

Package ‘tcp1’

November 5, 2015

Title ToxCast data analysis pipeline.

Version 1.0

Description <BETA VERSION> This package is used to process and model all of the high-throughput and high-content screening data generated by the US EPA ToxCast program. Note: This beta version has incomplete documentation, but was used to process all of the ToxCast data in the October 2014 release. An updated version with complete documentation will be available in the future.

URL <http://www.epa.gov/ncct/toxcast/>

Depends R (>= 3.2.0)

Imports data.table (>= 1.9.4),

DBI,
RSQLite,
MASS,
numDeriv,
RColorBrewer

Suggests RMySQL

License GPL-2

LazyData true

R topics documented:

| | |
|---------------|----|
| blinShift | 3 |
| flareFunc | 3 |
| interlaceFunc | 4 |
| is.odd | 5 |
| lu | 5 |
| lw | 6 |
| mc1 | 6 |
| mc2 | 7 |
| MC2_Methods | 8 |
| mc3 | 9 |
| MC3_Methods | 9 |
| mc4 | 12 |
| mc5 | 12 |
| MC5_Methods | 13 |
| mc6 | 14 |
| MC6_Methods | 15 |

| | |
|------------------|----|
| Models | 16 |
| registerMthd | 18 |
| sc1 | 18 |
| SC1_Methods | 19 |
| sc2 | 21 |
| SC2_Methods | 21 |
| sink.reset | 22 |
| tcplACVal | 23 |
| tcplACXX | 23 |
| tcplAddModel | 24 |
| tcplAICProb | 25 |
| tcplAppend | 25 |
| tcplAssignMthd | 26 |
| tcplCascade | 26 |
| tcplClearMthd | 27 |
| tcplCode2CASN | 27 |
| tcplDelete | 28 |
| tcplFit | 28 |
| tcplListFlds | 29 |
| tcplListMthd | 29 |
| tcplListOpts | 30 |
| tcplLoadAcid | 30 |
| tcplLoadAeid | 31 |
| tcplLoadAeidInfo | 31 |
| tcplLoadAid | 32 |
| tcplLoadAsid | 32 |
| tcplLoadAsidInfo | 33 |
| tcplLoadChem | 33 |
| tcplLoadClib | 34 |
| tcplLoadData | 34 |
| tcplLoadMthd | 35 |
| tcplLoadUnit | 35 |
| tcplMakeAeidPlts | 36 |
| tcplPlotFitc | 36 |
| tcplPlotFits | 37 |
| tcplPlotM4ID | 37 |
| tcplPlotPlate | 38 |
| tcplPrepOtp | 38 |
| tcplQuery | 39 |
| tcplRegister | 39 |
| tcplRun | 40 |
| tcplSendQuery | 41 |
| tcplSetOpts | 41 |
| tcplSpidMap | 42 |
| tcplSubsetChid | 42 |
| tcplUpdate | 43 |
| tcplVarMat | 43 |
| tcplWriteData | 45 |
| tcplWriteLvl0 | 45 |

| | |
|-----------|--------------------------------|
| blinShift | <i>Shift the baseline to 0</i> |
|-----------|--------------------------------|

Description

blinShift Takes in dose-response data and shifts the baseline to 0 based on the window.

Usage

```
blinShift(resp, logc, wndw)
```

Arguments

| | |
|------|---|
| resp | Numeric, the response values |
| logc | Numeric, the log10 concentration values |
| wndw | Numeric, the threshold window |

Details

Need to add...

Value

A numeric vector containing the shifted response values

Note

This function is not exported and is not intended to be used by the user.

See Also

[mc3_mthds](#), [mc3](#)

| | |
|-----------|---|
| flareFunc | <i>Calculate the weighted mean of a square to detect plate flares</i> |
|-----------|---|

Description

flareFunc calculates the weighted mean of square regions to detect plate flares.

Usage

```
flareFunc(val, coli, rowi, apid, r)
```

Arguments

| | |
|------|---|
| val | Numeric, the well values |
| coli | Integer, the well column index |
| rowi | Integer, the well row index |
| apid | Character, the assay plate id |
| r | Integer, the number of wells from the center well (in one direction) to make the square |

See Also

[mc6_mthds](#), [tcp1LoadMthd](#), [mc6](#)

| | |
|---------------|---|
| interlaceFunc | <i>Calculate the weighted mean of a square to detect interlace effect</i> |
|---------------|---|

Description

interlaceFunc calculates the distance weighted mean of square regions from a 384-well plate that is interlaced onto a 1536 well plate to detect non-random signals coming from the source plate

Usage

```
interlaceFunc(val, intq, coli, rowi, apid, r)
```

Arguments

| | |
|------|---|
| val | Numeric, the well values |
| intq | Numeric, interlace quadrant |
| coli | Integer, the well column index |
| rowi | Integer, the well row index |
| apid | Character, the assay plate id |
| r | Integer, the number of wells from the center well (in one direction) to make the square |

See Also

[mc6_mthds](#), [tcp1LoadMthd](#), [mc6](#)

| | |
|--------|------------------------------|
| is.odd | <i>Check for odd numbers</i> |
|--------|------------------------------|

Description

is.odd takes an integer vector, x, and returns TRUE for odd integers.

Usage

```
is.odd(x)
```

Arguments

| | |
|---|------------|
| x | An integer |
|---|------------|

Value

TRUE for odd integers and FALSE for even integers.

See Also

Other tcpl abbreviations: [lu](#); [lw](#); [sink.reset](#)

| | |
|----|---|
| lu | <i>Abbreviation for length(unique(x))</i> |
|----|---|

Description

lu takes a logical vector, x, and returns length(unique(x)).

Usage

```
lu(x)
```

Arguments

| | |
|---|-----------|
| x | A logical |
|---|-----------|

Value

The unique of the TRUE values in x

See Also

[unique](#), [which](#)

Other tcpl abbreviations: [is.odd](#); [lw](#); [sink.reset](#)

| | |
|----|--|
| lw | <i>Abbreviation for length(which(x))</i> |
|----|--|

Description

lw takes a logical vector, x, and returns length(which(x)).

Usage

```
lw(x)
```

Arguments

| | |
|---|-----------|
| x | A logical |
|---|-----------|

Value

The length of the TRUE values in x

See Also

[length](#), [which](#)

Other tcpl abbreviations: [is.odd](#); [lu](#); [sink.reset](#)

| | |
|-----|--|
| mc1 | <i>Perform level 1 multiple-concentration processing</i> |
|-----|--|

Description

mc1 loads level 0 data from the tcpl database for the given id and performs level 1 multiple-concentration processing. The processed data is then loaded into the mc1 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
mc1(ac, wr = FALSE)
```

Arguments

| | |
|----|--|
| ac | Integer of length 1, assay component id (acid) for processing. |
| wr | Logical, whether the processed data should be written to the tcpl database |

Details

Level 1 processing includes defining the concentration and replicate index, cndx and repi, respectively.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a `data.table` containing the processed data

See Also

Other multiple-concentration data processing functions: [mc2](#); [mc3](#); [mc4](#); [mc5](#); [mc6](#)

`mc2`*Perform level 2 multiple-concentration processing*

Description

`mc2` loads level 1 data from the `tcpl` database for the given `id` and performs level 2 multiple-concentration processing. The processed data is then loaded into the `mc2` table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
mc2(ac, wr = FALSE)
```

Arguments

| | |
|-----------------|---|
| <code>ac</code> | Integer of length 1, assay component id (acid) for processing. |
| <code>wr</code> | Logical, whether the processed data should be written to the <code>tcpl</code> database |

Details

Level 2 multiple-concentration processing includes defining the corrected value, `cval`, based on the correction methods listed in the `mc2_acid` and `mc2_methods` tables.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a `data.table` containing the processed data

See Also

[tcplLoadMthd](#), [MC2_Methods](#)

Other multiple-concentration data processing functions: [mc1](#); [mc3](#); [mc4](#); [mc5](#); [mc6](#)

Description

mc2_mthds returns a list of correction/transformation functions to be used during level 2 multiple-concentration processing.

Usage

```
mc2_mthds()
```

Details

The functions contained in the list returned by mc2_mthds return a list of expressions to be executed in the mc2 (not exported) function environment. The functions are described here for reference purposes. The mc2_mthds function is not exported, nor is it intended for use.

All available methods are described in the Available Methods section, listed by the function/method name.

Value

A list functions

Available Methods

More information about the level 2 multiple-concentration processing is available in the package vignette, "Pipeline_Overview."

log2 Take the logarithm of cval with the base 2.

log10 Take the logarithm of cval with the base 10.

rmneg Remove entries where cval is less than 0.

rmzero Remove entries where cval is 0.

mult25 Multiply cval by 25.

mult100 Multiply cval by 100.

negshift Shift cval by subtracting out the minimum of cval and adding 1, such that the new minimum of cval is 1.

mult25 Multiply cval by 2.5.

mult3 Multiply cval by 3.

mult6 Multiply cval by 6.

Note

This function is not exported and is not intended to be used by the user.

See Also

[mc2](#), [tcp1LoadMthd](#) to query what methods get applied to each acid

| | |
|-----|--|
| mc3 | <i>Perform level 3 multiple-concentration processing</i> |
|-----|--|

Description

mc3 loads level 2 data from the tcpl database for the given id and performs level 3 multiple-concentration processing. The processed data is then loaded into the mc3 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
mc3(ac, wr = FALSE)
```

Arguments

| | |
|----|--|
| ac | Integer of length 1, assay component id (acid) for processing. |
| wr | Logical, whether the processed data should be written to the tcpl database |

Details

Level 3 multiple-concentration processing includes mapping assay component to assay endpoint, duplicating the data when the assay component has multiple assay endpoints, and any normalization of the data. Data normalization based on methods listed in mc3_aeid and mc3_methods tables.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a data.table containing the processed data

See Also

[tcplLoadMthd](#), [MC3_Methods](#)

Other multiple-concentration data processing functions: [mc1](#); [mc2](#); [mc4](#); [mc5](#); [mc6](#)

| | |
|-------------|---|
| MC3_Methods | <i>List of level 3 multiple-concentration normalization methods</i> |
|-------------|---|

Description

mc3_mthds returns a list of normalization methods to be used during level 3 multiple-concentration processing.

Usage

```
mc3_mthds()
```

Details

The functions contained in the list returned by `mc3_mthds` take 'aids' (a numeric vector of aid values) and returns a list of expressions to be executed in the `mc3` (not exported) function environment. The functions are described here for reference purposes, The `mc3_mthds` function is not exported, nor is it intended for use.

All available methods are described in the Available Methods section, listed by the type of function and the function/method name.

Value

A list of functions

Available Methods

The methods are broken into three types, based on what fields they define. Different methods are used to define "bval" (the baseline value), "pval" (the positive control value), and "resp" (the final response value).

Although it does not say so specifically in each description, all methods are applied by `aid`.

More information about the level 3 multiple-concentration processing is available in the package vignette, "Pipeline_Overview."

bval Methods:

bval.apid.nwlls.med Calculate bval as the median of cval for wells with wllt equal to "n," by `apid`.

bval.apid.lowconc.med Calculate bval as the median of cval for wells with wllt equal to "t" and `cndx` equal to 1 or 2, by `apid`.

bval.apid.twlls.med Calculate bval as the median of cval for wells with wllt equal to "t," by `apid`.

bval.apid.tn.med Calculate bval as the median of cval for wells with wllt equal to "t" or "n," by `apid`.

bval.apid.nwllslowconc.med Calculate bval as the median of cval for wells with wllt equal to "n" or wells with wllt equal to "t" and `cndx` equal to 1 or 2, by `apid`.

bval.spid.lowconc.med Calculate bval as the median of cval for wells with wllt equal to "t" and `cndx` equal to 1, 2, or 3, by `spid`.

pval Methods:

pval.apid.pwlls.med Calculate pval as the median of cval for wells with wllt equal to "p," by `apid`.

pval.apid.mwlls.med Calculate pval as the median of cval for wells with wllt equal to "m," by `apid`.

pval.apid.medpbyconc.max First calculate the median of cval for wells with wllt equal to "p" or "c," by `wllt`, `conc`, and `apid`. Then calculate pval as the maximum of the calculated medians, by `apid`.

pval.apid.medpbyconc.min First calculate the median of cval for wells with wllt equal to "p" or "c," by `wllt`, `conc`, and `apid`. Then calculate pval as the minimum of the calculated medians, by `apid`.

pval.apid.medncbyconc.min First calculate the median of cval for wells with wllt equal to "m" or "o," by `wllt`, `conc`, and `apid`. Then calculate pval as the minimum of the calculated medians, by `apid`.

pval.apid.pmv.min First calculate the median of cval for wells with wllt equal to "p," "m," or "v," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.apid.pmv.max First calculate the median of cval for wells with wllt equal to "p," "m," or "v," by wllt, conc, and apid. Then calculate pval as the maximum of the calculated medians, by apid.

pval.apid.f.max First calculate the median of cval for wells with wllt equal to "f," by wllt, conc, and apid. Then calculate pval as the maximum of the calculated medians, by apid.

pval.apid.f.min First calculate the median of cval for wells with wllt equal to "f," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.apid.p.max First calculate the median of cval for wells with wllt equal to "p," by wllt, conc, and apid. Then calculate pval as the maximum of the calculated medians, by apid.

pval.apid.p.min First calculate the median of cval for wells with wllt equal to "p," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.apid.v.min First calculate the median of cval for wells with wllt equal to "v," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.zero Define pval as 0.

resp Methods:

resp.pc Calculate resp as $\frac{cval - bval}{pval - bval} 100$.

resp.fc Calculate resp as $cval / bval$.

resp.logfc Calculate resp as $cval - bval$.

resp.log2 Take the logarithm of resp with base 2.

resp.mult25 Multiply resp by 25.

resp.scale.mad.log2fc Multiply resp by the scale factor $\frac{\log_2(1.2)}{3bmad}$.

resp.scale.quant.log2fc Determine the maximum response md where $md = \text{abs}(1\text{st centile} - 50\text{th centile})$ or $\text{abs}(99\text{th centile} - 50\text{th centile})$, whichever is greater. Scale the response such that 20 percent of md equals $\log_2(1.2)$.

resp.multneg1 Multiply resp by -1.

resp.shiftneg.3bmad Shift all resp values less than $-3*bmad$ to 0.

resp.shiftneg.6bmad Shift all resp values less than $-6*bmad$ to 0.

resp.shiftneg.10bmad Shift all resp values less than $-10*bmad$ to 0.

resp.blineshift.3bmad.repi Shift resp values with the blineShift function by repi, where the window (wndw) is $3*bmad$.

resp.blineshift.50.repi Shift resp values with the blineShift function by repi, where the window (wndw) is 50.

resp.blineshift.3bmad.spid Shift resp values with the blineShift function by spid, where the window (wndw) is $3*bmad$.

resp.blineshift.50.spid Shift resp values with the blineShift function by spid, where the window (wndw) is 50.

none Do no normalization; make resp equal to cval.

Note

This function is not exported and is not intended to be used by the user.

See Also

[mc3](#), [tcp1LoadMthd](#) to query what methods get applied to each acid

mc4

Perform level 4 multiple-concentration processing

Description

mc4 loads level 3 data from the tcpl database for the given id and performs level 4 multiple-concentration processing. The processed data is then loaded into the mc4 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
mc4(ae, wr = FALSE)
```

Arguments

ae Integer of length 1, assay endpoint id (aeid) for processing.
wr Logical, whether the processed data should be written to the tcpl database

Details

Level 4 multiple-concentration modeling takes the dose-response data for chemical-assay pairs, and fits three models to the data: constant, hill, and gain-loss. For more information about the models see [Models](#). When a chemical has more than one sample, the function fits each sample separately.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a data.table containing the processed data

See Also

[tcplFit](#), [Models](#)

Other multiple-concentration data processing functions: [mc1](#); [mc2](#); [mc3](#); [mc5](#); [mc6](#)

mc5

Perform level 5 multiple-concentration processing

Description

mc5 loads level 4 data from the tcpl database for the given id and performs level 5 multiple-concentration processing. The processed data is then loaded into the mc5 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
mc5(ae, wr = FALSE)
```

Arguments

ae Integer of length 1, assay endpoint id (aeid) for processing.
wr Logical, whether the processed data should be written to the tcpl database

Details

Level 5 multiple-concentration hit-calling uses the fit parameters and the activity cutoff methods from `mc5_aeid` and `mc5_methods` to make an activity call and identify the winning model for each fit.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a `data.table` containing the processed data

See Also

[tcp1LoadMthd](#), [MC5_Methods](#)

Other multiple-concentration data processing functions: [mc1](#); [mc2](#); [mc3](#); [mc4](#); [mc6](#)

MC5_Methods

Load list of level 5 multiple-concentration cutoff methods

Description

`mc5_mthds` returns a list of additional activity cutoff methods to be used during level 5 multiple-concentration processing.

Usage

```
mc5_mthds()
```

Value

A list of functions

Available Methods

More information about the level 5 multiple-concentration processing is available in the package vignette, "Pipeline_Overview."

bm3 Add a cutoff value of 3*bm3.

pc20 Add a cutoff value of 20.

log2_1.2 Add a cutoff value of log2(1.2).

log10_1.2 Add a cutoff value of log10(1.2).

bmad5 Add a cutoff value of 5*bmad.

bmad6 Add a cutoff value of 6*bmad.

bmad10 Add a cutoff value of 10*bmad.

See Also

[mc5](#), [tcp1LoadMthd](#) to query what methods get applied to each aeid

mc6

Perform level 6 multiple-concentration processing

Description

mc6 loads level 5 data from the tcpl database for the given id and performs level 6 multiple-concentration processing. The processed data is then loaded into the mc6 table and all subsequent data is deleted with [tcp1Cascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcp1Run](#) wrapper function.

Usage

```
mc6(ae, wr = FALSE)
```

Arguments

| | |
|----|--|
| ae | Integer of length 1, assay endpoint id (aeid) for processing. |
| wr | Logical, whether the processed data should be written to the tcpl database |

Details

Level 6 multiple-concentration flagging uses both the plate level concentration-response data and the modeled parameters to flag potential false positives and false negative results.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a data.table containing the processed data

See Also

[tcp1LoadMthd](#), [MC6_Methods](#)

Other multiple-concentration data processing functions: [mc1](#); [mc2](#); [mc3](#); [mc4](#); [mc5](#)

Description

mc6_mthds returns a list of flag methods to be used during level 6 multiple-concentration processing.

Usage

```
mc6_mthds()
```

Value

A list functions

Available Methods

More information about the level 6 multiple-concentration processing is available in the package vignette, "Pipeline_Overview."

row.dev.up The row.dev.up flag looks at the individual point data, searching for row effects across an apid. To get flagged the point has to be greater than 3 standard deviations above the mean response for the plate, and the row mean must be greater than 3 standard deviations above the row means for the plate.

row.dev.dn The row.dev.dn flag is identical to the row.dev.up flag, but identifies points falling in rows with decreased signals.

col.dev.up The col.dev.up flag is identical to the row.dev.up flag, but identifies points falling in columns with increased signals.

col.dev.dn The col.dev.dn flag is identical to the row.dev.up flag, but identifies points falling in columns with decreased signals.

plate.flare The plate.flare flag looks at the individual point data, searching for overly active regions across an apid. Intended for use in fluorometric assays that are read by a plate-reader that measures the plate as a whole, rather than measuring individual wells. For each well the flare value is calculated as a weighted mean a 5 well by 5 well box centered on the well where the weight given to each well in the box is the euclidian distance from the center well. The flag then identifies points with flare values greater than 3 standard deviations above the mean flare values for the plate.

plate.interlace The plate.interlace flag is specific to one experimental design that plates chemicals from a 386 well chemical plate to a 1536 well assay plate. The flag looks for any chemical-plate affects, by looking for an increased signal in the wells originating from the same chemical plate.

rep.mismatch The rep.mismatch flag is still in development and is not suggested for use at this time.

pintool Deprecated. The pintool flag uses a complicated algorithm to look for signal potentially caused by residual in the pintool used to deliver the chemical to assay plates in some experimental designs. The gnls.lowconc is a faster and simpler way to identify where this problem may be driving the activity or hit-call.

- singlept.hit.high** The `singlept.hit.high` flag identifies concentration series where the median response was greater than $3 \cdot b_{mad}$ only at the highest tested concentration and the series had an active hit-call.
- singlept.hit.mid** The `singlept.hit.mid` flag identifies concentration series where the median response was greater than $3 \cdot b_{mad}$ at only one concentration (not the highest tested concentration) and the series had an active hit-call.
- multipoint.neg** The `multipoint.neg` flag identifies concentration series with response medians greater than $3 \cdot b_{mad}$ at multiple concentrations and an inactive hit-call.
- gnls.lowconc** The `gnls.lowconc` flag identifies concentration series where the gain-loss model won, the gain AC50 is less than the minimum tested concentration, and the loss AC50 is less than the mean tested concentration.
- noise** The noise flag attempts to identify noisy concentration series by flagging series where the root mean square error for the series is greater than the cutoff for the assay endpoint.
- border.hit** The `border.hit` flag identifies active concentration series where the top parameter of the winning model was less than or equal to $1.2 \cdot \text{cut-off}$ or the activity probability was less than 0.9.
- border.miss** The `border.miss` flag identifies inactive concentration series where either the Hill or gain-loss top parameter was greater than or equal to $0.8 \cdot \text{cut-off}$ and the activity probability was greater than 0.5.
- overfit.hit** The `overfit.hit` flag recalculates the model winner after applying a small sample correction factor to the AIC values. If the hit-call would be changed after applying the small sample correction factor the series is flagged. Series with less than 5 concentrations where the hill model won and series with less than 7 concentrations where the gain-loss model won are automatically flagged.
- efficacy.50** The `efficacy.50` flag identifies concentration series with efficacy values (either the modeled top parameter for the winning model or the maximum median response) are less than 50. Intended for use with biochemical assays where one might expect at least a 50% change in real responses.

See Also

[mc6](#), [tcp1LoadMthd](#) to query what methods get applied to each acid

Models

Model objective functions

Description

These functions take in the dose-response data and the model parameters, and return a likelihood value. They are intended to be optimized using `constrOptim` in the `tcp1Fit` function.

Usage

```
tcp1ObjCnst(p, resp)
```

```
tcp1ObjGnls(p, lconc, resp)
```

```
tcp1ObjHill(p, lconc, resp)
```

Arguments

| | |
|-------|--|
| p | Numeric, the parameter values. See details for more information. |
| resp | Numeric, the response values |
| lconc | Numeric, the log10 concentration values |

Details

These functions produce an estimated value based on the model and given parameters for each observation. Those estimated values are then used with the observed values and a scale term to calculate the log-likelihood.

Let $t(z, \nu)$ be the Student's t-distribution with ν degrees of freedom, y_i be the observed response at the i^{th} observation, and μ_i be the estimated response at the i^{th} observation. We calculate z_i as:

$$z_i = \frac{y_i - \mu_i}{e^\sigma}$$

where σ is the scale term. Then the log-likelihood is:

$$\sum_{i=1}^n [\ln(t(z_i, 4)) - \sigma]$$

Where n is the number of observations.

Value

The log-likelihood.

Constant Model (cnst)

tcp10bjCnst calculates the likelihood for a constant model at 0. The only parameter passed to tcp10bjCnst by p is the scale term σ . The constant model value μ_i for the i^{th} observation is given by:

$$\mu_i = 0$$

Gain-Loss Model (gnls)

tcp10bjGnls calculates the likelihood for a 5 parameter model as the product of two Hill models with the same top and both bottoms equal to 0. The parameters passed to tcp10bjGnls by p are (in order) top (tp), gain log AC50 (ga), gain hill coefficient (gw), loss log AC50 la , loss hill coefficient lw , and the scale term (σ). The gain-loss model value μ_i for the i^{th} observation is given by:

$$g_i = \frac{1}{1 + 10^{(ga-x_i)gw}}$$

$$l_i = \frac{1}{1 + 10^{(x_i-la)lw}}$$

$$\mu_i = tp(g_i)(l_i)$$

where x_i is the log concentration for the i^{th} observation.

Hill Model (hill)

tcplObjHill calculates the likelihood for a 3 parameter Hill model with the bottom equal to 0. The parameters passed to tcplObjHill by p are (in order) top (*tp*), log AC50 (*ga*), hill coefficient (*gw*), and the scale term (σ). The hill model value μ_i for the i^{th} observation is given by:

$$\mu_i = \frac{tp}{1 + 10^{(ga-x_i)gw}}$$

where x_i is the log concentration for the i^{th} observation.

| | |
|--------------|----------------------------------|
| registerMthd | <i>Add a new analysis method</i> |
|--------------|----------------------------------|

Description

registerMthd registers a new analysis method to the tcpl databases.

Usage

```
registerMthd(lvl, mthd, desc, nldr = 0L, type)
```

Arguments

| | |
|------|---|
| lvl | Integer of length 1, the level for the analysis method |
| mthd | Character, the name of the method |
| desc | Character, same length as mthd, the method description |
| nldr | Integer, 0 or 1, 1 if the method requires loading the dose- response data |
| type | Character of length 1, the data type, "sc" or "mc" |

Details

'mthd' must match a corresponding function name in the functions that load the methods, ie. mc2_mthds. 'nldr' only applies to level 6 methods.

| | |
|-----|--|
| sc1 | <i>Perform level 1 single-concentration processing</i> |
|-----|--|

Description

sc1 loads level 0 data from the tcpl database for the given id and performs level 1 single-concentration processing. The processed data is then loaded into the sc1 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
sc1(ac, wr = FALSE)
```

Arguments

| | |
|----|--|
| ac | Integer of length 1, assay component id (acid) for processing. |
| wr | Logical, whether the processed data should be written to the tcpl database |

Details

Level 1 single-concentration processing includes mapping assay component to assay endpoint, duplicating the data when the assay component has multiple assay endpoints, and any normalization of the data. Data normalization based on methods listed in `sc1_aeid` and `sc1_methods` tables.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a `data.table` containing the processed data

See Also

[tcp1LoadMthd](#), [SC1_Methods](#)

Other single-concentration data processing functions: [sc2](#)

SC1_Methods

List of level 1 single-concentration normalization functions

Description

`sc1_mthds` returns a list of functions to be used during level 1 single-concentration processing.

Usage

```
sc1_mthds()
```

Details

The functions contained in the list returned by `sc1_mthds` return a list of expressions to be executed in the `sc2` (not exported) function environment. The functions are described here for reference purposes. The `sc1_mthds` function is not exported, nor is it intended for use.

All available methods are described in the Available Methods section, listed by the function/method name.

Value

A list functions

Available Methods

The methods are broken into three types, based on what fields they define. Different methods are used to define "bval" (the baseline value), "pval" (the positive control value), and "resp" (the final response value).

Although it does not say so specifically in each description, all methods are applied by acid.

More information about the level 3 single-concentration processing is available in the package vignette, "Pipeline_Overview."

bval Methods:

bval.apid.nwlls.med Calculate bval as the median of rval for wells with wllt equal to "n," by apid.

bval.apid.twlls.med Calculate bval as the median of rval for wells with wllt equal to "t," by apid.

bval.apid.tn.med Calculate bval as the median of rval for wells with wllt equal to "t" or "n," by apid.

pval Methods:

pval.apid.pwlls.med Calculate pval as the median of rval for wells with wllt equal to "p," by apid.

pval.apid.mwlls.med Calculate pval as the median of rval for wells with wllt equal to "m," by apid.

pval.apid.medpbyconc.max First calculate the median of rval for wells with wllt equal to "p" or "c," by wllt, conc, and apid. Then calculate pval as the maximum of the calculated medians, by apid.

pval.apid.medpbyconc.min First calculate the median of rval for wells with wllt equal to "p" or "c," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.apid.medncbyconc.min First calculate the median of rval for wells with wllt equal to "m" or "o," by wllt, conc, and apid. Then calculate pval as the minimum of the calculated medians, by apid.

pval.zero Define pval as 0.

resp Methods:

resp.pc Calculate resp as $\frac{rval - bval}{pval - bval} 100$.

resp.fc Calculate resp as $rval / bval$.

resp.logfc Calculate resp as $rval - bval$.

resp.log2 Take the logarithm of resp with base 2.

resp.multneg1 Multiply resp by -1.

none Do no normalization; make resp equal to rval.

Note

This function is not exported and is not intended to be used by the user.

See Also

[sc1](#), [tcp1LoadMthd](#) to query what methods get applied to each acid

`sc2`*Perform level 2 single-concentration processing*

Description

sc2 loads level 1 data from the tcpl database for the given id and performs level 2 single-concentration processing. The processed data is then loaded into the sc2 table and all subsequent data is deleted with [tcplCascade](#). See details for more information.

The individual processing functions are no longer exported, as it is typically more convenient and suggested to use the [tcplRun](#) wrapper function.

Usage

```
sc2(ae, wr = FALSE)
```

Arguments

| | |
|----|--|
| ae | Integer of length 1, assay endpoint id (aeid) for processing. |
| wr | Logical, whether the processed data should be written to the tcpl database |

Details

Level 2 single-concentration processing defines the bmad value, and uses the activity cutoff methods from `sc2_aeid` and `sc2_methods` to make an activity call.

Value

A boolean of length 1, indicating the success of the processing, or when 'wr' is FALSE, a list where the first element is a boolean indicating the success of processing and the second element is a `data.table` containing the processed data

See Also

[tcplLoadMthd](#), [SC2_Methods](#)

Other single-concentration data processing functions: [sc1](#)

`SC2_Methods`*List of level 2 single-concentration hit-call functions*

Description

`sc2_mthds` returns a list of functions to be used during level 2 single-concentration processing.

Usage

```
sc2_mthds()
```

Details

The functions contained in the list returned by `sc2_mthds` return a list of expressions to be executed in the `sc2` (not exported) function environment. The functions are described here for reference purposes. The `sc2_mthds` function is not exported, nor is it intended for use.

All available methods are described in the Available Methods section, listed by the function/method name.

Value

A list functions

Available Methods

More information about the level 2 single-concentration processing is available in the package vignette, "Pipeline_Overview."

bmad3 Add a cutoff value of $3*bmad$.

pc20 Add a cutoff value of 20.

log2_1.2 Add a cutoff value of $\log_2(1.2)$.

log10_1.2 Add a cutoff value of $\log_{10}(1.2)$.

bmad5 Add a cutoff value of $5*bmad$.

bmad6 Add a cutoff value of $6*bmad$.

bmad10 Add a cutoff value of $10*bmad$.

pc30orbmad3 Add a cutoff value of either 30 or $3*bmad$, whichever is less.

Note

This function is not exported and is not intended to be used by the user.

See Also

[sc2](#), [tcpLoadMthd](#) to query what methods get applied to each acid

sink.reset

Reset all sinks

Description

`sink.reset` resets all sinks and returns all output to the console.

Usage

```
sink.reset()
```

Details

`sink.reset` identifies all sinks with `sink.number` then returns all output and messages back to the console.

See Also

[sink](#), [sink.number](#)

Other tcpl abbreviations: [is.odd](#); [lu](#); [lw](#)

| | |
|-----------|---|
| tcplACVal | <i>Calculate the activity concentration for a given value</i> |
|-----------|---|

Description

tcplACVal computes the activity concentration for a Hill model for a \ given value, assuming a positive hill coefficient corresponds to an increasing Hill model.

Usage

```
tcplACVal(val, tp, ga, gw, bt = 0)
```

Arguments

| | |
|-----|---|
| val | Numeric, the Hill model value |
| tp | Numeric, the top value from the Hill model |
| ga | Numeric, the logAC50 value from the Hill model |
| gw | Numeric, the Hill coefficient from the Hill model |
| bt | Numeric, the bottom value from the Hill model |

Value

The activity concentration for the given value.

See Also

[tcplACXX](#)

| | |
|----------|--|
| tcplACXX | <i>Calculate the activity concentration for a given activity level</i> |
|----------|--|

Description

tcplACXX computes the activity concentration for a Hill model for a given activity level, assuming a positive hill coefficient corresponds to an increasing Hill model.

Usage

```
tcplACXX(XX, tp, ga, gw, bt = 0)
```

Arguments

| | |
|----|---|
| XX | Numeric, the activity level |
| tp | Numeric, the top value from the Hill model |
| ga | Numeric, the logAC50 value from the Hill model |
| gw | Numeric, the Hill coefficient from the Hill model |
| bt | Numeric, the bottom value from the Hill model |

Value

The activity concentration for the given activity level

See Also

[tcplACVal](#)

| | |
|--------------|--|
| tcplAddModel | <i>Draw a tcpl Model onto an existing plot</i> |
|--------------|--|

Description

tcplAddModel draws a line for one of the tcpl Models (see [Models](#) for more information) onto an existing plot.

Usage

```
tcplAddModel(pars, modl = NULL, adj = NULL, ...)
```

Arguments

| | |
|------|---|
| pars | List of parameters from level 4 or 5 output |
| modl | Character of length 1, the model to plot: 'cnst,' 'hill,' or 'gnls' |
| adj | Numeric of length 1, an adjustment factor, see details for more information |
| ... | Additional arguments passed to curve |

Details

tcplAddModel draws the model line assuming the x-axis represents log base 10 concentration.

If modl is NULL, the function checks pars\$modl and will return an error if pars\$modl is also NULL.

adj is intended to scale the models, so that models with different response units can be visualized on a single plot. The recommended value for adj is $1/(3 \cdot bmad)$ for level 4 data and $1/coff$ for level 5 data. If adj is NULL the function will check pars\$adj and set adj to 1 if pars\$adj is also NULL.

See Also

[Models](#), [tcplPlotFits](#)

| | |
|-------------|--|
| tcplAICProb | <i>Calculate the AIC probabilities</i> |
|-------------|--|

Description

tcplAICProb Calculates the probability that the model best represents the data based on the AIC value for each model.

Usage

```
tcplAICProb(...)
```

Arguments

... Numeric vectors of AIC values

Details

The function takes vectors of AIC values. Each vector represents the model AIC values for multiple observation sets. Each vector must contain the same number and order of observation sets.

Value

A vector of probability values for each model given, as a list.

See Also

[tcplFit](#), [AIC](#) for more information about AIC values.

| | |
|------------|-------------------------------|
| tcplAppend | <i>Append rows to a table</i> |
|------------|-------------------------------|

Description

tcplAppend takes a data.table (dat) and appends the data.table into a database table.

Usage

```
tcplAppend(dat, tbl, db)
```

Arguments

dat data.table, the data to append to a table
tbl Character of length 1, the table to append to
db Character of length 1, the database containing tbl

Note

This function is not exported and not intended to be used by the user.

| | |
|----------------|----------------------------------|
| tcplAssignMthd | <i>Assign an analysis method</i> |
|----------------|----------------------------------|

Description

tcplAssignMthd assigns an analysis method for the given pipeline level and ids.

Usage

```
tcplAssignMthd(lvl, id, mthd_id, ordr = NULL, type)
```

Arguments

| | |
|---------|--|
| lvl | Integer of length 1, the method level |
| id | Integer, the assay component or assay endpoint id(s) |
| mthd_id | Integer, the method id(s) |
| ordr | Integer, the order in which to execute the analysis methods, must be the same length as mthd_id, does not apply to levels 5 or 6 |
| type | Character of length 1, the data type, "sc" or "mc" |

Details

Add details to documentation.

See Also

Other method functions: [tcplClearMthd](#); [tcplListMthd](#); [tcplLoadMthd](#)

| | |
|-------------|---|
| tcplCascade | <i>Do a cascading delete on tcpl screening data</i> |
|-------------|---|

Description

tcplCascade deletes the data for the given id(s) starting at the processing level given. The delete will cascade through all subsequent tables.

Usage

```
tcplCascade(lvl, type, id)
```

Arguments

| | |
|------|---|
| lvl | Integer of length 1, the first level to delete from |
| type | Character of length 1, the data type, "sc" or "mc" |
| id | Integer, the id(s) to delete. See details for more information. |

Details

The data type can be either 'mc' for multiple concentration data, or 'sc' for single concentration data. Multiple concentration data will be loaded into the level tables, whereas the single concentration will be loaded into the single tables.

If lvl is less than 3, id is interpreted as acid(s) and if lvl is greater than or equal to 3, id is interpreted as aeid(s).

Note

This function is not exported and not intended to be used by the user.

| | |
|---------------|---|
| tcp1ClearMthd | <i>Clear analysis method(s) for the given ids</i> |
|---------------|---|

Description

tcp1ClearMthd clears analysis method(s) for the given pipeline level and ids.

Usage

```
tcp1ClearMthd(lvl, id, mthd_id = NULL, type)
```

Arguments

| | |
|---------|--|
| lvl | Integer of length 1, the method level |
| id | Integer, the assay component or assay endpoint id(s) |
| mthd_id | Integer, the method id(s) |
| type | Character of length 1, the data type, "sc" or "mc" |

See Also

Other method functions: [tcp1AssignMthd](#); [tcp1ListMthd](#); [tcp1LoadMthd](#)

| | |
|---------------|---|
| tcp1Code2CASN | <i>Convert chemical code to CAS Registry Number</i> |
|---------------|---|

Description

tcp1Code2CASN takes a code and converts it CAS Registry Number.

Usage

```
tcp1Code2CASN(code)
```

Arguments

| | |
|------|--|
| code | Character of length 1, a chemical code |
|------|--|

Value

A CAS Registry Number.

| | |
|------------|--|
| tcplDelete | <i>Delete rows from tcpl databases</i> |
|------------|--|

Description

tcplDelete deletes rows from the given table and database.

Usage

```
tcplDelete(tbl, fld, val, db)
```

Arguments

| | |
|-----|---|
| tbl | Character, length 1, the table to delete from |
| fld | Character, the field(s) to query on |
| val | List, vectors of values for each field to query on. Must be in the same order as 'fld'. |
| db | Character, the database containing the table |

Note

This function is not exported and not intended to be used by the user.

See Also

[tcplSendQuery](#)

| | |
|---------|---|
| tcplFit | <i>Fit the data with the constant, hill, and gain-loss models</i> |
|---------|---|

Description

tcplFit fits the constant, hill, and gain-loss models to the given data and returns some summary statistics and the fit parameters in a list.

Usage

```
tcplFit(logc, resp, bmad, force.fit = FALSE, ...)
```

Arguments

| | |
|------|--|
| logc | Numeric, log concentration values |
| resp | Numeric, normalized response values |
| bmad | Numeric, the baseline median absolute deviation for the entire assay |
| ... | Any other data to be included in list output. |

Value

List of summary values and fit parameters for the given data.

See Also

[tcplObjCnst](#), [tcplObjHill](#), [tcplObjGnls](#), [constrOptim](#)

| | |
|--------------|---|
| tcplListFlds | <i>Load the field names for a table</i> |
|--------------|---|

Description

tcplListFlds loads the column names for the given table and database.

Usage

```
tcplListFlds(tbl, db = options()$TCPL_DB)
```

Arguments

| | |
|-----|--|
| tbl | Character of length 1, the tcpl database table |
| db | Character of length 1, the tcpl database |

Value

A string of field names for the given table.

| | |
|--------------|---|
| tcplListMthd | <i>List the methods available in the tcpl databases</i> |
|--------------|---|

Description

tcplListMthd lists the methods available in the tcpl databases for the given processing level.

Usage

```
tcplListMthd(lvl, type = "mc")
```

Arguments

| | |
|------|--|
| lvl | Integer of length 1, the method level |
| type | Character of length 1, the data type, "sc" or "mc" |

See Also

Other method functions: [tcplAssignMthd](#); [tcplClearMthd](#); [tcplLoadMthd](#)

| | |
|--------------|--|
| tcplListOpts | <i>List the current tcpl option values</i> |
|--------------|--|

Description

tcplListOpts lists the values assigned to the tcpl global options.

Usage

```
tcplListOpts(show.pass = FALSE)
```

Arguments

show.pass Logical, should the password be returned

Value

A list containing the tcpl options and their current values

See Also

[tcplSetOpts](#)

| | |
|--------------|--|
| tcplLoadAcid | <i>Load assay component id and name for the given fields</i> |
|--------------|--|

Description

tcplLoadAcid queries the tcpl databases and returns a data.table with the assay component id (acid) and assay component name (acnm) values for the given fields

Usage

```
tcplLoadAcid(fld = NULL, val = NULL, add.fld = NULL)
```

Arguments

fld Character, the field(s) to query/subset on
 val List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'.
 add.fld Character, additional field(s) to include, but not query/ subset on

Value

A data.table containing the acid, acnm, and given fields.

See Also

[tcplQuery](#), [data.table](#)

| | |
|--------------|---|
| tcplLoadAeid | <i>Load assay endpoint id and name for the given fields</i> |
|--------------|---|

Description

tcplLoadAeid queries the tcpl databases and returns a `data.table` with the assay endpoint id (aeid) and assay endpoint name (aenm) values for the given fields

Usage

```
tcplLoadAeid(fld = NULL, val = NULL, add.fld = NULL)
```

Arguments

| | |
|---------|--|
| fld | Character, the field(s) to query/subset on |
| val | List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'. |
| add.fld | Character, additional field(s) to include, but not query/ subset on |

Value

A `data.table` containing the aeid, aenm, and given fields.

See Also

[tcplQuery](#), [data.table](#)

| | |
|------------------|--|
| tcplLoadAeidInfo | <i>Load assay endpoint information</i> |
|------------------|--|

Description

tcplLoadAeidInfo queries the tcpl databases and returns a `data.table` with assay annotation information for the given assay endpoint ids (aeid).

Usage

```
tcplLoadAeidInfo(fld = NULL, val)
```

Arguments

| | |
|-----|--|
| fld | Character, the field(s) to query/subset on |
| val | List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'. |

Value

A `data.table` containing assay information for the given aeids.

See Also

[tcplQuery](#), [data.table](#)

| | |
|-------------|--|
| tcplLoadAid | <i>Load assay id and name for the given fields</i> |
|-------------|--|

Description

tcplLoadAid queries the tcpl databases and returns a data.table with the assay id (aid) and assay name (anm) values for the given fields.

Usage

```
tcplLoadAid(fld = NULL, val = NULL, add.fld = NULL)
```

Arguments

| | |
|---------|--|
| fld | Character, the field(s) to query/subset on |
| val | List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'. |
| add.fld | Character, additional field(s) to include, but not query/ subset on |

Value

A data.table containing the aid, anm, and given fields.

See Also

[tcplQuery](#), [data.table](#)

| | |
|--------------|---|
| tcplLoadAsid | <i>Load assay source id and name for the given fields</i> |
|--------------|---|

Description

tcplLoadAsid queries the tcpl databases and returns a data.table with the assay source id (asid) and assay source name (asnm) values for the given fields.

Usage

```
tcplLoadAsid(fld = NULL, val = NULL, add.fld = NULL)
```

Arguments

| | |
|---------|--|
| fld | Character, the field(s) to query/subset on |
| val | List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'. |
| add.fld | Character, additional field(s) to include, but not query/ subset on |

Value

A data.table containing the asid, asnm, and given fields.

See Also[tcplQuery](#), [data.table](#)

| | |
|------------------|--|
| tcplLoadAsidInfo | <i>Load assay source table from tcpl databases</i> |
|------------------|--|

Description

tcplLoadAsidInfo queries the tcpl databases and returns a data.table with the assay source information.

Usage

```
tcplLoadAsidInfo()
```

Value

A data.table containing the assay source information.

See Also[tcplQuery](#), [data.table](#)

| | |
|--------------|---|
| tcplLoadChem | <i>Load sample/chemical information</i> |
|--------------|---|

Description

tcplLoadChem queries the tcpl database and returns the chemical information for the given field and values.

Usage

```
tcplLoadChem(field = NULL, val = NULL, exact = TRUE,  
             include.spid = TRUE)
```

Arguments

| | |
|--------------|---|
| field | Character of length 1, the field to query on |
| val | Vector of values to subset on |
| exact | Logical, should chemical names be considered exact? |
| include.spid | Logical, should spid be included? |

Value

A data.table with the chemical information for the given parameters

| | |
|--------------|--|
| tcpIloadClib | <i>Load chemical library information</i> |
|--------------|--|

Description

tcpIloadClib queries the tcpI databases and returns information about the chemical library.

Usage

```
tcpIloadClib(field = NULL, val = NULL)
```

Arguments

| | |
|-----|--|
| val | The values to query on |
| fld | Character of length 1, 'chid' or 'clib', whether to search by chemical id (chid), or chemical library (clib) |

Value

A data.table with the chemical library information for the given parameters.

| | |
|--------------|-----------------------|
| tcpIloadData | <i>Load tcpI data</i> |
|--------------|-----------------------|

Description

tcpIloadData queries the tcpI databases and returns a data.table with data for the given level and data type.

Usage

```
tcpIloadData(lvl, fld = NULL, val = NULL, type = "mc")
```

Arguments

| | |
|------|---|
| lvl | Integer of length 1, the level of data to load |
| fld | Character, the field(s) to query on |
| val | List, vectors of values for each field to query on. Must be in the same order as 'fld'. |
| type | Character of length 1, the data type, "sc" or "mc" |

Details

The data type can be either 'mc' for multiple concentration data, or 'sc' for single concentration data. Multiple concentration data will be loaded into the 'mc' tables, whereas the single concentration will be loaded into the 'sc' tables.

Setting 'lvl' to "agg" will return an aggregate table containing the m4id with the concentration-response data and m3id to map back to well-level information.

Leaving fld NULL will return all data.

Value

A data.table containing data for the given fields.

See Also

[tcplQuery](#), [data.table](#)

| | |
|--------------|--|
| tcplLoadMthd | <i>Load analysis method(s) for the given ids</i> |
|--------------|--|

Description

tcplLoadMthd loads analysis method(s) for the given pipeline level and ids.

Usage

```
tcplLoadMthd(lvl, id = NULL, type = "mc")
```

Arguments

| | |
|------|--|
| lvl | Integer of length 1, the method level |
| id | Integer, the assay component or assay endpoint id(s) |
| type | Character of length 1, the data type, "sc" or "mc" |

See Also

Other method functions: [tcplAssignMthd](#); [tcplClearMthd](#); [tcplListMthd](#)

| | |
|--------------|--|
| tcplLoadUnit | <i>Load response units for assay endpoints</i> |
|--------------|--|

Description

tcplLoadUnit queries the tcpl databases and returns a data.table with the response units for the given assay endpoint ids (aeid).

Usage

```
tcplLoadUnit(aeid)
```

Arguments

| | |
|------|-----------------------------|
| aeid | Integer, assay endpoint ids |
|------|-----------------------------|

Value

A data.table containing level 3 correction methods for the given aeids.

See Also

[tcplQuery](#), [data.table](#)

tcplMakeAeidPlts *Create a .pdf with dose-response plots*

Description

tcplMakeAeidPlts creates a .pdf file with the dose-response plots for the given aeid.

Usage

```
tcplMakeAeidPlts(aeid, lvl = 4L, odir = getwd(), ordr.fitc = TRUE,  
  clib = NULL, srgx = NULL)
```

Arguments

| | |
|-----------|--|
| aeid | Integer of length 1, the assay endpoint id |
| lvl | Integer of length 1, the data level to use (4-6) |
| odir | The directory to save the .pdf file in |
| ordr.fitc | Logical, should the fits be ordered by fit category? |
| clib | Character, the chemical libraries to include |
| srgx | Character of length 1, a regular expression to select specific spids |

tcplPlotFitc *Plot the fit category tree*

Description

tcplPlotFitc makes a plot showing the level 5 fit categories.

Usage

```
tcplPlotFitc(fitc = NULL, main = NULL, fitc_sub = NULL)
```

Arguments

| | |
|-----------|---|
| fitc | Integer, the fit categories |
| main | Character of length 1, the title (optional) |
| fitc_sub, | Integer, a subset of fit categories to plot |

Note

For ideal plotting use: width = 10, height = 7.5, pointsize = 9

| | |
|--------------|--|
| tcplPlotFits | <i>Plot summary fits based on fit and dose-response data</i> |
|--------------|--|

Description

tcplPlotFits takes the dose-response and fit data and produces summary plot figures.

Usage

```
tcplPlotFits(dat, agg, flg = NULL, ordr.fitc = FALSE, browse = FALSE)
```

Arguments

| | |
|-----------|---|
| dat | data.table, level 4 or level 5 data, see details. |
| agg | data.table, concentration-response aggregate data, see details. |
| flg | data.table, level 6 data, see details. |
| ordr.fitc | Logical, should the fits be ordered by fit category? |
| browse | Logical, should browser() be called after every plot? |

Details

The data for 'dat', 'agg', and 'flg' should be loaded using the [tcplLoadData](#) function with the appropriate 'lvl' parameter. See help page for tcplLoadData for more information.

| | |
|--------------|--------------------------------------|
| tcplPlotM4ID | <i>Plot fit summary plot by m4id</i> |
|--------------|--------------------------------------|

Description

tcplPlotM4ID creates a summary plots for the given m4id(s) by loading the appropriate data from the tcpl databases and sending it to [tcplPlotFits](#)

Usage

```
tcplPlotM4ID(m4id, lvl = 4L)
```

Arguments

| | |
|------|------------------------------------|
| m4id | Integer, m4id(s) to plot |
| lvl | Integer, the level of data to plot |

| | |
|---------------|---------------------------|
| tcplPlotPlate | <i>Plot plate heatmap</i> |
|---------------|---------------------------|

Description

tcplPlotPlate generates a heatmap of assay plate data

Usage

```
tcplPlotPlate(dat, apid, id = NULL)
```

Arguments

| | |
|------|---|
| dat | data.table containing tcpl data |
| apid | Character of length 1, the apid to plot |
| id | Integer of length 1, the assay component id (acid) or assay endpoint id (aeid), depending on level. Only need to specify for multiplexed assays when more than one acid/aeid share an apid. |

Note

For the optimal output size, use width = 10, height = 10*(2/3), pointsize = 10, units = "in"

| | |
|-------------|--|
| tcplPrepOtp | <i>Prepare data for output with chemical and assay information</i> |
|-------------|--|

Description

tcplPrepOtp queries the chemical and assay information from the tcpl databases, and maps the information to the given data

Usage

```
tcplPrepOtp(dat, clib = NULL, srgx = NULL, argx = NULL)
```

Arguments

| | |
|------|--|
| dat | Output from the data loading functions |
| clib | Character, the chemical libraries to include |
| srgx | Character of length 1, a regular expression to select specific spids |
| argx | Character of length 1, a regular expression to select specific assay names |

Value

The given data.table with chemical and assay information mapped

| | |
|-----------|---------------------------------|
| tcplQuery | <i>Query the tcpl databases</i> |
|-----------|---------------------------------|

Description

tcplQuery queries the tcpl databases and returns a data.table.

Usage

```
tcplQuery(query, db = options()$TCPL_DB)
```

Arguments

query, db A character of length 1

Value

A data.table containing the results from the given query.

See Also

[dbConnect](#), [data.table](#)

| | |
|--------------|---|
| tcplRegister | <i>Register new assay or chemical information</i> |
|--------------|---|

Description

tcplRegister adds new assay or chemical information to the tcpl database.

Usage

```
tcplRegister(what, flds)
```

Arguments

what Character of length 1, what to register

flds Named list, the values to register

| | |
|---------|--------------------------------|
| tcp1Run | <i>Perform data processing</i> |
|---------|--------------------------------|

Description

tcp1Run is the function for performing the data processing, for both single-concentration and multiple-concentration formats.

Usage

```
tcp1Run(asid = NULL, slvl, elvl, id = NULL, type = "mc",
        mc.cores = NULL, outfile = NULL, runname = NULL)
```

Arguments

| | |
|----------|--|
| asid | Integer, assay source id |
| slvl | Integer of length 1, the starting level to process |
| elvl | Integer of length 1, the ending level to process |
| id | Integer, rather than assay source id, the specific assay component or assay end-point id(s) (optional) |
| type | Character of length 1, the data type, "sc" or "mc" |
| mc.cores | Integer of length 1, the number of cores to use, set to 1 when using Windows operating system |
| outfile | Character of length 1, the name of the log file (optional) |
| runname | Character of length 1, the name of the run to be used in the outfile (optional) |

Details

The tcp1Run function is the core processing function within the package. The function acts as a wrapper for individual processing functions, (ie. mc1, sc1, etc.) that are not exported. If possible, the processing is done in parallel by 'id' by utilizing the [mclapply](#) function within the parallel package.

If slvl is less than 4, 'id' is interpreted as acid and if slvl is 4 or greater 'id' is interpreted as aeid. Must give either 'asid' or 'id'. If an id fails no results get loaded into the database, and the id does not get placed into the cue for subsequent level processing.

The 'type' parameter specifies what type of processing to complete: "mc" for multiple-concentration processing, and "sc" for single-concentration processing.

Value

A list containing the results from each level of processing. Each level processed will return a named logical vector, indicating the success of the processing for the id.

| | |
|---------------|---|
| tcplSendQuery | <i>Send query to the tcpl databases</i> |
|---------------|---|

Description

tcplSendQuery sends queries to the tcpl databases.

Usage

```
tcplSendQuery(query, db = options()$TCPL_DB)
```

Arguments

query, db A character of length 1

See Also

[dbConnect](#), [data.table](#)

| | |
|-------------|-----------------------------|
| tcplSetOpts | <i>Set the tcpl options</i> |
|-------------|-----------------------------|

Description

tcplSetOpts changes options to set the tcpl-specific options, most importantly to configure the connection to the tcpl databases.

Usage

```
tcplSetOpts(drvr = NULL, user = NULL, pass = NULL, host = NULL,  
          db = NULL, int = NULL)
```

Arguments

| | |
|------|--|
| drvr | Character of length 1, which database driver to use |
| user | Character of length 1, the database server username |
| pass | Character of length 1, the database server password |
| host | Character of length 1, the database server |
| db | Character of length 1, the name of the tcpl database |
| int | Logical, FALSE if not using internal ToxCast databases for chemical information. |
| log | Character of length 1, the directory for the log file |

Details

tcplSetOpts will only change non-null values, and can be used to change a single value if needed.

See Also

[tcplListOpts](#) [dbConnect](#)

| | |
|-------------|---|
| tcplSpidMap | <i>Map old spid to new spid (plate_well_id)</i> |
|-------------|---|

Description

tcplSpidMap takes a data.table with level 0 screening data and maps the spid column (source spid) to the updated spid, where needed.

Usage

```
tcplSpidMap(dat)
```

Arguments

| | |
|-----|---|
| dat | data.table, the screening data to update spid map |
|-----|---|

Details

This function maps the spid_map spid_source, asid, aid and srcf to dat. (INTERNAL FUNCTION ONLY)

Note

This function should only be used internally to update spids

| | |
|----------------|--|
| tcplSubsetChid | <i>Subset level 5 data to a single sample per chemical</i> |
|----------------|--|

Description

tcplSubsetChid creates a chemical by assay matrix for the given variable. Additional subsetting can be done with the clib, srgx, and argx parameters.

Usage

```
tcplSubsetChid(dat, flag = TRUE)
```

Arguments

| | |
|------|---|
| dat | data.table, a data.table with level 5 data |
| flag | Integer, the mc6_mthd_id values to go into the flag count, see details for more information |

Details

tcplSubsetChid is intended to work with level 5 data that has chemical and assay information mapped with [tcplPrep0tpt](#).

To select a single sample, first a "consensus hit-call" is made by majority rule, with ties defaulting to active. After the chemical-wise hit call is made, the samples corresponding to to chemical-wise hit call are logically ordered using the fit category, the number of the flags, and the modl_ga, then the first sample for every chemical is selected.

The flag param can be used to specify a subset of flags to be used in the flag count. Leaving flag TRUE utilize all the available flags. Setting flag to FALSE will do the subsetting without considering any flags.

Value

A data.table with a single sample for every given chemical-assay pair.

See Also

[tcplPrep0tpt](#)

| | |
|------------|---|
| tcplUpdate | <i>Update assay or chemical information</i> |
|------------|---|

Description

tcplUpdate updates assay or chemical information in the tcpl database.

Usage

```
tcplUpdate(what, id, flds)
```

Arguments

| | |
|------|---------------------------------------|
| what | Character of length 1, what to update |
| id | Integer, the id(s) to update |
| flds | Named list, the values to update |

| | |
|------------|---|
| tcplVarMat | <i>Create a chemical by assay matrix for the given variable</i> |
|------------|---|

Description

tcplVarMat creates a chemical by assay matrix for the given variable. Additional subsetting can be done with the clib, srgx, and argx parameters.

Usage

```
tcplVarMat(var, flag = TRUE, clib = NULL, srgx = NULL, argx = NULL,
  odir = NULL, only.public = TRUE, row.id = "code",
  include.na.chem = FALSE, file.prefix = NULL)
```

Arguments

| | |
|------------------------------|---|
| <code>var</code> | Character, the mc4 or mc5 field to define the matrix values |
| <code>flag</code> | Integer, the mc6_mthd_id values to go into the flag count, see details for more information |
| <code>clib</code> | Character, the chemical libraries to include |
| <code>srgx</code> | Character of length 1, a regular expression to select specific spids |
| <code>argx</code> | Character of length 1, a regular expression to select specific assay names |
| <code>odir</code> | Directory to write a comma separated file |
| <code>only.public</code> | Logical, if TRUE subset the assay endpoints to only those that have an "export_ready" field of 1 |
| <code>row.id</code> | Character, the chemical identifier to use in the output |
| <code>include.na.chem</code> | Logical of length 1, whether to include the chemicals not listed in the tcpl databases (ie. controls) |
| <code>file.prefix</code> | Character of length 1, prefix to the file name when odir is not NULL |

Details

tcplVarMat aggregates multiple samples to a chemical level call after any subsetting. The subsetting by `clib` and `srgx`, along with the chemical/assay mapping, is done by calling [tcplPrep0tpt](#).

When there are still multiple samples after subsetting the data, a single sample is chosen by `tcplSubsetChid` with the given `flag` parameter. See `?tcplSubsetChid` for more details.

By setting `odir` the function will write out a csv with, naming the file with the convention: `paste("AllResults", var,`

The default values for `clib` and `srgx` are NULL. To produce the standard output for the ToxCast program, use values of "toxcast:ph1v2_ph2_e1k", and "(?!^Tox21_1.*)(?!^Tox21_2.*)(?=^.*)", respectively.

`var` accepts two special values. One, 'tested', that will return a matrix indicating if the chemical has ever been tested, whether in single concentration format or multiple concentration format. Two, 'zscore', that will return a list containing the AC50 z score matrix (named 'zscr') based on the burst assays and table with the cytotoxicity distribution for each chemical (named 'zdst').

When a concentration series has a sample id not listed in the tcpl databases, the rowname for that series will be the concatenation of "SPID_" and the spid.

Value

A matrix with one row-per chemical and a column for every assay containing the chemical value for the given field (`var`), or a list where the first element is the zscore matrix and the second element is a `data.table` with the cytotoxicity distribution for each chemical when `var` is 'zscore'.

See Also

[tcplLoadClib](#), [tcplPrep0tpt](#)

| | |
|---------------|---|
| tcplWriteData | <i>Write screening data into the tcpl databases</i> |
|---------------|---|

Description

tcplWriteData takes a data.table with screening data and writes the data into the given level table in the tcpl databases.

Usage

```
tcplWriteData(dat, lvl, type)
```

Arguments

| | |
|------|--|
| dat | data.table, the screening data to load |
| lvl | Integer of length 1, the data processing level |
| type | Character of length 1, the data type, "sc" or "mc" |

Details

This function appends data onto the existing table. It also deletes all the data for any acids or aoids dat contains from the given and all downstream tables.

The data type can be either 'mc' for mutiple concentration data, or 'sc' for single concentration data. Multiple concentration data will be loaded into the level tables, whereas the single concentration will be loaded into the single tables.

Note

This function is not exported and is not inteded to be used by the user. The user should only write level 0 data, which is written with [tcplWriteLv10](#).

See Also

[tcplCascade](#), [tcplAppend](#), [tcplWriteLv10](#)

| | |
|---------------|---|
| tcplWriteLv10 | <i>Write level 0 screening data into the tcpl databases</i> |
|---------------|---|

Description

tcplWriteLv10 takes a data.table with level 0 screening data and writes the data into the level 0 tables in the tcpl databases.

Usage

```
tcplWriteLv10(dat, type)
```

Arguments

| | |
|------|--|
| dat | data.table, the screening data to load |
| type | Character of length 1, the data type, "sc" or "mc" |

Details

This function appends data onto the existing table. It also deletes all the data for any acids or aoids dat contains from the given and all downstream tables.

Before writing any data the function maps the assay component source name(s) (acsn) to assay component id (acid), ensures the proper class on each field and checks for every test compound sample id (spid where wllt == "t") in the tcpl chemical database. If field types get changed a warning is given listing the affected fields and they type they were coerced to. If the acsn(s) or spid(s) do not map to the tcpl databases the function will return an error and the data will not be written.

The data type can be either 'mc' for mutiple concentration data, or 'sc' for single concentration data. Multiple concentration data will be loaded into the level tables, whereas the single concentration will be loaded into the single tables.

Note

This function should only be used to load level 0 data.

See Also

[tcplCascade](#), [tcplAppend](#)

Index

- AIC, [25](#)
- blinShift, [3](#)

- constrOptim, [16, 29](#)

- data.table, [30–33, 35, 39, 41](#)
- dbConnect, [39, 41](#)

- flareFunc, [3](#)

- interlaceFunc, [4](#)
- is.odd, [5, 5, 6, 23](#)

- length, [6](#)
- lu, [5, 5, 6, 23](#)
- lw, [5, 6, 23](#)

- mc1, [6, 7, 9, 12–14](#)
- mc2, [7, 7, 8, 9, 12–14](#)
- MC2_Methods, [7, 8](#)
- mc2_mthds (MC2_Methods), [8](#)
- mc3, [3, 7, 9, 11–14](#)
- MC3_Methods, [9, 9](#)
- mc3_mthds, [3](#)
- mc3_mthds (MC3_Methods), [9](#)
- mc4, [7, 9, 12, 13, 14](#)
- mc5, [7, 9, 12, 12, 14](#)
- MC5_Methods, [13, 13](#)
- mc5_mthds (MC5_Methods), [13](#)
- mc6, [4, 7, 9, 12, 13, 14, 16](#)
- MC6_Methods, [14, 15](#)
- mc6_mthds, [4](#)
- mc6_mthds (MC6_Methods), [15](#)
- mc1apply, [40](#)
- Models, [12, 16, 24](#)

- registerMthd, [18](#)

- sc1, [18, 20, 21](#)
- SC1_Methods, [19, 19](#)
- sc1_mthds (SC1_Methods), [19](#)
- sc2, [19, 21, 22](#)
- SC2_Methods, [21, 21](#)
- sc2_mthds (SC2_Methods), [21](#)

- sink, [23](#)
- sink.number, [23](#)
- sink.reset, [5, 6, 22](#)

- tcplACVal, [23, 24](#)
- tcplACXX, [23, 23](#)
- tcplAddModel, [24](#)
- tcplAICProb, [25](#)
- tcplAppend, [25, 45, 46](#)
- tcplAssignMthd, [26, 27, 29, 35](#)
- tcplCascade, [6, 7, 9, 12, 14, 18, 21, 26, 45, 46](#)
- tcplClearMthd, [26, 27, 29, 35](#)
- tcplCode2CASN, [27](#)
- tcplDelete, [28](#)
- tcplFit, [12, 16, 25, 28](#)
- tcplListFlds, [29](#)
- tcplListMthd, [26, 27, 29, 35](#)
- tcplListOpts, [30, 41](#)
- tcplLoadAcid, [30](#)
- tcplLoadAeid, [31](#)
- tcplLoadAeidInfo, [31](#)
- tcplLoadAid, [32](#)
- tcplLoadAsid, [32](#)
- tcplLoadAsidInfo, [33](#)
- tcplLoadChem, [33](#)
- tcplLoadClib, [34, 44](#)
- tcplLoadData, [34, 37](#)
- tcplLoadMthd, [4, 7–9, 11, 13, 14, 16, 19–22, 26, 27, 29, 35](#)
- tcplLoadUnit, [35](#)
- tcplMakeAeidPlts, [36](#)
- tcplObjCnst, [29](#)
- tcplObjCnst (Models), [16](#)
- tcplObjGnls, [29](#)
- tcplObjGnls (Models), [16](#)
- tcplObjHill, [29](#)
- tcplObjHill (Models), [16](#)
- tcplPlotFitc, [36](#)
- tcplPlotFits, [24, 37, 37](#)
- tcplPlotM4ID, [37](#)
- tcplPlotPlate, [38](#)
- tcplPrepOtppt, [38, 43, 44](#)
- tcplQuery, [30–33, 35, 39](#)
- tcplRegister, [39](#)

tcplRun, [6](#), [7](#), [9](#), [12](#), [14](#), [18](#), [21](#), [40](#)

tcplSendQuery, [28](#), [41](#)

tcplSetOpts, [30](#), [41](#)

tcplSpidMap, [42](#)

tcplSubsetChid, [42](#)

tcplUpdate, [43](#)

tcplVarMat, [43](#)

tcplWriteData, [45](#)

tcplWriteLvl0, [45](#), [45](#)

unique, [5](#)

which, [5](#), [6](#)