R_sq</strong></td>
<td>-0.068</td>
<td>0.545</td>
<td>0.723</td>
<td>0.675</td>
<td>0.881</td>
<td>1.000</td>
<td>4421</td>
</tr>
<tr>
<td><strong>compound_effect</strong></td>
<td>destablized: 146</td>
<td>stabilized : 89</td>
<td>NA's :4421</td>
<td>TRUE :119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>meets_FC_requirement</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>passed_filter</strong></td>
<td>1.000</td>
<td>1.000</td>
<td>9.149</td>
<td>87.000</td>
<td>6.07074587494624</td>
<td>6.75587159170968</td>
<td>6.83469502048232</td>
</tr>
<tr>
<td><strong>model_converged</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>pEC50_quality_check</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>sufficient_data_for_fit</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>protein_identified_in</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>representative</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
<tr>
<td><strong>qupm</strong></td>
<td>5.728</td>
<td>6.696</td>
<td>7.778</td>
<td>7.346</td>
<td>8.126</td>
<td>8.126</td>
<td>4421</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>qusm</strong></td>
<td>1.000</td>
<td>2.063e+05</td>
<td>3.819e+05</td>
<td>7.696e+07</td>
<td>7.604e+07</td>
<td>4584</td>
<td></td>
</tr>
<tr>
<td><strong>clusternne</strong></td>
<td>A2M : 12</td>
<td>1st Qu.:7.696e+07</td>
<td>1st Qu.:7.604e+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sumionarea_protein_5</strong></td>
<td>2.063e+05</td>
<td>3.819e+05</td>
<td>7.696e+07</td>
<td>7.604e+07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sumionarea_protein_1</strong></td>
<td>2.063e+05</td>
<td>3.819e+05</td>
<td>7.696e+07</td>
<td>7.604e+07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sufficient_data_for_fit</strong></td>
<td>1.000</td>
<td>3.000</td>
<td>7.000</td>
<td>9.149</td>
<td>12.000</td>
<td>87.000</td>
<td>4584</td>
</tr>
<tr>
<td><strong>protein_identified_in</strong></td>
<td>1.000</td>
<td>3.000</td>
<td>7.000</td>
<td>9.149</td>
<td>12.000</td>
<td>87.000</td>
<td>4584</td>
</tr>
<tr>
<td><strong>representative</strong></td>
<td>1.000</td>
<td>3.000</td>
<td>7.000</td>
<td>9.149</td>
<td>12.000</td>
<td>87.000</td>
<td>4584</td>
</tr>
</tbody>
</table>
Moreover, we can also invoke the single functions of the workflow manually. Therefore, we start with importing the data. Using the import function the data is subsequently imported and stored in a single dataframe containing all the required data columns and those that the user likes to take along through the analysis to be displayed together with the results of this workflow.
```r
data2d <- tpp2dImport(configTable = config_tpp2d,
data = data_tpp2d,
idVar = "representative",
intensityStr = "sumionarea_protein_",
nonZeroCols = "qusm",
addCol = "clustername")

## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
## Warning in function_list[[k]](value): NAs introduced by coercion
```

```r
head(data2d)
```

<table>
<thead>
<tr>
<th>representative</th>
<th>qupm</th>
<th>qusm</th>
<th>clustername</th>
<th>sumionarea_protein_5</th>
<th>sumionarea_protein_1</th>
<th>sumionarea_protein_0.143</th>
<th>sumionarea_protein_0.02</th>
<th>sumionarea_protein_0</th>
<th>temperature</th>
<th>experiment</th>
<th>unique_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1193994914</td>
<td>1337957734</td>
<td>1375948494</td>
<td>1956350223</td>
<td>1801848318</td>
<td>42.0</td>
<td>X020466</td>
</tr>
<tr>
<td>2</td>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1272771185</td>
<td>1473572092</td>
<td>1273285951</td>
<td>1669312103</td>
<td>1404292404</td>
<td>44.1</td>
<td>X020466</td>
</tr>
</tbody>
</table>
If we haven’t computed fold changes from the raw “sumionarea” data, as it is the case in this example, we can invoke the function `tpp2dComputeFoldChanges` in order to do so:

```r
fcData2d <- tpp2dComputeFoldChanges(data = data2d)
```

Thereon the function adds additional columns to our dataframe containing corresponding fold changes:

```r
head(fcData2d)
```

<table>
<thead>
<tr>
<th>representative</th>
<th>qupm</th>
<th>qusm</th>
<th>clustername</th>
<th>sumionarea_protein_5</th>
<th>sumionarea_protein_1</th>
<th>sumionarea_protein_0.143</th>
<th>sumionarea_protein_0.02</th>
<th>sumionarea_protein_0</th>
<th>temperature</th>
<th>experiment</th>
<th>unique_ID</th>
<th>rel_fc_5</th>
<th>rel_fc_1</th>
<th>rel_fc_0.143</th>
<th>rel_fc_0.02</th>
<th>rel_fc_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1193999414</td>
<td>1337957734</td>
<td>1375948494</td>
<td>1669312103</td>
<td>1404292404</td>
<td>42.0</td>
<td>1</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1272771185</td>
<td>1473572092</td>
<td>1273285951</td>
<td>1669312103</td>
<td>1404292404</td>
<td>44.1</td>
<td>2</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>13</td>
<td>22</td>
<td>STAU1</td>
<td>1482437522</td>
<td>1513181000</td>
<td>1284434575</td>
<td>1487032006</td>
<td>1422365664</td>
<td>46.2</td>
<td>3</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>13</td>
<td>22</td>
<td>STAU1</td>
<td>1157290962</td>
<td>1050288621</td>
<td>1110810226</td>
<td>1128507681</td>
<td>999666282</td>
<td>48.1</td>
<td>4</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>24</td>
<td>STAU1</td>
<td>396823892</td>
<td>458022616</td>
<td>453860821</td>
<td>412257039</td>
<td>439399665</td>
<td>50.4</td>
<td>5</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>24</td>
<td>STAU1</td>
<td>345196960</td>
<td>350182409</td>
<td>352193788</td>
<td>344410388</td>
<td>309019704</td>
<td>51.9</td>
<td>6</td>
<td>1.008712</td>
<td>1.087754</td>
<td>1.118721</td>
<td>1.0857463</td>
<td>1.0857463</td>
<td>1.008712</td>
</tr>
</tbody>
</table>
We can then normalize the data by performing a median normalization on the fold changes, in order to account for experiment specific noise.

```
normData2d <- tpp2dNormalize(data = fcData2d)
head(normData2d)
```

<table>
<thead>
<tr>
<th>representative</th>
<th>qum</th>
<th>qusm</th>
<th>clustrname</th>
<th>sumionarea_protein_5</th>
<th>sumionarea_protein_1</th>
<th>sumionarea_protein_0.143</th>
<th>sumionarea_protein_0.02</th>
<th>sumionarea_protein_0</th>
<th>temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1193994914</td>
<td>1337957734</td>
<td>1375948494</td>
<td>1956350223</td>
<td>1801848318</td>
<td>42.0</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>25</td>
<td>STAU1</td>
<td>1272771185</td>
<td>1473572092</td>
<td>1273285951</td>
<td>1669312103</td>
<td>1404292404</td>
<td>44.1</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>13</td>
<td>22</td>
<td>STAU1</td>
<td>1482437522</td>
<td>1513181000</td>
<td>1284434575</td>
<td>1487032006</td>
<td>1422365645</td>
<td>46.2</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>13</td>
<td>22</td>
<td>STAU1</td>
<td>1157290962</td>
<td>1050288621</td>
<td>1110810226</td>
<td>1128507681</td>
<td>999666282</td>
<td>48.1</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>24</td>
<td>STAU1</td>
<td>396823892</td>
<td>458022616</td>
<td>453860821</td>
<td>412257039</td>
<td>439399665</td>
<td>50.4</td>
</tr>
<tr>
<td>IPI00000001.2</td>
<td>15</td>
<td>24</td>
<td>STAU1</td>
<td>345169960</td>
<td>350182409</td>
<td>352193788</td>
<td>344410388</td>
<td>309019704</td>
<td>51.9</td>
</tr>
</tbody>
</table>

To run the TPP-CCR main function on our 2D-TPP data we now invoke:

```
ccr2dResults <- tpp2dCurveFit(data = normData2d)
```

Now we can plot the curves for any of the proteins for which at least one CCR curve could be fitted. In this case we choose HDAC2:

```
drPlots <- tpp2dCreateDRplots(data = ccr2dResults, type = "good")
```

```
# Find IPI id for HDAC2 (in column representative):
IPI_id_HDAC2 <- unique(filter(ccr2dResults, clustrname == "HDAC2")$representative)
```

```
# Show corresponding plot:
drPlots[[IPI_id_HDAC2]]
```
And we can also plot the single curves for each of the proteins with:

```r
drPlotsByTemperature <- tpp2dCreateDRplots(data = ccr2dResults, type = "single")
drPlotsByTemperature[[IPI_id_HDAC2]][["S4"]]
```

### 2.3 Quality control analyses

In order to access the quality of the experimental 2D-TPP data set acquired in a specific cell line, we recommend to compare the data with vehicle TR experiments (at least two replicates) of the same cell line. For the analysis of this data we supply a QC-workflow that enables comparison of treatment and non-treatment samples with reference data.

In order to start this workflow the first thing we need to do is to generate a cell line specific TR reference object. We also need to specify the result path where this object should be stored:

```r
resultPath = file.path(getwd(), 'Panobinostat_Vignette_Example_2D')
if (!file.exists(resultPath)) dir.create(resultPath, recursive = TRUE)
trConfig <- file.path(system.file("example_data", package="TPP"), "2D_example_data/panobinostat_ex_confg.csv")
tpp2dCreateTPPTRreference(trConfigTable = trConfig, resultPath = resultPath, outputName = "desired_file_name", createFCboxplots = FALSE)
```

For the purpose of explaining this workflow, we will use a reference data set of a HepG2 cell line supplied with this package. Originating from this object we can now perform various quality control steps. First of all by setting the `createFCboxplots` flag to true, we can generate box plot melting curves of the reference data which are first of all informative of the quality of the reference data and illustrate melting behavior of all proteins without any treatment.

Calling the function will generate a couple of output files in the indicated output directory.

- The `tppRefData.RData` file is the most important one. This is the file that has to be referenced by indication of a system path to this file when calling functions to generate the 2D-TPP spline plots and perform an F test. When loaded in R the object `tppRefData` represents a list with the following elements:
  - `tppCfgTable`: the TPP-TR configtable which was used for generating this object
  - `sumResTable`: a list of two elements:
    - `detail`: the exact result data from the TR analysis and
    - `summary`: a summary of the analyzed TR data comprising the median and standard deviation values of the measurements at the different temperatures (encoded by the isobaric labels)
  - `temperatures`: a table listing the temperatures which were used in the TR experiment in the different replicates
  - `lblsByTemp`: a table matching each temperature to an isobaric label
- An excel file which summarizes the data present in `tppRefData` on different sheets
Secondly, we can generate plots which visualize the melting point temperatures of the 2D-TPP data in comparison to the TR reference data. Here we demonstrate this function on a subset of the proteins:

```r
# set the system path for the HepG2 TR reference data set:
trRef <- file.path(system.file("data", package="TPP"), "TPPTR_reference_results_HepG2.RData")

plotData <- ccr2dResults %>% filter(clustername %in% IPI_id_HDAC2)

pEC50QC_HDAC1 <- tpp2dPlotQCpEC50(resultTable = plotData,
                                 resultPath = resultPath,
                                 trRef = trRef,
                                 idVar = "representative")

print(pEC50QC_HDAC1)
## named list()
```

We have therefore used the `ccr2dResults` data frame which we previously generated by invoking the TPP-CCR routine and the respective configTable.

Moreover, we can generate plots that visualize the distributions of fold changes over the different treatment concentrations and temperatures and how the normalization affected them (of course only if we previously performed a normalization). The function automatically also visualizes various other characteristics of the data, such as how proteins behave in neighboring temperatures which are multiplexed. It can be invoked as follows:

```r
tpp2dPlotQChist(configFile = config_tpp2d,
                 resultTable = ccr2dResults,
                 resultPath = resultPath,
                 trRef = trRef,
                 idVar = "representative")

dir(resultPath)
## [1] "qc_Histograms"
```

### 2.4 Spline fits of treatment effects over temperature

In order to access whether the drug treatment has a significant impact on altering the thermal stability of specific proteins a function was implemented which illustrates the course of stability of a certain protein over different temperatures based on a reference data set. A natural cubic spline fitted to the reference data is then used to infer the relative stability curves of proteins with different concentrations of treatment which are in turn fitted by natural cubic splines. The cubic spline with \( n \) degrees of freedom on \([a,b]\) obeys:

- \( S(x) \in C^2[a,b] \)
- \( a = t_0 < t_1 < ... < t_n = b \)

and:

\[
S(x) = \begin{cases} 
S_0(x) = a_0x^3 + b_0x^2 + c_0x + d_0, & t_0 \leq x \leq t_1 \\
S_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1, & t_1 \leq x \leq t_2 \\
\vdots \\
S_{n-1}(x) = a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1}, & t_{n-1} \leq x \leq t_n 
\end{cases}
\]

(1)

A natural cubic spline additionally constrains that it's function has to be linear beyond the boundary knots with constrains that both the first and the last section of the cubic spline has to be linear.
The function to perform this analysis can be invoked by:

```r
analysisResults <- tpp2dSplineFitAndTest(data = normData2d,
                                         dataRef = trRef,
                                         refIDVar = "Protein_ID",
                                         refFcStr = "norm_rel_fc_protein_",
                                         doPlot = FALSE,
                                         resultPath = resultPath,
                                         nCores = 1)
```

```r
head(analysisResults)
```

```
## representative qupm qusm clustername sumionarea_protein_5 sumionarea_protein_1
## 1 IPI00000001.2 15 25 STAU1 1193994914 1337957734
## 2 IPI00000001.2 15 25 STAU1 1272771185 1473572092
## 3 IPI00000001.2 13 22 STAU1 1482437522 1513181000
## 4 IPI00000001.2 13 22 STAU1 1157290962 1050288621
## 5 IPI00000001.2 15 24 STAU1 396823892 458022616
## 6 IPI00000001.2 15 24 STAU1 345169960 35182409
```

Moreover, these fits can be used then, in order to access confidence on whether the curves fitting the relative treatment data points represent the data better than a model which does not distinguish between the different treatment concentrations. The confidence assessment is thereby based on a moderated F statistic adapted from a method by Storey and others [6] which they developed for microarray time course data. The method calculates a
moderated F statistic following:

\[ F = \frac{SS_0 - SS_1}{\tilde{s}^2(\sigma^2, df_2)} \]  

with SS_0 representing the sum of squares of the null model (fitting the data without distinguishing between different treatment concentrations) and SS_1 those of the full model (which fits the data by in this case 5 different splines for every treatment concentration respectively). With \( \tilde{s}^2 \) representing the empirical Bayes estimator for SS_1, with df_2 = n – \( \nu_1 \), where \( \nu_1 \) denoted the parameters of the full model and n denotes the number of data points.

```
analysisResults %>% filter(representative == IPI_id_HDAC2) %>%
select(temperature, p_NPARC, p_adj_NPARC)
```

## temperature p_NPARC p_adj_NPARC
## 1 42.0 0 0
## 2 44.1 0 0
## 3 46.2 0 0
## 4 48.1 0 0
## 5 50.4 0 0
## 6 51.9 0 0
## 7 54.0 0 0
## 8 56.1 0 0
## 9 58.2 0 0
## 10 60.1 0 0
## 11 62.4 0 0
## 12 63.9 0 0

By defining the methods argument to include "splineFit", one prompts the main function analyze2DTPP to directly perform spline fits and a moderated F-test for each protein in the data set.

**References**


