

# Package: tcpl (via r-universe)

December 9, 2024

**Title** ToxCast Data Analysis Pipeline

**Version** 3.2.0

**Description** The ToxCast Data Analysis Pipeline ('tcpl') is an R package that manages, curve-fits, plots, and stores ToxCast data to populate its linked MySQL database, 'invitrodb'. The package was developed for the chemical screening data curated by the US EPA's Toxicity Forecaster (ToxCast) program, but 'tcpl' can be used to support diverse chemical screening efforts.

**URL** <https://github.com/USEPA/CompTox-ToxCast-tcpl>,  
<https://www.epa.gov/comptox-tools/toxicity-forecasting-toxcast>

**Depends** R (>= 3.5.0)

**Imports** data.table (>= 1.15.0), DBI, RMariaDB, numDeriv, RColorBrewer, utils, stats, methods, graphics, grDevices, sqldf, dplyr, tidyr, plotly, tcplfit2, ggplot2, gridExtra, stringr, rlang, ctxR

**Suggests** roxygen2, knitr, prettydoc, rmarkdown, htmlTable, testthat (>= 3.0.0), reshape2, viridis, kableExtra, colorspace, magrittr, vdiff, httpptest, rmdformats

**License** MIT + file LICENSE

**LazyData** true

**RoxygenNote** 7.3.1

**VignetteBuilder** knitr

**Encoding** UTF-8

**Config/testthat/edition** 3

**BugReports** <https://github.com/USEPA/CompTox-ToxCast-tcpl/issues>

**NeedsCompilation** no

**Author** Richard S Judson [ctb, ths]  
(<<https://orcid.org/0000-0002-2348-9633>>), Dayne L Filer [aut],  
Jason Brown [cre] (<<https://orcid.org/0009-0000-2294-641X>>),

Sarah E Davidson-Fritz [ctb]  
 (<<https://orcid.org/0000-0002-2891-9380>>), Madison Feshuk [ctb]  
 (<<https://orcid.org/0000-0002-1390-6405>>), Carter Thunes [ctb],  
 Ashley Ko [ctb], Kelly Carstens [ctb]  
 (<<https://orcid.org/0000-0002-1746-5379>>), Todd Zurlinden [ctb]  
 (<<https://orcid.org/0000-0003-1372-3913>>), Lori Kolaczowski  
 [ctb], Kurt Dunham [ctb], Parth Kothiya [ctb], Woodrow R Setzer  
 [ctb], Matthew T Martin [ctb, ths], Katie Paul Friedman [ctb]  
 (<<https://orcid.org/0000-0002-2710-1691>>)

**Maintainer** Jason Brown <brown.jason@epa.gov>

**Repository** CRAN

**Date/Publication** 2024-10-09 18:10:02 UTC

**Config/pak/sysreqs** make libicu-dev libmysqlclient-dev libssl-dev

## Contents

.buildAssayQ . . . . .	4
.convertNames . . . . .	5
.load6DR . . . . .	5
.plateHeat . . . . .	6
.prepField . . . . .	6
blinShift . . . . .	7
check_tcp1_db_schema . . . . .	7
Configure functions . . . . .	8
flareFunc . . . . .	9
Hill model utilites . . . . .	10
interlaceFunc . . . . .	11
invitrodb_dd . . . . .	12
is.odd . . . . .	12
Load assay information . . . . .	13
load_data_columns . . . . .	14
lu . . . . .	15
lw . . . . .	16
mc1 . . . . .	16
mc2 . . . . .	17
MC2_Methods . . . . .	18
mc3 . . . . .	19
MC3_Methods . . . . .	20
mc4 . . . . .	24
MC4_Methods . . . . .	25
mc5 . . . . .	26
MC5_Methods . . . . .	27
mc6 . . . . .	29
MC6_Methods . . . . .	30
mcdat . . . . .	31
mc_test . . . . .	32

mc_vignette . . . . .	35
Method functions . . . . .	43
Models . . . . .	44
mthd_list_defaults . . . . .	47
Query functions . . . . .	47
Register/update annotation . . . . .	49
registerMthd . . . . .	50
sc1 . . . . .	51
SC1_Methods . . . . .	52
sc2 . . . . .	54
SC2_Methods . . . . .	55
scdat . . . . .	57
sc_test . . . . .	57
sc_vignette . . . . .	59
sink.reset . . . . .	61
tcplAddModel . . . . .	62
tcplAICProb . . . . .	63
tcplAppend . . . . .	64
tcplCascade . . . . .	64
tcplCode2CASN . . . . .	65
tcplCytoPt . . . . .	66
tcpldbStats . . . . .	68
tcplDefine . . . . .	68
tcplDelete . . . . .	69
tcplFit . . . . .	70
tcplFit2 . . . . .	71
tcplFit2_nest . . . . .	72
tcplFit2_unnest . . . . .	72
tcplGetAeid . . . . .	73
tcplggplot . . . . .	73
tcplggplotCompare . . . . .	74
tcplHit2 . . . . .	75
tcplLegacyPlot . . . . .	75
tcplListFlds . . . . .	76
tcplLoadChem . . . . .	76
tcplLoadChemList . . . . .	77
tcplLoadConcUnit . . . . .	79
tcplLoadData . . . . .	79
tcplLoadUnit . . . . .	81
tcplLvlCount . . . . .	81
tcplMakeAeidMultiPlts . . . . .	82
tcplMakeAeidPlts . . . . .	83
tcplMakeChidMultiPlts . . . . .	84
tcplMultiplot . . . . .	85
tcplPlot . . . . .	86
tcplPlotFitc . . . . .	87
tcplPlotFits . . . . .	88
tcplPlotLoadData . . . . .	89

tcplPlotlyPlot . . . . .	91
tcplPlotM4ID . . . . .	91
tcplPlotPlate . . . . .	92
tcplPlotSetYRange . . . . .	93
tcplPlotValidate . . . . .	94
tcplPrepOtp . . . . .	94
tcplRun . . . . .	95
tcplSubsetChid . . . . .	96
tcplVarMat . . . . .	98
tcplWriteData . . . . .	100
tcplWriteLvl0 . . . . .	100
test_api . . . . .	101
write_lvl_4 . . . . .	102

**Index** **103**

---

<i>.buildAssayQ</i>	<i>Generate query for assay information</i>
---------------------	---

---

**Description**

*.buildAssayQ* generates a query string to load assay information

**Usage**

```
.buildAssayQ(out, tblo, fld = NULL, val = NULL, add.fld = NULL)
```

**Arguments**

<i>out</i>	Character, the default fields to include
<i>tblo</i>	Integer, the order to send the fields to prepOutput
<i>fld</i>	Character, the field(s) to query/subset on
<i>val</i>	List, vectors of values for each field to query/subset on. Must be in the same order as 'fld'.
<i>add.fld</i>	Character, additional field(s) to include, but not query/ subset on

**Value**

A character containing the query to send to tcplQuery

---

.convertNames                      *Convert assay names to their abbreviations*

---

### **Description**

.convertNames converts the assay names as they appear in the tcpl database to their respective abbreviations

### **Usage**

.convertNames(names)

### **Arguments**

names                      Character, strings to convert

### **Value**

The same character vector given with any name strings converted to the abbreviated version

---

.load6DR                      *Load data for tcpl6*

---

### **Description**

.load6DR loads dose-response data for tcpl6.

### **Usage**

.load6DR(ae)

### **Arguments**

ae                      String aeid to query on

---

`.plateHeat`                      *Plot plate heatmap*

---

### Description

Plot plate heatmap, to be used with `tcplPlotPlate`

### Usage

`.plateHeat(vals, rowi, coli, wllt, wllq, rown, coln, main, arng)`

### Arguments

<code>vals</code>	Numeric, the well values
<code>rowi</code>	Integer, the row index
<code>coli</code>	Integer, the column index
<code>wllt</code>	Character, the well type
<code>wllq</code>	Logical, the well quality
<code>rown</code>	Integer, the number of rows on the plate
<code>coln</code>	Integer, the number of columns on the plate
<code>main</code>	Character of length 1, the title/main
<code>arng</code>	Numeric of length 2, the minimum and maximum values to constrain the color scale

### Note

Optimized for an output with height = 20/3, width = 10, and pointsize = 10

---

`.prepField`                      *Paste appropriate table name to field name*

---

### Description

Paste appropriate table name to field name

### Usage

`.prepField(fld, tbl, db)`

### Arguments

<code>fld</code>	Character, the table fields
<code>tbl</code>	Character, the possible tables
<code>db</code>	Character, the database containing the tables

**Details**

The function loops through the given tables, and for each field *i* it assigns the last table containing *i* to *i*. ORDER OF FLD MATTERS!!

---

blineShift	<i>Shift the baseline to 0</i>
------------	--------------------------------

---

**Description**

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

**Usage**

```
blineShift(resp, conc, wndw)
```

**Arguments**

resp	Numeric, the response values
conc	Numeric, the concentration values
wndw	Numeric, the threshold window

**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](#)

---

check_tcp1_db_schema	<i>Function that checks if the most recent v3 table schema is used in the database schema</i>
----------------------	---

---

**Description**

Function that checks if the most recent v3 table schema is used in the database schema

**Usage**

```
check_tcp1_db_schema()
```

**Value**

boolean TRUE if param tables are listed in schema FALSE otherwise

**Examples**

```
## Not run:
#connect to database first with tcplConf
tcplConf(user=user,
  pass= pass,
  db=dbname,
  drvr='MySQL',
  host=hostname)

#check if it is part of the new schema
new_schema <- check_tcpl_db_schema()

## End(Not run)
```

---

Configure functions      *Functions for configuring the tcpl package*

---

**Description**

These functions are used to configure the tcpl settings.

**Usage**

```
tcplConf(drvr = NULL, user = NULL, pass = NULL, host = NULL, db = NULL, ...)

tcplConfDefault()

tcplConfList(show.pass = FALSE)

tcplConfLoad(list.new = TRUE)

tcplConfReset()

tcplConfSave()
```

**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



...	Additional arguments that should be passed to dbConnect function
show.pass	Logical, should the password be returned
list.new	Logical of length 1, should the new settings be printed?

## Details

Currently, the tcpl package supports the "MySQL", "example", and "API" database drivers.

The settings can be stored in a configuration file to make the using the package more user-friendly. To create the configuration file, the user must first create a system environment variable ('TCPL\_CONF') that points to the file. There is more information about system environment variables in [Startup](#) and [Sys.getenv](#). Briefly, the user needs to modify the '.Renviro' file in their home directory. If the file does not exist, create it, and add the following line:

```
TCPL_CONF=path/to/confFile.conf
```

Here 'path/to/confFile.conf' can be any path to a file. One suggestion would be to include .tcplConf in the home directory, e.g. TCPL\_CONF=~/.tcplConf. Note, '~' may not indicate the home directory on every operating system. Once the environment variable is added, the user can change the settings using tcplConf, then save the settings to the file given by the TCPL\_CONF environment variable running tcplConfSave().

tcplConf changes options to set the tcpl-specific options, most importantly to configure the connection to the tcpl databases. tcplConf will only change non-null values, and can be used to change a single value if needed.

tcplConfSave modifies the configuration file to reflect the current tcpl settings.

tcplConfList lists the values assigned to the tcpl global options.

tcplConfLoad updates the tcpl settings to reflect the current configuration file.

tcplConfDefault changes the options to reflect the default settings for the API connection, but does not alter the configuration file.

tcplConfReset is used to generate the initial configuration script, and can be used to reset or regenerate the configuration script by the user.

---

flareFunc

*Calculate the weighted mean of a square to detect plate flares*

---

## 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\\_Methods](#), [Method functions](#), [mc6](#)

---

Hill model utilites    *Functions to solve the Hill model*

---

**Description**

These functions solve for Hill model parameters.

**Usage**

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

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

```
tcplHillVal(logc, tp, ga, gw, bt = 0)
```

**Arguments**

XX	Numeric, the activity level (percentage of the top 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
val	Numeric, the activity value
logc	Numeric, the log concentration

**Details**

tcplHillVal computes the value of the Hill model for a given log concentration.

tcplHillACXX computes the activity concentration for a Hill model for a given activity level.

tcplHillConc computes the Hill model concentration for a given value.

**Examples**

```
## The following code gives examples for a Hill model with a top of 50,
## bottom of 0, AC50 of 1 and Hill coefficient of 1.
## tcplHillVal calculates activity value given a concentration. tcplHillVal
## will return the tp/2 when logc equals ga:
tcplHillVal(logc = 1, tp = 50, ga = 1, gw = 1, bt = 0)

## Here, tcplHillConc returns the concentration where the value equals 20
tcplHillConc(val = 20, tp = 50, ga = 1, gw = 1, bt = 0)

## Note how this differs from tcplHillACXX:
tcplHillACXX(XX = 20, tp = 50, ga = 1, gw = 1, bt = 0)

## tcplHillACXX is based on the top value and allows the user to calculate
## specific activity concentrations based on a percentage of the top value

## For example, we can calculate the value for the concentration 0.25, then
## use that value to check the other two functions.

value <- tcplHillVal(logc = 0.25, tp = 50, ga = 1, gw = 1, bt = 0)
c1 <- tcplHillConc(val = value, tp = 50, ga = 1, gw = 1, bt = 0)
c2 <- tcplHillACXX(XX = value/50*100, tp = 50, ga = 1, gw = 1, bt = 0)
all.equal(0.25, c1, c2)

## Notice, the value had to be transformed to a percentage of the top value
## when using tcplHillACXX
```

---

interlaceFunc

*Calculate the weighted mean of a square to detect interlace effect*


---

**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\\_Methods](#), [Method functions](#), [mc6](#)

---

invitrodb_dd	<i>Short descriptions of fields for different tables are stored in a data dictionary.</i>
--------------	---

---

**Description**

Short descriptions of fields for different tables are stored in a data dictionary.

**Usage**

```
invitrodb_dd
```

**Format**

A data frame with 44 rows and 3 variables:

**invitrodb\_table** Table of the data dictionary

**invitrodb\_field** Field of the data dictionary

**description** Description

**Source**

ToxCast database

---

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()`

---

Load assay information

*Functions for loading assay information*

---

**Description**

These functions query the tcpl databases and returns a `data.table` with assay ID and name information. More information about the assay hierarchy is available in the overview vignette.

**Usage**

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

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

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

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

**Arguments**

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

**Details**

Each element in the assay hierarchy has its own function, loading the ID and name for the given assay element. For example, `tcplLoadAsid` will return the assay source ID (`asid`) and assay source name (`asnm`).

**Value**

A `data.table` containing the ID, name, and any additional fields.

**Examples**

```
## Not run:  
## The load assay functions can be used without any parameters to list the  
## full list of registered assay elements:  
tcplLoadAsid()  
tcplLoadAeid()
```

```

## Similarly, the user can add fields without doing any element selection:
tcpLoadAeid(add.fld = c("asid", "aid", "acid"))

## Or, the user can look only at a subset:
tcpLoadAeid(fld = "aeid", val = 1, add.fld = "asid")

## The field can be any value in one of the corresponding assay element
## tables, but the functions also recognize the abbreviated version of
## the name fields.
tcpListFlds("assay")
a1 <- tcpLoadAeid(fld = "anm", val = "Steroidogenesis")
a2 <- tcpLoadAeid(fld = "assay_name", val = "Steroidogenesis")
identical(a1, a2)

## End(Not run)

```

---

load_data_columns	<i>Lists of column names returned from tcpLoadData invitrodb v4.1 (same as CCTE Bioactivity API version).</i>
-------------------	---

---

## Description

Lists of column names returned from tcpLoadData invitrodb v4.1 (same as CCTE Bioactivity API version).

## Usage

```
load_data_columns
```

## Format

A list with 12 items:

- mc0** Column names returned requesting mc lvl 0 data
- mc1** Column names returned requesting mc lvl 1 data
- mc2** Column names returned requesting mc lvl 2 data
- mc3** Column names returned requesting mc lvl 3 data
- mc4** Column names returned requesting mc lvl 4 data
- mc5** Column names returned requesting mc lvl 5 data
- mc6** Column names returned requesting mc lvl 6 data
- mcagg** Column names returned requesting mc lvl "agg" data
- sc0** Column names returned requesting sc lvl 0 data
- sc1** Column names returned requesting sc lvl 1 data
- sc2** Column names returned requesting sc lvl 2 data
- scagg** Column names returned requesting sc lvl "agg" data

**Source**

ToxCast database

---

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

---

**Description**

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

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

**Usage**

```
lu(x)
```

```
lu(x)
```

**Arguments**

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

**Value**

The unique of the TRUE values in x

The unique of the TRUE values in x

**See Also**

[unique, which](#)

[unique, which](#)

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

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)).

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

**Usage**

```
lw(x)
```

```
lw(x)
```

**Arguments**

x                    A logical

**Value**

The length of the TRUE values in x

The length of the TRUE values in x

**See Also**

[length, which](#)

[length, which](#)

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

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: [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**

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 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**

[Method functions, MC2\\_Methods](#)

Other multiple-concentration: [mc1\(\)](#), [mc3\(\)](#), [mc4\(\)](#), [mc5\(\)](#), [mc6\(\)](#)

---

MC2\_Methods

*List of level 2 multiple-concentration correction functions*

---

**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, "Data\_processing."

**Correction Methods:**

**log2** Transform the corrected response value (cval) to log-scale (base 2).

**log10** Transform the corrected response value (cval) to log-scale (base 10).

**rmneg** Exclude wells with negative corrected response values (cval) and downgrading their well quality (wllq); if  $cval < 0$ ,  $wllq = 0$ .

**rmzero** Exclude wells with corrected response values (cval) equal to zero and downgrading their well quality (wllq); if  $cval = 0$ ,  $wllq = 0$ .

- mult25** Multiply corrected response value (*cval*) by 25;  $25 * cval$ .
- mult100** Multiply corrected response value (*cval*) by 100;  $100 * cval$ .
- negshift** Shift corrected response values (*cval*) by subtracting the minimum *cval* and adding 1, such that the new minimum is 1;  $cval - min + 1$ .
- mult2.5** Multiply corrected response value (*cval*) by 2.5;  $2.5 * cval$ .
- mult3** Multiply corrected response value (*cval*) by 3;  $3 * cval$ .
- mult6** Multiply corrected response value (*cval*) by 6;  $6 * cval$ .
- sub100** Center data around zero by subtracting the corrected response value (*cval*) from 100;  $100 - cval$ . Typically used if data was pre-normalized around 100 with responses decreasing to 0.
- zscore.npwlls** Convert the corrected response value (*cval*) to an absolute Z-Score based on the neutral and positive control wells (*wllts* = *n* and *p*), by assay plate ID (*apid*)
- sub1** Center data around zero by subtracting the corrected response value (*cval*) from 1;  $1 - cval$ . Typically used if data was pre-normalized around 1 with responses decreasing to 0.

#### Aggregation Methods:

- agg.mean.rep.apid** Aggregate technical test replicates (*wllt*=*t*) by taking the plate-wise mean per sample id (*spid*), assay plate (*apid*), and concentration index (*cndx*).
- agg.median.rep.apid** Aggregate technical test replicates (*wllt*=*t*) by taking the plate-wise median per sample id (*spid*), assay plate (*apid*), and concentration index (*cndx*).
- agg.percent.rep.spid** Use for binary data. Aggregate technical replicates as percentage by taking the sum of *rval* (raw values) relative to total replicates per sample id (*spid*) and concentration index (*cndx*);  $cval = (sum(rval)/.N) * 100$ .
- agg.percent.rep.spid.min1** Use for binary data with variable number of replicates. Aggregate technical replicates as percentage by taking the sum of *rval* (raw values) relative to total replicates per per sample id (*spid*) and concentration index (*cndx*), where there is more than one replicate;  $cval = (sum(rval)/.N) * 100$ , where  $.N > 1$ . *Rvals* are collapsed to one value per *cndx*.

#### Note

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

#### See Also

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

#### 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**

[Method functions, MC3\\_Methods](#)

Other multiple-concentration: `mc1()`, `mc2()`, `mc4()`, `mc5()`, `mc6()`

---

MC3\_Methods

*List of level 3 multiple-concentration normalization methods*

---

**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 `aeids` (a numeric vector of `aeid` 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 acid.

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

**bval Methods:**

**bval.apid.nwlls.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for neutral control wells (wlft = n).

**bval.apid.lowconc.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for test compound wells (wlft = t) with a concentration index (cndx) of 1 or 2.

**bval.apid.twlls.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) of test compound wells (wlft = t).

**bval.apid.tn.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for test compound wells (wlft = t) and neutral control wells (wlft = n).

**bval.apid.nwllslowconc.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) of test compound wells (wlft = t) with a concentration index (cndx) of 1 or 2 or neutral control wells (wlft = n).

**bval.spid.lowconc.med** Calculate the baseline value (bval) as the sample-wise median, by sample ID (spid), of the corrected values (cval) of the three lowest concentration test compound wells (wlft = t and cndx = 1, 2, & 3).

**bval.apid.nwllstcwllslowconc.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for neutral control wells (wlft = n) or wells with a concentration index (cndx) of 1 or 2 and well type of test compound (wlft = t) or gain-of-signal control in multiple concentrations (wlft = c).

**bval.aeid.nwlls.med** Calculate the baseline value (bval) as the endpoint-wise median, by assay component endpoint ID (aeid), corrected value (cval) for neutral control wells (wlft = n).

**pval Methods:**

**pval.apid.pwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for single-concentration gain-of-signal positive control wells (wlft = p).

**pval.apid.mwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for multiple-concentration loss-of-signal negative control wells (wlft = m).

**pval.apid.medpchyconc.max** Calculate the positive control value (pval) as the plate-wise maximum, by assay plate ID (apid), of the medians of the corrected values (cval) for gain-of-signal single- or multiple-concentration negative control wells (wlft = m or o) by apid, well type, and concentration.

- pval.apid.medpcbyconc.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of corrected value (cval) of gain-of-signal single- or multiple-concentration positive control wells (wllt = p or c) by apid, well type, and concentration.
- pval.apid.medncbyconc.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of the corrected values (cval) for gain-of-signal single- or multiple-concentration negative control wells (wllt = m or o) by apid, well type, and concentration.
- pval.apid.pmv.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of the corrected values (cval) for single-concentration gain-of-signal, multiple-concentration loss-of-signal, or viability control wells (wllt = p, m, or v) by apid, well type, and concentration.
- pval.apid.pmv.max** Calculate the positive control value (pval) as the plate-wise maximum, by assay plate ID (apid), of the medians of the corrected values (cval) for single-concentration gain-of-signal, multiple-concentration loss-of-signal, or viability control wells (wllt = p, m, or v) by apid, well type, and concentration.
- pval.apid.f.max** Calculate the positive control value (pval) as the plate-wise maximum, by assay plate ID (apid), of the medians of important reference wells (wllt = f) values by apid and concentration.
- pval.apid.f.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of important reference wells (wllt = f) values by apid and concentration.
- pval.apid.p.max** Calculate the positive control value (pval) as the plate-wise maximum, by assay plate ID (apid), of the medians of the corrected values (cval) for single-concentration gain-of-signal control wells (wllt = p) by apid.
- pval.apid.p.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of corrected values (cval) for single-concentration gain-of-signal control wells (wllt = p) by apid.
- pval.apid.v.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of the corrected values (cval) for viability control wells (wllt = v) by apid and concentration.
- pval.zero** Set the positive control value (pval) to 0;  $pval = 0$ .
- pval.apid.owlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for single-concentration negative control wells (wllt = o).
- pval.2bval** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for neutral control wells (wllt = n) multiplied by 2.
- pval.maxp** Calculate the positive control value (pval) as the endpoint-wise maximum, by assay component ID (aeid), of the corrected values for single-concentration gain-of-signal wells (wllt = p).
- pval.apid.bwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the corrected values (cval) for blank wells (wllt = b).
- pval.twlls.99pct** Calculate positive control value (pval) as the 99th percentile of all corrected value (cvals) of the test compound wells (wllt = t).
- pval.neg.100** Calculate positive control value (pval) as -100 for endpoints in the down direction;  $pval = -100$ .

**resp Methods:**

- resp.pc** Calculate the normalized response (*resp*) as a percent of control, i.e. the ratio of the difference between the corrected (*cval*) and baseline (*bval*) values divided the difference between the positive control (*pval*) and baseline (*bval*) values multiplied by 100;  $resp = (cval - bval)/(pval - bval) * 100$ .
- resp.pc.pval.cor** Calculate the normalized response (*resp*) as a percent of control, i.e. the ratio of the difference between the corrected (*cval*) and baseline (*bval*) values divided the positive control (*pval*) value multiplied by 100;  $resp = (cval - bval)/pval * 100$ .
- resp.fc** Calculate the normalized response (*resp*) as the fold change, i.e. the ratio of the corrected (*cval*) and baseline (*bval*) values;  $resp = cval/bal$ .
- resp.logfc** Calculate the normalized response (*resp*) as the fold change of logged, i.e. the difference between corrected (*cval*) and baseline (*bval*) log-scale values.
- resp.log2** Transform the response values to log-scale (base 2).
- resp.mult25** Multiply the normalized response value (*resp*) by 25;  $25 * resp$ .
- resp.scale.mad.log2fc** Scale the normalized response value (*resp*) by the ratio of  $\log_2(1.2)$  and 3 multiplied by the baseline median absolute deviation (*bmad*) of the unscaled normalized response values (*resp*);  $(\log_2 1.2)/3 * bmad * resp$ .
- resp.scale.quant.log2fc** Scale the normalized response value (*resp*). First, determine the maximum difference (*md*) by finding the maximum between the absolute difference of the 1st percentile minus the 50th percentile and the absolute difference of the 99th percentile minus the 50th percentile. Then multiply *resp* by  $\log_2(1.2)$  divided by 20 percent of *md*;  $(\log_2 1.2)/0.2 * md * resp$ .
- resp.multneg1** Multiply the normalized response value (*resp*) by -1;  $-1 * resp$ .
- resp.shiftneg.3bmad** Shift all the normalized response values (*resp*) less than -3 multiplied by the baseline median absolute deviation (*bmad*) to 0; if  $resp < -3 * bmad$ ,  $resp = 0$ .
- resp.shiftneg.6bmad** Shift all the normalized response values (*resp*) less than -6 multiplied by the baseline median absolute deviation (*bmad*) to 0; if  $resp < -6 * bmad$ ,  $resp = 0$ .
- resp.shiftneg.10bmad** Shift all the normalized response values (*resp*) less than 10 multiplied by the baseline median absolute deviation (*bmad*) to 0; if  $resp < -10 * bmad$ ,  $resp = 0$ .
- resp.blineshift.3bmad.repi** Shift the normalized response value (*resp*) with a baseline correction, by replicate index (*repi*), with a window of 3 multiplied by the baseline median absolute deviation (*bmad*).
- resp.blineshift.50.repi** Shift the normalized response value (*resp*) with a baseline correction, by replicate index (*repi*), with a window of 50.
- resp.blineshift.3bmad.spid** Shift the normalized response value (*resp*) with a baseline correction, by sample ID (*spid*), with a window of 3 multiplied by the baseline median absolute deviation (*bmad*).
- resp.blineshift.50.spid** Shift the normalized response value (*resp*) with a baseline correction, by sample ID (*spid*), with a window of 50.
- none** Set the corrected response value (*cval*) as the normalized response value (*resp*);  $cval = resp$ . No additional mc3 methods needed for endpoint-specific normalization.
- resp.zerocenter.fc** Calculate the normalized response (*resp*) as a zero center fold change, i.e. 1 minus the ratio of corrected (*cval*) and baseline (*bval*) values;  $resp = 1 - cval/bval$ . Typically used for increasing responses.
- resp.incr.zerocenter.fc** Calculate the normalized response (*resp*) as a zero center fold change, i.e. the ratio of the the corrected (*cval*) and baseline (*bval*) values minus 1;  $resp = cval/bval - 1$ . Typically used for increasing responses.

**resp.mult100** Multiply the normalized response value (resp) by 100;  $100 * resp$ .

### Note

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

### See Also

[mc3](#), [Method functions](#) to query what methods get applied to each aeid

---

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: [mc1\(\)](#), [mc2\(\)](#), [mc3\(\)](#), [mc5\(\)](#), [mc6\(\)](#)



**Description**

mc4\_mthds returns a list of methods to be used during level 4 multiple-concentration processing for calculating bmad

**Usage**

```
mc4_mthds()
```

**Details**

The functions contained in the list returned by mc4\_mthds take aeids (a numeric vector of aeid values) and returns a list of expressions to be executed in the mc4 (not exported) function environment. The functions are described here for reference purposes, The mc4\_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**

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

More information about the level 4 multiple-concentration processing is available in the package vignette, "Data\_processing."

**bmad.aeid.lowconc.twells** Calculate the baseline median absolute value (bmad) as the median absolute deviation of normalized response values (rep) for test compound wells (wlft = t) with concentration index (cndx) equal to 1 or 2. Calculate one standard deviation of the normalized response for test compound wells (wlft = t) with a concentration index (cndx) of 1 or 2; onesd =  $\sqrt{\text{sum}((\text{resp} - \text{mean resp})^2)/\text{sample size} - 1}$ . Onesd is used to establish BMR and therefore required for tcplfit2 processing.

**bmad.aeid.lowconc.nwells** Calculate the baseline median absolute value (bmad) as the median absolute deviation of normalized response values (resp) for neutral control wells (wlft = n). Calculate one standard deviation of the normalized response for neutral control wells (wlft = n); onesd =  $\sqrt{\text{sum}((\text{resp} - \text{mean resp})^2)/\text{sample size} - 1}$ . Onesd is used to establish BMR and therefore required for tcplfit2 processing.

**bidirectional.false** Limits bidirectional fitting and processes data in positive analysis direction only. Use for gain-of-signal or inverted data.

**bm5.onesd16.static** Replace baseline median absolute deviation (bmad) with 5 and one standard deviation (osd) of the normalized response for test compound wells (wlft = t) with a concentration index (cndx) of 1 or 2 with 16. Typically used for binary data where values would otherwise be 0; non-zero values are required for tcplfit2 processing.

**no.unbounded.models** Exclude unbounded models and only fit data to bounded models (hill, gnls, exp4 and exp5).

### Note

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

### See Also

[mc4](#), [Method functions](#) to query what methods get applied to each aeid

---

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.

### 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

[Method functions](#), [MC5\\_Methods](#)

Other multiple-concentration: [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(ae)
```

### Arguments

ae                      Integer of length 1, the assay endpoint id

### Details

The functions contained in the list returned by mc5\_mthds take aeids (a numeric vector of aeid values) and returns a list of expressions to be executed in the mc5 (not exported) function environment. The functions are described here for reference purposes, The mc5\_mthds function is not exported, nor is it intended for use.

All available methods are described in the "Available Methods" section, listed by the cutoff type in ascending order of cutoff value.

### Value

A list of functions

### Available Methods

The methods are broken down into five categories based on the type of cutoff they assign. Different methods are used to define cutoffs for "bmad" (baseline median absolute value), "fc" (fold change), "log" ( $\log_2$  or  $\log_{10}$ ), "pc" (percent of control), and "other" (uncategorized cutoffs).

All methods are applied by aeid.

Although there are method exceptions (notably within the "other" category), only highest calculated cutoff value based on assigned methods will be selected for hitcalling. Therefore, only the largest cutoff method per method type should be assigned.

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

#### **BMAD Methods:**

**bmad1** Add a cutoff value of 1 multiplied by baseline median absolute value (bmad). By default, bmad is calculated using test compound wells (wlIt = t) for the endpoint.

**bmad2** Add a cutoff value of 2 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlIt = t) for the endpoint.

**bmad3** Add a cutoff value of 3 multiplied by the baseline median absolute deviation (bmad) as defined at Level 4.

**bmad4** Add a cutoff value of 4 multiplied the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlit = t) for the endpoint.

**bmad5** Add a cutoff value of 5 multiplied the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlit = t) for the endpoint.

**bmad6** Add a cutoff value of 6 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlit = t) for the endpoint.

**bmad10** Add a cutoff value of 10 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlit = t) for the endpoint.

#### **Fold Change Methods:**

**fc0.2** Add a cutoff value of 0.2. Typically for zero centered fold change data.

**fc0.25** Add a cutoff value of 0.25. Typically for zero centered fold change data.

**fc0.3** Add a cutoff value of 0.3. Typically for zero centered fold change data.

**fc0.5** Add a cutoff value of 0.5. Typically for zero centered fold change data.

#### **Log Methods: Log Base 2**

**neglog2\_0.88** Add a cutoff value of  $-\log_2 0.88$ .

**log2\_1.2** Add a cutoff value of  $\log_2 1.2$ . Typically for fold change data.

**log2\_2** Add a cutoff value  $\log_2 2$ . Typically for fold change data.

#### **Log Base 10**

**log10\_1.2** Add a cutoff value of  $\log_{10} 1.2$ . Typically for fold change data.

**log10\_2** Add a cutoff value of  $\log_{10} 2$ . Typically for fold change data.

#### **Percent of Control Methods:**

**pc05** Add a cutoff value of 5. Typically for percent of control data.

**pc10** Add a cutoff value of 10. Typically for percent of control data.

**pc16** Add a cutoff value of 16. Typically for percent of control data.

**pc20** Add a cutoff value of 20. Typically for percent of control data.

**pc25** Add a cutoff value of 25. Typically for percent of control data.

**pc30** Add a cutoff value of 30. Typically for percent of control data.

**pc40** Add a cutoff value of 40. Typically for percent of control data.

**pc50** Add a cutoff value of 50. Typically for percent of control data.

**pc70** Add a cutoff value of 70. Typically for percent of control data.

**pc95** Add a cutoff value of 95. Typically for percent of control data.

#### **Other Methods:**

**maxmed20pct** Add a cutoff value of 20 percent of the maximum of all endpoint maximal average response values (max\_med).

**coff\_2.32** Add a cutoff value of 2.32.

**loec.coff** Method not yet updated for tcpl implementation. Identify the lowest observed effective concentration (loec) compared to baseline.

**ow\_bidirectional\_loss** Multiply winning model hitcall (hitc) by -1 for models fit in the positive analysis direction. Typically used for endpoints where only negative responses are biologically relevant.

**ow\_bidirectional\_gain** Multiply winning model hitcall (hitc) by -1 for models fit in the negative analysis direction. Typically used for endpoints where only positive responses are biologically relevant.

**osd\_coff\_bmr** Overwrite the osd value so that bmr equals cutoff.

### Note

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

### See Also

[mc5](#), [Method functions](#) 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 [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

```
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**

[Method functions](#), [MC6\\_Methods](#)

Other multiple-concentration: [mc1\(\)](#), [mc2\(\)](#), [mc3\(\)](#), [mc4\(\)](#), [mc5\(\)](#)

MC6\_Methods

*Load list of level 6 multiple-concentration flag methods***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, "Data\_processing."

**modl.directionality.fail** Flag series if model directionality is questionable, i.e. if the winning model direction was opposite, more responses (`resp`) would have exceeded the cutoff (`coff`). If loss was winning directionality (`top < 0`), flag if  $\text{count}(\text{resp} < -1 * \text{coff}) < 2 * \text{count}(\text{resp} > \text{coff})$ . If gain was winning directionality (`top > 0`), flag if  $\text{count}(\text{resp} > \text{coff}) < 2 * \text{count}(\text{resp} < -1 * \text{coff})$ .

**low.nrep** Flag series if the average number of replicates per concentration is less than 2;  $nrep < 2$ .

**low.nconc** Flag series if 4 concentrations or less were tested;  $nconc \leq 4$ .

**bmd.high** Flag series if modeled benchmark dose (BMD) is greater than AC50 (concentration at 50 percent maximal response). This indicates high variability in baseline response in excess of more than half of the maximal response.

**singlept.hit.high** Flag single-point hit that's only at the highest conc tested, where series is an active hit call (`hitc >= 0.9`) with the median response observed above baseline occurring only at the highest tested concentration tested.

**singlept.hit.mid** Flag single-point hit that's not at the highest conc tested, where series is an active hit call (`hitc >= 0.9`) with the median response observed above baseline occurring only at one concentration and not the highest concentration tested.

**multipoint.neg** Flag multi-point miss, where series is an inactive hit call (`hitc < 0.9`) with multiple median responses observed above baseline.

**gnls.lowconc** Flag series where winning model is gain-loss (`gnls`) and the gain AC50 is less than the minimum tested concentration, and the loss AC50 is less than the mean tested concentration.

- noise** Flag series as noisy if the quality of fit as calculated by the root mean square error (rmse) for the series is greater than the cutoff (coff);  $rmse > coff$ .
- border** Flag series if borderline activity is suspected based on modeled top parameter (top) relative to cutoff (coff);  $|top| \leq 1.2 * coff$  or  $|top| \geq 0.8 * coff$ .
- overfit.hit** Method not yet updated for tcpl implementation. Flag hit-calls that would get changed after doing the small N correction to the aic values.
- efficacy.50** Flag low efficacy hits if series has an active hit call ( $hitc \geq 0.9$ ) and efficacy values (e.g. top and maximum median response) less than 50 percent; intended for biochemical assays. If  $hitc \geq 0.9$  and  $coff \geq 5$ , then flag when  $top < 50$  or  $maxmed < 50$ . If  $hitc \geq 0.9$  and  $coff < 5$ , then flag when  $top < \log_2 1.5$  or  $maxmed < \log_2 1.5$ .
- ac50.lowconc** Flag series with an active hit call ( $hitc \geq 0.9$ ) if AC50 (concentration at 50 percent maximal response) is less than the lowest concentration tested; if  $hitc \geq 0.9$  and  $ac50 < 10^{\log_{e} min}$ , then flag.
- viability.gnls** Flag series with an active hit call ( $hitc \geq 0.9$ ) if denoted as cell viability assay with winning model is gain-loss (gnls); if  $hitc \geq 0.9$ ,  $modl = "gnls"$  and  $cell\_viability\_assay = 1$ , then flag.
- no.med.gt.3bmad** Flag series where no median response values are greater than baseline as defined by 3 times the baseline median absolute deviation (bmad);  $nmed\_gtbl\_pos$  and  $nmed\_gtbl\_neg$  both = 0, where  $nmed\_gtbl\_pos/_neg$  is the number of medians greater than  $3 * bmad$ /less than  $-3 * bmad$ .

### Note

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

### See Also

[mc6](#), [Method functions](#) to query what methods get applied to each acid.

---

mcdat

*A subset of ToxCast data showing changes in the activity of the intracellular estrogen receptor.*

---

### Description

The example dataset is used to illustrate how the user can pipeline multiple-concentration data from chemical screening using tcpl.

### Usage

mcdat

**Format**

A data frame with 14183 rows and 10 variables:

**spid** sample ID  
**apid** assay plate ID  
**rowi** well-plate row number  
**coli** well-plate column number  
**wllt** well type  
**wllq** well quality  
**conc** concentration in micromolar  
**rval** raw assay component readout value  
**srcf** source file containing the data  
**acsn** assay component source name

**Source**

ToxCast database

---

mc_test	<i>List of lists containing queries sent to tcplQuery associated with each test case. Each list also contains the associated ids with each case. Only meant to be used with automated testing with mocking for mc data.</i>
---------	---

---

**Description**

List of lists containing queries sent to tcplQuery associated with each test case. Each list also contains the associated ids with each case. Only meant to be used with automated testing with mocking for mc data.

**Usage**

mc\_test

**Format**

A list with 30 items:

**tcplConfQuery** Data table with 1 row and 2 columns used for each test case for establishing connection using tcplConf. This data table mocks the response one would get from connecting with invitrodb.

**mc0\_by\_m0id** List containing the queries used for loading mc0 data by m0id via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'm0id' labeled item storing the id used to load the data, for use in tests.





- mcagg\_by\_aeid** List containing the queries used for loading mc 'agg' data by aeid via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the id used to load the data, for use in tests.
- plot\_single\_m4id** List containing the queries used for loading and plotting mc data by m4id via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'm4id' labeled item storing the id used to load the data, for use in tests.
- plot\_multiple\_m4id** List containing the queries used for loading and plotting mc data by multiple m4ids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'm4id' labeled item storing the ids used to load the data, for use in tests.
- plot\_single\_aeid** List containing the queries used for loading and plotting mc data by aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the id used to load the data, for use in tests.
- plot\_multiple\_aeid** List containing the queries used for loading and plotting mc data by multiple aeids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the ids used to load the data, for use in tests.
- plot\_single\_spid** List containing the queries used for loading and plotting mc data by spid/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid' and 'aeid' labeled items storing the ids used to load the data, for use in tests.
- plot\_multiple\_spid** List containing the queries used for loading and plotting mc data by multiple spids/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid' and 'aeid' labeled items storing the ids used to load the data, for use in tests.
- plot\_single\_m4id\_compare** List containing the queries used for loading and plotting compared mc data by m4id via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'm4id' and 'compare.m4id' labeled items storing the ids used to load the data, for use in tests.
- plot\_multiple\_m4id\_compare** List containing the queries used for loading and plotting compared mc data by multiple m4ids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'm4id' and 'compare.m4id' labeled items storing the ids used to load the data, for use in tests.
- plot\_single\_aeid\_compare** List containing the queries used for loading and plotting compared mc data by aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'aeid' and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.
- plot\_multiple\_aeid\_compare** List containing the queries used for loading and plotting compared mc data by multiple aeids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'aeid' and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.
- plot\_single\_spid\_compare** List containing the queries used for loading and plotting compared mc data by spid/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid', 'compare.spid', 'aeid', and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_multiple\_spid\_compare** List containing the queries used for loading and plotting compared mc data by multiple spids/aeid via tcpIPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid', 'compare.spid', 'aeid', and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

### Source

ToxCast database

---

mc\_vignette

*List with multi-concentration data for the vignette*

---

### Description

This dataset is a list with 6 data.tables (mc0,mc1,mc2,mc3,mc4,mc5).

### Usage

mc\_vignette

### Format

1. **mc0** A data frame with 78 rows and 18 columns containing level 0 formatted raw data.
  - spid** Sample ID
  - chid** Unique chemical ID number for tcpl
  - casn** Chemical Abstract Service(CAS) number
  - chnm** Chemical name
  - dsstox\_substance\_id** Chemical-specific DTXSID
  - code** CAS number compressed into numeric string
  - acid** Assay Component ID
  - acnm** Assay Component Name
  - m0id** Level 0 (mc0) ID
  - apid** Assay plate ID
  - rowi** Row Index
  - coli** Column Index
  - wllt** Well Type
  - wllq** Well Quality (0 or 1)
  - conc** Concentration in micromolar
  - rval** Raw assay component readout value
  - srcf** Source file containing the raw data
  - conc\_unit** Concentration Units
2. **mc1** A data frame with 78 rows and 21 columns containing level 1 replicate and concentration level indicated data.
  - spid** Sample ID

- chid** Unique chemical ID number for tcpl  
**casn** Chemical Abstract Service(CAS) number  
**chnm** Chemical name  
**dsstox\_substance\_id** Chemical-specific DTXSID  
**code** CAS number compressed into numeric string  
**acid** Assay Component ID  
**acnm** Assay Component Name  
**m0id** Level 0 (mc0) ID  
**m1id** Level 1 (mc1) ID  
**apid** Assay plate ID  
**rowi** Row Index  
**coli** Column Index  
**wllt** Well Type  
**wllq** Well Quality (0 or 1)  
**conc** Concentration in micromolar  
**rval** Raw assay component readout value  
**ndx** Concentration index defined by ranking the unique concentrations, with the lowest concentration starting at 1.  
**repi** Temporary replicate ID is defined, the data are scanned from top to bottom and increment the replicate index every time a replicate ID is duplicated  
**srcf** Source file containing the raw data  
**conc\_unit** Concentration Units
3. **mc2** A data frame with 78 rows and 20 columns containing level 2 assay component-specific corrections.
- spid** Sample ID  
**chid** Unique chemical ID number for tcpl  
**casn** Chemical Abstract Service(CAS) number  
**chnm** Chemical name  
**dsstox\_substance\_id** Chemical-specific DTXSID  
**code** CAS number compressed into numeric string  
**acid** Assay Component ID  
**acnm** Assay Component Name  
**m0id** Level 0 (mc0) ID  
**m1id** Level 1 (mc1) ID  
**m2id** Level 2 (mc2) ID  
**apid** Assay plate ID  
**rowi** Row Index  
**coli** Column Index  
**wllt** Well Type  
**conc** Concentration in micromolar  
**eval** Corrected Value  
**ndx** Concentration index defined by ranking the unique concentrations, with the lowest concentration starting at 1.

- repi** Temporary replicate ID is defined, the data are scanned from top to bottom and increment the replicate index every time a replicate ID is duplicated
- conc\_unit** Concentration Units
4. **mc3** A data frame with 78 rows and 22 columns containing level 3 assay endpoint normalized data.
- spid** Sample ID
- chid** Unique chemical ID number for tcpl
- casn** Chemical Abstract Service(CAS) number
- chnm** Chemical name
- dsstox\_substance\_id** Chemical-specific DTXSID
- code** CAS number compressed into numeric string
- aeid** Assay Component Endpoint ID
- aenm** Assay endpoint name (i.e., assay\_component\_endpoint\_name)
- m0id** Level 0 (mc0) ID
- m1id** Level 1 (mc1) ID
- m2id** Level 2 (mc2) ID
- m3id** Level 3 (mc3) ID
- logc** Log base 10 concentration
- resp** Normalized response value
- cndx** Concentration index defined by ranking the unique concentrations, with the lowest concentration starting at 1.
- wllt** Well Type
- apid** Assay plate ID
- rowi** Row Index
- coli** Column Index
- repi** Temporary replicate ID is defined, the data are scanned from top to bottom and increment the replicate index every time a replicate ID is duplicated
- resp\_unit** Response Units
- conc\_unit** Concentration Units
5. **mc4** A data frame with 5 rows and 149 columns containing level 4 concentration-response fitting data (all fits).
- spid** Sample ID
- chid** Unique chemical ID number for tcpl
- casn** Chemical Abstract Service(CAS) number
- chnm** Chemical name
- dsstox\_substance\_id** Chemical-specific DTXSID
- code** CAS number compressed into numeric string
- aeid** Assay Component Endpoint ID
- aenm** Assay endpoint name (i.e., assay\_component\_endpoint\_name)
- m4id** Level 4 (mc4) ID
- bmad** The median absolute deviation of all treatment wells (default option) or blank wells
- resp\_max** Maximum observed response

**resp\_min** Minimum observed response  
**max\_mean** Maximum mean response  
**max\_mean\_conc** Concentration of the maximum mean response  
**max\_med** Maximum median response  
**max\_med\_conc** Concentration of the maximum median response  
**logc\_max** Maximum concentration on the log scale  
**logc\_min** Minimum concentration on the log scale  
**nconc** The total number of concentration groups  
**npts** Total number of observed responses (i.e. data points in the concentration series)  
**nrep** Number of replicates in concentration groups  
**nmed\_gtbl** The number of median responses greater than 3BMAD  
**cnst\_success** Success indicator for the Constant model; 1 if the optimization was successful, otherwise 0  
**cnst\_aic** Akaike Information Criteria (AIC) for the Constant model  
**cnst\_rme** Root mean square error for the Constant model  
**cnst\_er** Error term for the Constant model  
**hill\_success** Success indicator for the Hill model; 1 if the optimization was successful, otherwise 0  
**hill\_aic** Akaike Information Criteria (AIC) for the Hill model  
**hill\_cov** Success indicator for the Hill model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0  
**hill\_rme** Root mean square error for the Hill model  
**hill\_tp** The top parameter indicating the maximal estimated response  
**hill\_ga** The gain parameter for the Hill model, gain AC50  
**hill\_p** The power parameter for the Hill model  
**hill\_er** Error term for the Hill model  
**hill\_tp\_sd** Standard deviation of the Hill model top parameter  
**hill\_ga\_sd** Standard deviation of the Hill model gain parameter  
**hill\_p\_sd** Standard deviation of the Hill model power parameter  
**hill\_er\_sd** Standard deviation of the Hill model error term  
**hill\_top** The maximal response on the resulting Hill model fit  
**hill\_ac50** Concentration at 50% of the maximal response on the Hill model fit  
**gnls\_success** Success indicator for the Gain-loss model; 1 if the optimization was successful, otherwise 0  
**gnls\_aic** Akaike Information Criteria (AIC) for the Gain-loss model  
**gnls\_cov** Success indicator for the Gain-loss model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0  
**gnls\_rme** Root mean square error for the Gain-loss model  
**gnls\_tp** The top parameter indicating the maximal estimated response  
**gnls\_ga** The gain parameter for the Gain-loss model, gain AC50  
**gnls\_p** The gain power parameter for the Gain-loss model  
**gnls\_la** The loss parameter for the Gain-loss model, loss AC50  
**gnls\_q** The loss power parameter for the Gain-loss model

**gnls\_er** Error term for the Gain-loss model

**gnls\_tp\_sd** Standard deviation of the Gain-loss model top parameter

**gnls\_ga\_sd** Standard deviation of the Gain-loss model gain parameter

**gnls\_p\_sd** Standard deviation of the Gain-loss model gain power parameter

**gnls\_la\_sd** Standard deviation of the Gain-loss model loss parameter

**gnls\_q\_sd** Standard deviation of the Gain-loss model loss power parameter

**gnls\_er\_sd** Standard deviation of the Gain-loss model error term

**gnls\_top** The maximal response on the resulting Gain-loss model fit

**gnls\_ac50** Concentration at 50% of the maximal response on the Gain-loss model fit, gain AC50

**gnls\_ac50\_loss** Concentration at 50% of the maximal response on the Gain-loss model fit, loss AC50

**poly1\_success** Success indicator for the Polynomial 1 model; 1 if the optimization was successful, otherwise 0

**poly1\_aic** Akaike Information Criteria (AIC) for the Polynomial 1 model

**poly1\_cov** Success indicator for the Polynomial 1 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0

**poly1\_rme** Root mean square error for the Polynomial 1 model

**poly1\_a** The y-scale parameter for the Polynomial 1 model

**poly1\_er** Error term for the Polynomial 1 model

**poly1\_a\_sd** Standard deviation of the Polynomial 1 model y-scale parameter

**poly1\_er\_sd** Standard deviation of the Polynomial 1 model error term

**poly1\_top** The maximal response on the resulting Polynomial 1 model fit

**poly1\_ac50** Concentration at 50% of the maximal response on the Polynomial 1 model fit

**poly2\_success** Success indicator for the Polynomial 2 model; 1 if the optimization was successful, otherwise 0

**poly2\_aic** Akaike Information Criteria (AIC) for the Polynomial 2 model

**poly2\_cov** Success indicator for the Polynomial 2 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0

**poly2\_rme** Root mean square error for the Polynomial 2 model

**poly2\_a** The y-scale parameter for the Polynomial 2 model

**poly2\_b** The x-scale parameter for the Polynomial 2 model

**poly2\_er** Error term for the Polynomial 2 model

**poly2\_a\_sd** Standard deviation of the Polynomial 2 model y-scale parameter

**poly2\_b\_sd** Standard deviation of the Polynomial 2 model x-scale parameter

**poly2\_er\_sd** Standard deviation of the Polynomial 2 model error term

**poly2\_top** The maximal response on the resulting Polynomial 2 model fit

**poly2\_ac50** Concentration at 50% of the maximal response on the Polynomial 2 model fit

**pow\_success** Success indicator for the Power model; 1 if the optimization was successful, otherwise 0

**pow\_aic** Akaike Information Criteria (AIC) for the Power model

**pow\_cov** Success indicator for the Power model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0

**pow\_rme** Root mean square error for the Power model  
**pow\_a** The y-scale parameter for the Power model  
**pow\_p** The power parameter for the Power model  
**pow\_er** Error term for the Power model  
**pow\_a\_sd** Standard deviation of the Power model y-scale parameter  
**pow\_p\_sd** Standard deviation of the Power model power parameter  
**pow\_er\_sd** Standard deviation of the Power model error term  
**pow\_top** The maximal response on the resulting Power model fit  
**pow\_ac50** Concentration at 50% of the maximal response on the Power model fit  
**exp2\_success** Success indicator for the Exponential 2 model; 1 if the optimization was successful, otherwise 0  
**exp2\_aic** Akaike Information Criteria (AIC) for the Exponential 2 model  
**exp2\_cov** Success indicator for the Exponential 2 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0  
**exp2\_rme** Root mean square error for the Exponential 2 model  
**exp2\_a** The y-scale parameter for the Exponential 2 model  
**exp2\_b** The x-scale parameter for the Exponential 2 model  
**exp2\_er** Error term for the Exponential 2 model  
**exp2\_a\_sd** Standard deviation of the Exponential 2 model y-scale parameter  
**exp2\_b\_sd** Standard deviation of the Exponential 2 model x-scale parameter  
**exp2\_er\_sd** Standard deviation of the Exponential 2 model error term  
**exp2\_top** The maximal response on the resulting Exponential 2 model fit  
**exp2\_ac50** Concentration at 50% of the maximal response on the Exponential 2 model fit  
**exp3\_success** Success indicator for the Exponential 3 model; 1 if the optimization was successful, otherwise 0  
**exp3\_aic** Akaike Information Criteria (AIC) for the Exponential 3 model  
**exp3\_cov** Success indicator for the Exponential 3 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0  
**exp3\_rme** Root mean square error for the Exponential 3 model  
**exp3\_a** The y-scale parameter for the Exponential 3 model  
**exp3\_b** The x-scale parameter for the Exponential 3 model  
**exp3\_p** The power parameter for the Exponential 3 model  
**exp3\_er** Error term for the Exponential 3 model  
**exp3\_a\_sd** Standard deviation of the Exponential 3 model y-scale parameter  
**exp3\_b\_sd** Standard deviation of the Exponential 3 model x-scale parameter  
**exp3\_p\_sd** Standard deviation of the Exponential 3 model power parameter  
**exp3\_er\_sd** Standard deviation of the Exponential 3 model error term  
**exp3\_top** The maximal response on the resulting Exponential 3 model fit  
**exp3\_ac50** Concentration at 50% of the maximal response on the Exponential 3 model fit  
**exp4\_success** Success indicator for the Exponential 4 model; 1 if the optimization was successful, otherwise 0  
**exp4\_aic** Akaike Information Criteria (AIC) for the Exponential 4 model



- exp4\_cov** Success indicator for the Exponential 4 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0
  - exp4\_rme** Root mean square error for the Exponential 4 model
  - exp4\_tp** The top parameter indicating the maximal estimated response
  - exp4\_ga** The gain parameter for the Exponential 4 model, gain AC50
  - exp4\_er** Error term for the Exponential 4 model
  - exp4\_tp\_sd** Standard deviation of the Exponential 4 model top parameter
  - exp4\_ga\_sd** Standard deviation of the Exponential 4 model gain parameter
  - exp4\_er\_sd** Standard deviation of the Exponential 4 model error term
  - exp4\_top** The maximal response on the resulting Exponential 4 model fit
  - exp4\_ac50** Concentration at 50% of the maximal response on the Exponential 4 model fit
  - exp5\_success** Success indicator for the Exponential 5 model; 1 if the optimization was successful, otherwise 0
  - exp5\_aic** Akaike Information Criteria (AIC) for the Exponential 5 model
  - exp5\_cov** Success indicator for the Exponential 5 model covariance calculation; 1 if the Hessian matrix inversion is successful, otherwise 0
  - exp5\_rme** Root mean square error for the Exponential 5 model
  - exp5\_tp** The top parameter indicating the maximal estimated response
  - exp5\_ga** The gain parameter for the Exponential 5 model, gain AC50
  - exp5\_p** The power parameter for the Exponential 5 model
  - exp5\_er** Error term for the Exponential 5 model
  - exp5\_tp\_sd** Standard deviation of the Exponential 5 model top parameter
  - exp5\_ga\_sd** Standard deviation of the Exponential 5 model gain parameter
  - exp5\_p\_sd** Standard deviation of the Exponential 5 model power parameter
  - exp5\_er\_sd** Standard deviation of the Exponential 5 model error term
  - exp5\_top** The maximal response on the resulting Exponential 5 model fit
  - exp5\_ac50** Concentration at 50% of the maximal response on the Exponential 5 model fit
  - all\_onesd** Standard deviation of the baseline response for all models
  - all\_bmed** Median noise estimation of the baseline response for all models
  - resp\_unit** Response Units
  - conc\_unit** Concentration Units
6. **mc5** A data frame with 5 rows and 54 columns containing level 5 best curve-fit and hitcall data.
- spid** Sample ID
  - chid** Unique chemical ID number for tcpl
  - casn** Chemical Abstract Service(CAS) number
  - chnm** Chemical name
  - dsstox\_substance\_id** Chemical-specific DTXSID
  - code** CAS number compressed into numeric string
  - aeid** Assay Component Endpoint ID
  - aenm** Assay endpoint name (i.e., assay\_component\_endpoint\_name)
  - m5id** Level 5 (mc5) ID

**m4id** Level 4 (mc4) ID  
**bmad** The median absolute deviation of all treatment wells (default option) or blank wells  
**resp\_max** Maximum observed response  
**resp\_min** Minimum observed response  
**max\_mean** Maximum mean response  
**max\_mean\_conc** Concentration of the maximum mean response  
**max\_med** Maximum median response  
**max\_med\_conc** Concentration of the maximum median response  
**logc\_max** Maximum concentration on the log scale  
**logc\_min** Minimum concentration on the log scale  
**nconc** The total number of concentration groups  
**npts** Total number of observed responses (i.e. data points in the concentration series)  
**nrep** Number of replicates in concentration groups  
**nmed\_gtbl** The number of median responses greater than 3BMAD  
**hitc** Hitcall  
**modl** Best model fit from tcplFit2 curve-fitting  
**fitc** Fit category  
**coff** Cutoff  
**top\_over\_cutoff** Ratio of the top of the best model fit curve and the cutoff  
**rmse** Root mean squared error  
**a** The y-scale parameter for poly1, poly2, pow, exp2, or exp3 model  
**er** Error term  
**bmr** Benchmark response  
**bmdl** Lower 95% confidence bound on the benchmark dose/concentration estimate  
**caikwt** Akaike Information Criteria weight of constant model relative to the best model fit  
**mll** Maximum log-likelihood of the best model fit  
**hitcall** Continuous hitcall  
**ac50** Concentration where 50% of the maximal response occurs - if 'modl' is the Hill or Gain-loss model this is for the "gain" side of the response  
**top** The maximal response on the best model curve fit - i.e. top of the curve fit  
**ac5** Concentration where 5% of the maximal response occurs  
**ac10** Concentration where 10% of the maximal response occurs  
**ac20** Concentration where 20% of the maximal response occurs  
**acc** Concentration where the efficacy cutoff response occurs  
**ac1sd** Concentration where one standard deviation of the background response occurs  
**bmd** Benchmark response/concentration estimate - concentration where the benchmark response occurs  
**bmdu** Upper 95% confidence bound on the benchmark dose/concentration estimate  
**tp** The top curve parameter for the exp4, exp5, hill, or gnls model  
**ga** The gain parameter for the hill or gnls model - gain AC50  
**p** The power parameter for the pow, exp3, exp5, gnls, or hill model - for gnls this is the gain power parameter

- q** The loss power parameter for the gnls model
- la** The loss parameter for the gnls model, loss AC50
- ac50\_loss** Concentration where 50% of the maximal response occurs - if 'modl' is the Hill or Gain-loss model this is for the "loss" side of the response
- b** The x-scale parameter for poly2, exp2, or exp3 model
- resp\_unit** Response Units
- conc\_unit** Concentration Units

---

 Method functions

*Functions for managing processing methods*


---

## Description

These functions are used to manage which methods are used to process data. They include methods for assigning, clearing, and loading the assigned methods. Also, `tcplMthdList` lists the available methods.

## Usage

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

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

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

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

## Arguments

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

## Details

`tcplMthdLoad` loads the assigned methods for the given level and ID(s). Similarly, `tcplMthdList` displays the available methods for the given level. These two functions do not make any changes to the database.

Unlike the `-Load` and `-List` functions, the `-Assign` and `-Clear` functions alter the database and trigger a delete cascade. `tcplMthdAssign` assigns methods to the given ID(s), and `tcplMthdClear` removes methods. In addition to the method ID (`'mthd_id'`), assigning methods at some levels require an order (`'ordr'`). The `'ordr'` parameter is necessary to allow progression of methods at

level one for single-concentration processing, and levels two and three for multiple-concentration processing. More information about method assignments and the delete cascade are available in the package vignette.

### Examples

```
## Not run:
## tcplListMthd allows the user to display the available methods for
## a given level and data type
head(tcplMthdList(lvl = 2, type = "mc"))

## tcplLoadMthd shows which methods are assigned for the given ID, level,
## and data type. Here we will show how to register, load, and clear methods
## using an acid not in the example database. Note: There is no check for
## whether an ID exists before assigning/clearing methods.
tcplMthdLoad(lvl = 2, id = 55, type = "mc")

## ACID 55 does not have any methods. Assign methods from the list above.
tcplMthdAssign(lvl = 2,
               id = 55,
               mthd_id = c(3, 4, 2),
               ordr = 1:3,
               type = "mc")
## Method assignment can be done for multiple assays, too.
tcplMthdAssign(lvl = 2,
               id = 53:54,
               mthd_id = c(3, 4, 2),
               ordr = 1:3,
               type = "mc")

## Cleanup example method assignments
tcplMthdClear(lvl = 2, id = 53:55, type = "mc")

## End(Not run)
```

---

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 `tcplFit` function.

### Usage

```
tcplObjCnst(p, resp)

tcplObjGnls(p, lconc, resp)

tcplObjHill(p, lconc, resp)
```

```
tcplObjCnst(p, resp)
```

```
tcplObjGnls(p, lconc, resp)
```

```
tcplObjHill(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  $\nu$  degrees of freedom,  $y_i$  be the observed response at the  $i^{th}$  observation, and  $\mu_i$  be the estimated response at the  $i^{th}$  observation. We calculate  $z_i$  as:

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

where  $\sigma$  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)

`tcplObjCnst` calculates the likelihood for a constant model at 0. The only parameter passed to `tcplObjCnst` by `p` is the scale term  $\sigma$ . The constant model value  $\mu_i$  for the  $i^{th}$  observation is given by:

$$\mu_i = 0$$

`tcplObjGnls` calculates the likelihood for a constant model at 0. The only parameter passed to `tcplObjGnls` by `p` is the scale term  $\sigma$ . The constant model value  $\mu_i$  for the  $i^{th}$  observation is given by:

$$\mu_i = 0$$

**Gain-Loss Model (gnls)**

tcp1objGnls 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 tcp1objGnls 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 ( $\sigma$ ). The gain-loss model value  $\mu_i$  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.

tcp1objGnls 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 tcp1objGnls 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 ( $\sigma$ ). The gain-loss model value  $\mu_i$  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)**

tcp1objHill calculates the likelihood for a 3 parameter Hill model with the bottom equal to 0. The parameters passed to tcp1objHill by p are (in order) top ( $tp$ ), log AC50 ( $ga$ ), hill coefficient ( $gw$ ), and the scale term ( $\sigma$ ). The hill model value  $\mu_i$  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.

tcp1objHill calculates the likelihood for a 3 parameter Hill model with the bottom equal to 0. The parameters passed to tcp1objHill by p are (in order) top ( $tp$ ), log AC50 ( $ga$ ), hill coefficient ( $gw$ ), and the scale term ( $\sigma$ ). The hill model value  $\mu_i$  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.

---

mthd\_list\_defaults      *Lists of data frames returned from tcplMthdList invitrodb v4.2*

---

### Description

Lists of data frames returned from tcplMthdList invitrodb v4.2

### Usage

```
mthd_list_defaults
```

### Format

A list with 7 items:

**mc2** displays the available methods for mc lvl 2 data

**mc3** displays the available methods for mc lvl 3 data

**mc4** displays the available methods for mc lvl 4 data

**mc5** displays the available methods for mc lvl 5 data

**mc6** displays the available methods for mc lvl 6 data

**sc1** displays the available methods for sc lvl 1 data

**sc2** displays the available methods for sc lvl 2 data

### Source

ToxCast database

---

Query functions      *Wrappers for sending queries and fetching results*

---

### Description

These functions send a query to the given database, and are the access point for all tcpl functions that query or update the tcpl database.

### Usage

```
tcplQuery(  
  query,  
  db = getOption("TCPL_DB"),  
  drvr = getOption("TCPL_DRVR"),  
  tbl = NULL  
)
```

```

tcplQueryAPI(resource = "data", fld = NULL, val = NULL, return_flds = NULL)

tcplSendQuery(
  query,
  db = getOption("TCPL_DB"),
  drvr = getOption("TCPL_DVR"),
  tbl = NULL,
  delete = F
)

```

### Arguments

query	Character of length 1, the query string
db	Character of length 1, the name of the tcpl database
drvr	Character of length 1, which database driver to use
tbl	Tables to be read queried
resource	must be either data or assay to determine which api endpoint to hit
fld	field that should be used to query the api
val	value for specified field to query on
return_flds	optional list of fields that should be returned
delete	Logical of length 1, execute delete on queried table

### Details

Currently, the tcpl package supports the "MySQL", "example", and "API" database drivers.

tcplQuery returns a data.table object with the query results. tcplSendQuery sends a query, but does not fetch any results, and returns 'TRUE' or the error message given by the database. tcplQueryAPI returns a data.table object with the query results when connected using "API" as driver.

### Examples

```

## Not run:
# only with MySQL driver
tcplQuery("SELECT 'Hello World';")

# only with API driver
tcplConfDefault()
tcplQueryAPI(resource = "data", fld = "aeid", val = 2)

## End(Not run)

```



---

Register/update annotation

*Functions for registering & updating annotation information*

---

## Description

These functions are used to register and update the chemical and assay annotation information.

## Usage

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

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

## Arguments

what	Character of length 1, the name of the ID to register or update
flds	Named list, the other fields and their values
id	Integer, the ID value(s) to update

## Details

These functions are used to populate the tcpl database with the necessary annotation information to complete the processing. As shown in the package vignette, the package requires some information about the samples and assays before data can be loaded into the tcpl database.

Depending on what is being registered, different information is required. The following table lists the fields that can be registered/updated by these functions, and the minimal fields required for registering a new ID. (The database table affected is in parentheses.)

- asid (assay\_source): assay\_source\_name
- aid (assay): asid, assay\_name, assay\_footprint
- acid (assay\_component): aid, assay\_component\_name
- aeid (assay\_component\_endpoint): acid, assay\_component\_endpoint\_name, normalized\_data\_type
- acsn (assay\_component\_map): acid, acsn
- spid (sample): spid, chid
- chid (chemical): chid, casn
- clib (chemical\_library): chid, clib

Note: The functions accept the abbreviated forms of the names, ie. "aenm" rather than the full "assay\_component\_endpoint\_name." More information about the registration process and all of the fields is available in the vignette.

**Examples**

```

## Not run:
## Load current ASID information
tcplLoadAsid()

## Register a new assay source
tcplRegister(what = "asid", flds = list(asnm = "example_asid"))

## Show the newly registered ASID
tcplLoadAsid(add.fld = "assay_source_desc")

## Notice that the newly created ASID does not have an assay_source_desc.
## The field could have been defined during the registration process, but
## can also be updated using tcplUpdate
i1 <- tcplLoadAsid()[asnm == "example_asid", asid]
tcplUpdate(what = "asid",
           id = i1,
           flds = list(assay_source_desc = "example asid description"))
tcplLoadAsid(add.fld = "assay_source_desc")

## Remove the created ASID. Note: Manually deleting primary keys can cause
## serious database problems and should not generally be done.

## End(Not run)

```

---

registerMthd

*Add a new analysis method*


---

**Description**

registerMthd registers a new analysis method to the tcpl databases.

**Usage**

```
registerMthd(lvl, mthd, desc, nddr = 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
nddr	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. 'nddr' 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

[Method functions, SC1\\_Methods](#)

Other single-concentration: [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, "Data\_processing."

**bval Methods:**

**bval.apid.nwlls.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for neutral control wells (wlft = n).

**bval.apid.twlls.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for test compound wells (wlft = t).

**bval.apid.tn.med** Calculate the baseline value (bval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for test compound wells (wlft = t) and neutral control wells (wlft = n).

**bval.nwlls.med** Calculate the baseline value (bval) as the median of the raw values (rval) for neutral control wells (wlft = n) by assay endpoint id (aeid).

**pval Methods:**

**pval.apid.pwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for single-concentration gain-of-signal positive control wells (wlft = p).

**pval.apid.mwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for multiple-concentration loss-of-signal negative control wells (wlft = m).

**pval.apid.medpbyconc.max** Calculate the positive control value (pval) as the plate-wise maximum, by assay plate ID (apid), of the medians of the raw values (rval) for gain-of-signal single- or multiple-concentration positive control wells (wlft = p or c) by apid, well type, and concentration.

**pval.apid.medpbyconc.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of the raw values (rval) for gain-of-signal single- or multiple-concentration positive control wells (wlft = p or c) by apid, well type, and concentration.

**pval.apid.medncbyconc.min** Calculate the positive control value (pval) as the plate-wise minimum, by assay plate ID (apid), of the medians of the raw values (rval) for gain-of-signal single- or multiple-concentration negative control wells (wlft = m or o) by apid, well type, and concentration.

**pval.zero** Set the positive control value (pval) to 0;  $pval = 0$ .

**pval.apid.or.aid.pwlls.med** Calculate the positive control value (pval) as the plate-wise median, by assay plate ID (apid), of the raw values (rval) for single-concentration gain-of-signal positive control wells (wlft = p). For plates without p wells, set the pval as the median pval calculated from all plates.

#### resp Methods:

**resp.pc** Calculate the normalized response (resp) as a percent of control, i.e. the ratio of the difference between the raw (rval) and baseline (bval) values divided by the difference between positive control (pval) and baseline (bval) values multiplied by 100;  $resp = (rval - bval) / (pval - bval) * 100$ .

**resp.fc** Calculate the normalized response (resp) as fold change, i.e. the ratio of the raw (rval) and baseline (bval) values;  $resp = rval / bval$ .

**resp.logfc** Calculate the normalized response (resp) as the fold change of logged, i.e. the difference between raw (rval) and baseline (bval) log-scale values.

**resp.log2** Transform the response values to log-scale (base 2).

**resp.multneg1** Multiply the normalized response value (resp) by -1;  $-1 * resp$ .

**none** Use raw value (rval) as is. This may be necessary for additional endpoint-specific adjustments, or where no additional sc1 methods are needed.

**resp.incr.zerocenter.fc** Calculate the normalized response (resp) as a zero center fold change, i.e. the ratio of the raw (rval) and baseline (bval) values minus 1;  $resp = rval / bval - 1$ . Typically used for increasing responses.

#### Note

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

#### See Also

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

---

sc2	<i>Perform level 2 single-concentration processing</i>
-----	--

---

## 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

[Method functions](#), [SC2\\_Methods](#)

Other single-concentration: [sc1\(\)](#)

**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**

The methods are broken down into four categories based on the type of cutoff they assign. Different methods are used to define cutoffs for "bmad" (baseline median absolute value), "pc" (percent of control), "pc or bmad", "log" ( $\log_2$  or  $\log_{10}$ ), and "other" (uncategorized methods).

All methods are applied by aeid.

Although there are method exceptions (notably within the "other" category), only highest calculated cutoff value based on assigned methods will be selected for hitcalling. Therefore, only the largest cutoff method per method type should be assigned.

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

**BMAD Methods:**

**bmad1** Add a cutoff value of 1 multiplied by baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlft = t) for the endpoint.

**bmad1.5** Add a cutoff value of 1.5 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlft = t) for the endpoint.

**bmad2** Add a cutoff value of 2 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlft = t) for the endpoint.

**bmad3** Add a cutoff value of 3 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlft = t) for the endpoint.

**bmad5** Add a cutoff value of 5 multiplied the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wlft = t) for the endpoint.

**bmad6** Add a cutoff value of 6 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wllt = t) for the endpoint.

**bmad10** Add a cutoff value of 10 multiplied by the baseline median absolute deviation (bmad). By default, bmad is calculated using test compound wells (wllt = t) for the endpoint.

#### Percent of Control Methods:

**pc0.88** Add a cutoff value of 0.88. Typically for percent of control data.

**pc16** Add a cutoff value of 16. Typically for percent of control data.

**pc20** Add a cutoff value of 20. Typically for percent of control data.

**pc25** Add a cutoff value of 25. Typically for percent of control data.

**pc30** Add a cutoff value of 30. Typically for percent of control data.

#### Percent of Control or BMAD Methods:

**pc30orbmad3** Add a cutoff value of either 30 or 3 multiplied by the baseline median absolute deviation (bmad), whichever is less. By default, bmad is calculated using test compound wells (wllt = t) for the endpoint.

#### Log Methods: Log Base 2

**log2\_0.76** Add a cutoff value of 0.76 for log<sub>2</sub>-transformed data. This was a custom threshold value set for endpoint id 1690 (formerly acid 1691).

**log2\_1.2** Add a cutoff value of log<sub>2</sub>1.2. Typically for fold change data.

**log2\_1.5** Add a cutoff value of log<sub>2</sub>1.5. Typically for fold change data.

#### Log Base 10

**log10\_1.2** Add a cutoff value of log<sub>10</sub>1.2. Typically for fold change data.

#### Other Methods:

**ow\_bmad\_nwells** Overwrite the default baseline median absolute value (bmad) with a bmad calculated using neutral control wells (wllt = n).

**ow\_bidirectional\_gain** Where responses only in the positive direction are biologically relevant, overwrite the max\_med and max\_tmp values, which were calculated using absolute value, to a calculation using a true maximum for uni-directional data.

**ow\_bidirectional\_loss** Where responses only in the negative direction are biologically relevant, overwrite the max\_med and max\_tmp values, which were calculated using absolute value, to a calculation using a true minimum for uni-directional data.

#### Note

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

#### See Also

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



---

smdat	<i>A subset of ToxCast data showing changes in transcription factor activity for multiple targets.</i>
-------	--

---

### Description

The example dataset is used to illustrate how the user can pipeline single-concentration data from chemical screening using tcpl.

### Usage

smdat

### Format

A data frame with 320 rows and 10 variables:

**spid** sample ID  
**apid** assay plate ID  
**rowi** well-plate row number (N/A)  
**coli** well-plate column number (N/A)  
**wllt** well type (N/A)  
**wllq** well quality (N/A)  
**conc** concentration in micromolar  
**rval** raw assay component readout value  
**srcf** source file containing the data  
**acsn** assay component source name

### Source

ToxCast database

---

sc_test	<i>List of lists containing queries sent to tcplQuery associated with each test case. Each list also contains the associated ids with each case. Only meant to be used with automated testing with mocking for sc data.</i>
---------	---

---

### Description

List of lists containing queries sent to tcplQuery associated with each test case. Each list also contains the associated ids with each case. Only meant to be used with automated testing with mocking for sc data.

## Usage

sc\_test

## Format

A list with 20 items:

**tcplConfQuery** Data table with 1 row and 2 columns used for each test case for establishing connection using tcplConf. This data table mocks the response one would get from connecting with invitrodb.

**sc0\_by\_s0id** List containing the queries used for loading sc0 data by s0id via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 's0id' labeled item storing the id used to load the data, for use in tests.

**sc0\_by\_acid** List containing the queries used for loading sc0 data by acid via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'acid' labeled item storing the id used to load the data, for use in tests.

**sc1\_by\_s1id** List containing the queries used for loading sc1 data by s1id via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 's1id' labeled item storing the id used to load the data, for use in tests.

**sc1\_by\_acid** List containing the queries used for loading sc1 data by acid via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'acid' labeled item storing the id used to load the data, for use in tests.

**sc2\_by\_s2id** List containing the queries used for loading sc2 data by s2id via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 's2id' labeled item storing the id used to load the data, for use in tests.

**sc2\_by\_aeid** List containing the queries used for loading sc2 data by aeid via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the id used to load the data, for use in tests.

**scagg\_by\_aeid** List containing the queries used for loading sc 'agg' data by aeid via tcplLoadData. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the id used to load the data, for use in tests.

**plot\_single\_s2id** List containing the queries used for loading and plotting sc data by s2id via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 's2id' labeled item storing the id used to load the data, for use in tests.

**plot\_multiple\_s2id** List containing the queries used for loading and plotting sc data by multiple s2ids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 's2id' labeled item storing the ids used to load the data, for use in tests.

**plot\_single\_aeid** List containing the queries used for loading and plotting sc data by aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the id used to load the data, for use in tests.

**plot\_multiple\_aeid** List containing the queries used for loading and plotting sc data by multiple aeids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains one 'aeid' labeled item storing the ids used to load the data, for use in tests.

**plot\_single\_spid** List containing the queries used for loading and plotting sc data by spid/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid' and 'aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_multiple\_spid** List containing the queries used for loading and plotting sc data by multiple spids/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid' and 'aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_single\_s2id\_compare** List containing the queries used for loading and plotting compared sc data by s2id via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 's2id' and 'compare.s2id' labeled items storing the ids used to load the data, for use in tests.

**plot\_multiple\_s2id\_compare** List containing the queries used for loading and plotting compared sc data by multiple s2ids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 's2id' and 'compare.s2id' labeled items storing the ids used to load the data, for use in tests.

**plot\_single\_aeid\_compare** List containing the queries used for loading and plotting compared sc data by aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'aeid' and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_multiple\_aeid\_compare** List containing the queries used for loading and plotting compared sc data by multiple aeids via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'aeid' and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_single\_spid\_compare** List containing the queries used for loading and plotting compared sc data by spid/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid', 'compare.spid', 'aeid', and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

**plot\_multiple\_spid\_compare** List containing the queries used for loading and plotting compared sc data by multiple spids/aeid via tcplPlot. Each query has an associated data table response for mocking an actual connection. Contains 'spid', 'compare.spid', 'aeid', and 'compare.aeid' labeled items storing the ids used to load the data, for use in tests.

## Source

ToxCast database

---

sc\_vignette

*List with single-concentration data for the vignette*

---

## Description

This dataset is a list with 3 data.tables (sc0,sc1,sc2).

## Usage

sc\_vignette

**Format**

1. **sc0** A data frame with 10 rows and 18 columns containing level 0 formatted raw data.

**spid** Sample ID  
**chid** Unique chemical ID number for tcpl  
**casn** Chemical Abstract Service(CAS) number  
**chnm** Chemical name  
**dsstox\_substance\_id** Chemical-specific DTXSID  
**code** CAS number compressed into numeric string  
**acid** Assay Component ID  
**acnm** Assay Component Name  
**s0id** Level 0 (sc0) ID  
**apid** Assay plate ID  
**rowi** Row Index  
**coli** Column Index  
**wllt** Well Type  
**wllq** Well Quality (0 or 1)  
**conc** Concentration in micromolar  
**rval** Raw assay component readout value  
**srcf** Source file containing the raw data  
**conc\_unit** Concentration Units

2. **sc1** A data frame with 10 rows and 20 columns containing level 1 normalized data.

**spid** Sample ID  
**chid** Unique chemical ID number for tcpl  
**casn** Chemical Abstract Service(CAS) number  
**chnm** Chemical name  
**dsstox\_substance\_id** Chemical-specific DTXSID  
**code** CAS number compressed into numeric string  
**acid** Assay Component Endpoint ID  
**aenm** Assay endpoint name (i.e., assay\_component\_endpoint\_name)  
**acid** Assay Component ID  
**acnm** Assay Component Name  
**s0id** Level 0 (sc0) ID  
**s1id** Level 1 (sc1) ID  
**apid** Assay plate ID  
**rowi** Row Index  
**coli** Column Index  
**wllt** Well Type  
**logc** Log base 10 concentration  
**resp** Normalized response value  
**resp\_unit** Response Units  
**conc\_unit** Concentration Units

3. **sc2** A data frame with 10 rows and 15 columns containing level 2 efficacy/hit designation data.

**spid** Sample ID  
**chid** Unique chemical ID number for tcpl  
**casn** Chemical Abstract Service(CAS) number  
**chnm** Chemical name  
**dsstox\_substance\_id** Chemical-specific DTXSID  
**code** CAS number compressed into numeric string  
**aeid** Assay Component Endpoint ID  
**aenm** Assay endpoint name (i.e., assay\_component\_endpoint\_name)  
**s2id** Level 2 (sc2) ID  
**bmad** The median absolute deviation of all treatment wells (default option) or blank wells  
**max\_med** Maximum median response  
**hitc** Hitcall  
**coff** Cutoff  
**resp\_unit** Response Units  
**conc\_unit** Concentration Units

---

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\(\)](#)

---

 tcplAddModel

*Draw a tcpl Model onto an existing plot*


---

### Description

tcplAddModel draws a 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 adl is  $1/(3 \cdot \text{bmad})$  for level 4 data and  $1/\text{coff}$  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](#)

### Examples

```
## Create some dummy data to plot
logc <- 1:10
r1 <- sapply(logc, tcplHillVal, ga = 5, tp = 50, gw = 0.5)
r2 <- log2(sapply(logc, tcplHillVal, ga = 4, tp = 30, gw = 0.5))
p1 <- tcplFit(logc = logc, resp = r1, bmad = 10)
p2 <- tcplFit(logc = logc, resp = r2, bmad = log2(1.5))

## In the dummy data above, the two plots are on very different scales
plot(r1 ~ logc, pch = 16, ylab = "raw response")
tcplAddModel(pars = p1, modl = "hill")
points(r2 ~ logc)
tcplAddModel(pars = p2, modl = "hill", lty = "dashed")
```

```
## To visualize the two curves on the same plot for comparison, we can
## scale the values to the bmad, such that a scaled response of 1 will equal
## the bmad for each curve.
plot(r1/10 ~ logc, pch = 16, ylab = "scaled response")
tcplAddModel(pars = p1, modl = "hill", adj = 1/10)
points(r2/log2(5) ~ logc)
tcplAddModel(pars = p2, modl = "hill", adj = 1/log2(5), lty = "dashed")
```

---

tcplAICProb

*Calculate the AIC probabilities*


---

### 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. The calculation assumes every possible model is accounted for, and the results should be interpreted accordingly.

### Value

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

### See Also

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

### Examples

```
## Returns the probability for each model, given models with AIC values
## ranging from 80 to 100
tcplAICProb(80, 85, 90, 95, 100)

## Also works for vectors
m1 <- c(95, 195, 300) ## model 1 for three different observations
```

```
m2 <- c(100, 200, 295) ## model 2 for three different observations
tcplAICProb(m1, m2)
```

---

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, lvl = NULL)
```

### 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
lvl	Usually Integer to indicate what level to auto-increment

### Note

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

---

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.

---

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
------	--

**Details**

The function checks for the validity of the CAS Registry Number. Also, the ToxCast data includes chemicals for which there is no CASRN. The convention for these chemicals is to give them a CASRN as NOCAS\_chid; the code for these compounds is CNOCASchid. The function handles the NOCAS compounds as they are stored in the database, as shown in the example below.

**Value**

A CAS Registry Number.

**Examples**

```
tcp1Code2CASN("C80057")  
tcp1Code2CASN("C09812420") ## Invalid CASRN will give a warning  
tcp1Code2CASN("CNOCAS0015") ## The underscore is reinserted for NOCAS codes
```

---

 tcplCytoPt
 

---



---

*Calculate the cytotoxicity point based on the "burst" endpoints*


---

### Description

tcplCytoPt calculates the cytotoxicity point and average cytotoxicity distribution based on the activity in the "burst" assay endpoints.

### Usage

```
tcplCytoPt(
  chid = NULL,
  aeid = NULL,
  flag = TRUE,
  min.test = TRUE,
  default.pt = 3
)
```

### Arguments

chid	Integer, chemical ID values to subset on
aeid	Integer, assay endpoint ID values to override the "burst assay" definitions
flag	Integer, mc6_mthd_id values to be passed to <a href="#">tcplSubsetChid</a>
min.test	Integer or Boolean, the number of tested assay endpoints required for a chemical to be used in calculating the "global MAD."
default.pt	Numeric of length 1, the default cytotoxicity point value

### Details

tcplCytoPt provides estimates for chemical-specific cytotoxicity distributions (more information available in the vignette.) Before calculating the cytotoxicity distributions, the level 5 data is subsetted by the [tcplSubsetChid](#) function.

The 'chid' parameter specifies a subset of chemicals to use in the calculations, given by chemical ID (chid). The 'aeid' parameter specifies which assays to use in calculating the cytotoxicity point and distribution. By default tcplCytoPt will use all available chemicals and the assay endpoints defined by the 'burst\_assay' field in the "assay\_component\_endpoint" table. The examples show how to identify the "burst" endpoints.

tcplCytoPt returns the cytotoxicity point (the AC50 values of the active "burst" endpoints), the corresponding MAD, and the global MAD (median of the calculated MAD values). Not every chemical must be tested in every "burst" endpoint. The 'min.test' parameter allows the user to specify a minimum number of tested assay endpoints as a requirement for MAD values to be included in the global MAD calculation. For example, suppose the user supplies 10 "burst" assays. The user can choose to require a chemical to be tested in at least 5 of those assays for its MAD value to be included in the global MAD calculation. Having chemicals with many less "burst" endpoints tested may inflate or deflate the global MAD calculation. By default (values of TRUE or NULL),

tcplCytoPt requires a chemical to be tested in at least 80% of the given "burst" assays. The user can also provide 'min.test' values of FALSE (indicating to include all MAD values), or a number (indicating a specific number of endpoints).

Chemicals without at least 2 active "burst" assays do not have a MAD value, and the cytotoxicity point is defined by the 'default.pt' parameter. The default value for 'default.pt' is 3.

The resulting data.table has the following fields:

1. "chid" – The chemical ID.
2. "code" – The chemical code.
3. "chnm" – The chemical name.
4. "casn" – The chemical CASRN.
5. "med" – The median of the "burst" endpoint log(AC50)
6. "mad" – The MAD of the "burst" endpoint log(AC50) values.
7. "ntst" – The number of "burst" endpoints tested.
8. "nhit" – The number of active "burst" endpoints.
9. "used\_in\_global\_mad\_calc" – TRUE/FALSE, whether the mad value was used in the global MAD calculation.
10. "global\_mad" – The median of the "mad" values where "used\_in\_global\_mad\_calc" is TRUE.
11. "cyto\_pt" – The cytotoxicity point, or the value in "med" when "nhit" is at least 2.
12. "cyto\_pt\_um" –  $10^{\text{cyto\_pt}}$
13. "lower\_bnd\_um" –  $10^{\text{cyto\_pt} - 3\text{global\_mad}}$

## Value

A data.table with the cytotoxicity distribution for each chemical. The definition of the field names are listed under "details."

## Examples

```
## Not run:
## Can only calculate the cytotox burst if using the MySQL database and
## TCPL_DRVR == 'MySQL'

if (getOption("TCPL_DRVR") == "MySQL") {

## Load the "burst" endpoints -- none are defined in the example dataset
tcplLoadAeid(fld = "burst_assay", val = 1)

## Calculate the cytotoxicity distributions using both example endpoints
tcplCytoPt(aeid = 1:2)

## The above example does not calculate a global MAD, because no chemical
## hit both endpoints. (This makes sense, because both endpoints are
## derived from one component, where one endpoint is activity in the
## up direction, and the other is activity in the down direction.)
## Note, the cyto_pt is also 3 for all chemicals, because the function
```

```

## requires at least two endpoints to calculate a cytotoxicity point. If
## the user wishes to use one assay, this function is not necessary.

## Changing 'default.pt' will change cyto_pt in the resulting data.table
tcplCytoPt(aeid = 1:2, default.pt = 6)
}

## End(Not run)

```

---

tcpldbStats	<i>Get summary statistics for the database</i>
-------------	--

---

### Description

tcpldbStats takes a string(*type*) and an optional parameter(*val*) to return the summary statistics on the entire tcpl database. When *type* = "all" the *val* is ignored. the function returns the number of distinct spid and aeids in the database at each level. When *type* = "aeid", the *val* parameter has to be a valid aeid in the database. The function returns a table consisting of the number of distinct spids at each level of processing for the aeid given in '*val*'. When *type* = "spid", the *val* parameter has to be a valid spid in the database. The function returns a table consisting of the number of distinct aeids at each level of processing for the given spid in '*val*'.

### Usage

```
tcpldbStats(type = "all", val = NULL)
```

### Arguments

<i>type</i>	String either "all", "aeid" or "spid"
<i>val</i>	integer if <i>type</i> = "aeid" , string if <i>type</i> = "spid"

---

tcplDefine	<i>Load data dictionary descriptions</i>
------------	--

---

### Description

tcplDefine queries the tcpl databases and returns field descriptions from the data dictionary.

### Usage

```
tcplDefine(val = NULL)
```

### Arguments

<i>val</i>	The values to query on. Can be any combination of table names (to return all of its field descriptions) and field names
------------	---

**Details**

Short descriptions of fields for different tables are stored in a data dictionary. Query by table name to retrieve descriptions of each field in the given table, and/or query by field name to retrieve descriptions on every field with the given name, regardless of which table they belong to.

**Value**

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

**Examples**

```
## Store the current config settings, so they can be reloaded at the end
## of the examples
conf_store <- tcplConfList()
tcplConf(drvr = "example")

## Passing no parameters returns all of the fields described in the data
## dictionary
tcplDefine()

## Specifying table names of 'chemical' and 'sample' yields all of the
## fields from the 'chemical' and 'sample' tables
tcplDefine(c("chemical", "sample"))

## Specifying a field of 'wllt' yields all of the fields from any table that
## contains 'wllt' as a field
tcplDefine("wllt")

## Specifying a combination of table and field names results in all of the
## fields which are contained in the given tables and all of the given fields
## found in any table
tcplDefine(c("chemical", "spid", "wllt"))

## Reset configuration
options(conf_store)
```

---

 tcplDelete

*Delete rows from tcpl databases*


---

**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

*Fit the data with the constant, hill, and gain-loss models*

---

**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,
  bidirectional = FALSE,
  verbose = FALSE,
  ...
)
```

**Arguments**

logc	Numeric, log concentration values
resp	Numeric, normalized response values
bmad	Numeric, the baseline median absolute deviation for the entire assay
force.fit	Logical, TRUE indicates to attempt fitting every concentration series
bidirectional	Boolean If TRUE, bidirectional negative data before fitting (default=FALSE) The original version of the code required the data to start at small values and rise, so that negative curves had to be bidirectionalped outside the function, and TOP was always positive. Setting bidirectional to TRUE allows both rising and falling curves

verbose            Boolean If TRUE print warning messages  
 ...                Any other data to be included in list output.

### Details

when at least one median value is greater than  $3*bmad$ .

### Value

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

### See Also

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

### Examples

```
logc <- 1:10
resp <- sapply(1:10, tcplHillVal, ga = 5, tp = 50, gw = 0.5)
params <- tcplFit(logc = logc, resp = resp, bmad = 10)
plot(resp ~ logc)
tcplAddModel(pars = params, modl = "hill")
```

---

tcplFit2	<i>tcpl Wrapper for tcplfit2_core including additional calculations to fit into new schema</i>
----------	--

---

### Description

tcpl Wrapper for tcplfit2\_core including additional calculations to fit into new schema

### Usage

```
tcplFit2(
  dat,
  fitmodels = c("cnst", "hill", "gnls", "poly1", "poly2", "pow", "exp2", "exp3", "exp4",
    "exp5"),
  bmed = NULL
)
```

### Arguments

dat                output from level 3 processing  
 fitmodels        list of the models that should be fit with the data  
 bmed              baseline value, typically should be 0

### Value

Data.table with an additional column fitparams that includes all of the fitting parameters

---

tcpFit2_nest	<i>Nest dataframe into a list that is readable by tcpfit2</i>
--------------	---

---

**Description**

Nest dataframe into a list that is readable by tcpfit2

**Usage**

```
tcpFit2_nest(dat)
```

**Arguments**

dat                    a dataframe that has all of the fitting parameters in the style of tcploaddata

**Value**

a list of fitting parameters that can be consumed by tcpfit2

---

tcpFit2_unnest	<i>Unnest tcpfit2 parameters into a dataframe</i>
----------------	---

---

**Description**

Unnest tcpfit2 parameters into a dataframe

**Usage**

```
tcpFit2_unnest(output)
```

**Arguments**

output                list of output from tcpfit2

**Value**

list of parameters unnested and compiled into a dataframe



---

tcplGetAeid	<i>get Aeid for endpoint name</i>
-------------	-----------------------------------

---

**Description**

tcplGetAeid takes a string(name) and finds the assay component endpoint names that match the string and the aeid associated with those names. The function performs a regular expression like matching for strings in the assay component endpoint name column in the assay component endpoint table.

**Usage**

```
tcplGetAeid(name)
```

**Arguments**

name	A string that will be matched to the assay component endpoint name
------	--

**Examples**

```
## Not run:
## Search for aenm (assay name) case insensitive
tcplGetAeid("TOX21")
tcplGetAeid("tox21")

## End(Not run)
```

---

tcplggplot	<i>tcplggplot</i>
------------	-------------------

---

**Description**

tcplggplot

**Usage**

```
tcplggplot(dat, lvl = 5, verbose = FALSE, flags = FALSE, yrange = c(NA, NA))
```

**Arguments**

dat	data table with all required conc/resp data
lvl	integer level of data that should be plotted level 2 - for 'sc' plotting level 5 - for 'mc' plotting, all fit models and winning model with hitcall
verbose	boolean should plotting include table of values next to the plot
flags	boolean should plotting include level 6 flags in plot caption
yrange	Integer of length 2, for overriding the y-axis range, c(<min>,<max>). By default, c(NA,NA).

**Value**

A ggplot object or grob with accompanied table depending on verbose option

---

tcplggplotCompare	<i>tcplggplotCompare</i>
-------------------	--------------------------

---

**Description**

tcplggplotCompare

**Usage**

```
tcplggplotCompare(
  dat,
  compare.dat,
  lvl = 5,
  verbose = FALSE,
  flags = FALSE,
  yrange = c(NA, NA)
)
```

**Arguments**

dat	data table with all required conc/resp data
compare.dat	data table with all required conc/resp data for comparison overlay
lvl	integer level of data that should be plotted level 2 - for 'sc' plotting level 5 - for 'mc' plotting, all fit models and winning model with hitcall
verbose	boolean should plotting include table of values next to the plot
flags	boolean should plotting include level 6 flags in plot caption
yrange	Integer of length 2, for overriding the y-axis range, c(<min>,<max>). By default, c(NA,NA).

**Value**

A ggplot object or grob with accompanied table depending on verbose option

---

tcplHit2	<i>Hitcalling with tcplfit2</i>
----------	---------------------------------

---

**Description**

Hitcalling with tcplfit2

**Usage**

```
tcplHit2(mc4, coff)
```

**Arguments**

mc4	data.table with level 4 data
coff	cutoff value for hitcalling

**Value**

Data.table with key value pairs of hitcalling parameters

---

tcplLegacyPlot	<i>tcplLegacyPlot</i>
----------------	-----------------------

---

**Description**

tcplLegacyPlot

**Usage**

```
tcplLegacyPlot()
```

**Value**

a ggplot based on old plotting methodology

---

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

---

**Description**

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

**Usage**

```
tcpListFlds(tbl, db = getOption("TCPL_DB"))
```

**Arguments**

tbl	Character of length 1, the tcp database table
db	Character of length 1, the tcp database

**Details**

This function can be particularly useful in defining the 'fld' param in the tcpLoad- functions.

**Value**

A string of field names for the given table.

**Examples**

```
## Not run:  
## Gives the fields in the mc1 table  
tcpListFlds("mc1")  
  
## End(Not run)
```

---

tcpLoadChem	<i>Load sample/chemical information</i>
-------------	---

---

**Description**

tcpLoadChem queries the tcp database and returns the chemical information for the given field and values.

**Usage**

```
tcpLoadChem(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?

### Details

The 'field' parameter is named differently from the 'fld' parameter seen in other functions because it only takes one input.

In the MySQL environment the user should be able to give partial chemical name strings, to find chemicals with similar names. For example, setting 'val' to "phenol" when 'field' is "chnm" and 'exact' is FALSE might pull up the chemicals "Bisphenol A" and "4-Butylphenol". More technically, setting 'exact' to FALSE passes the string in 'val' to an RLIKE statement within the MySQL query.

### Value

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

### Examples

```
## Not run:
## Passing no parameters gives all of the registered chemicals with their
## sample IDs
tcplLoadChem()

## Or the user can exclude spid and get a unique list of chemicals
tcplLoadChem(include.spid = FALSE)

## In addition, the user can retrieve only the registered chemicals from the chemical table
tcplLoadChem(field = 'chem.only')

## Other examples:
tcplLoadChem(field = "chnm", val = "Bisphenol A")
tcplLoadChem(field = "chid", val = 20182)

## End(Not run)
```

---

tcplLoadChemList	<i>Load chemical list information</i>
------------------	---------------------------------------

---

### Description

tcplLoadChemList queries the tcpl databases and returns information about the chemical lists.

**Usage**

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

**Arguments**

field	Character of length 1, 'chid', 'dsstox_substance_id' or 'list_acronym', whether to search by chemical id (chid), dsstox_substance_id, or list_acronym
val	The values to query on

**Details**

Chemicals are stored in different lists by chemical ID. Therefore, it is not possible to delineate samples with the same chemical ID into two distinct chemical lists. However, it is possible for a chemical ID to belong to more than one (or no) chemical lists.

When chemicals belong to more than one list, the chemical is listed multiple times (one for each distinct list).

**Value**

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

**Examples**

```
## Not run:
## Passing no parameters gives all of the chemical IDs that have a chemical
## list registered
clist <- tcplLoadChemList()

## Notice there are different number of rows in tcplLoadChemList than in tcplLoadChem,
## indicating some chemicals must belong to more than list (or no lists).
chem <- tcplLoadChem(include.spid = TRUE)
nrow(chem)
nrow(clist)

## Show the unique chemical lists
clist[, unique(list_acronym)]

## Specifying a chemical list will not show what other libraries a
## chemical might belong to.
tcplLoadChemList(field = "list_acronym", val = "CPDBAS")
tcplLoadChemList(field = "chid", val = 20182)
tcplLoadChemList(field = "dsstox_substance_id", val = "DTXSID7020182")

## End(Not run)
```

---

tcplLoadConcUnit	<i>Load concentration units for assay endpoints</i>
------------------	---

---

**Description**

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

**Usage**

```
tcplLoadConcUnit(spId)
```

**Arguments**

spid	Integer, assay endpoint ids
------	-----------------------------

**Value**

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

**See Also**

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

---

tcplLoadData	<i>Load tcpl data</i>
--------------	-----------------------

---

**Description**

tcplLoadData queries the tcpl databases and returns a data.table with data for the given level and data type.

**Usage**

```
tcplLoadData(lvl, fld = NULL, val = NULL, type = "mc", add.fld = TRUE)
```

**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"
add.fld	Boolean if true we want to return the additional parameters fit with tcplfit2

## 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.

If tcp1Conf() was set with "API" as the driver, then tcp1LoadData will return data from the CCTE Bioactivity API. API data is available for type = 'mc' and lvl = c(3,4,5,6) and 'agg'. Only fields relating to the requested level are returned, but not all fields that usually return from invitrodb are available from the API. To have all fields available from the API return, regardless of what lvl is set to, set add.fld to TRUE. API query-able fields include "aeid", "spid", "m4id", and "dtxsid".

Leaving fld NULL will return all data.

Valid fld inputs are based on the data level and type:

type	lvl	Queried tables
sc	0	sc0
sc	1	sc0, sc1
sc	agg	sc1, sc2_agg
sc	2	sc2
mc	0	mc0
mc	1	mc0, mc1
mc	2	mc0, mc1, mc2
mc	3	mc0, mc1, mc3
mc	agg	mc3, mc4_agg
mc	4	mc4
mc	5	mc4, mc5
mc	6	mc4, mc6
mc	7	mc4, mc7

## Value

A data.table containing data for the given fields.

## See Also

[tcp1Query](#), [data.table](#)

## Examples

```
## Not run:
## Load all of level 0 for multiple-concentration data, note 'mc' is the
## default value for type
tcp1LoadData(lvl = 0)

## Load all of level 1 for single-concentration
tcp1LoadData(lvl = 1, type = "sc")

## List the fields available for level 1, coming from tables mc0 and mc1
```



```
tcplListFlds(tbl = "mc0")
tcplListFlds(tbl = "mc1")

## Load level 0 data where the well type is "t" and the concentration
## index is 3 or 4
tcplLoadData(lvl = 1, fld = c("wllt", "cndx"), val = list("t", c(3:4)))

## End(Not run)
```

---

tcplLoadUnit                    *Load response units for assay endpoints*

---

### 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](#)

---

tcplLvlCount                    *Load tcpl level counts*

---

### Description

tcplLvlCount queries the tcpl databases and returns a data frame with count totals for the given levels and data type.

### Usage

```
tcplLvlCount(lvls = NULL, type = "mc")
```

**Arguments**

lvls            Integer or list of Integers, The levels of data to load  
type            Character of length 1, the data type, "sc" or "mc"

**Details**

The data type can be either 'mc' for mutiple concentration data, or 'sc' for single concentration data.

Leaving lvls NULL will return all data.

**Value**

A data.table containing data for the given fields.

**See Also**

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

**Examples**

```
## Not run:  
## Get all counts for level 1 for multiple-concentration  
tcplLvlCount(lvls = 1)  
  
## Get all counts for levels 4 through 7 for multiple-concentration  
tcplLvlCount(lvls = 4:7)  
  
## Get all counts for multiple-concentration data, note 'mc' is the  
## default value for type  
tcplLvlCount()  
  
## End(Not run)
```

---

tcplMakeAeidMultiPlts *Create a .pdf with all dose-response plots for a given aeid, 6 per page*

---

**Description**

tcplMakeAeidMultiPlts Create a .pdf with all dose-response plots for a given aeid

**Usage**

```
tcplMakeAeidMultiPlts(  
  aeid,  
  lvl = 4L,  
  fname = NULL,  
  odir = getwd(),
```

```
    hitc.all = TRUE
  )
```

### Arguments

aeid	Integer of length 1, the assay endpoint id
lvl	Integer of length 1, the data level to use (4-7)
fname	Character, the filename
odir	The directory to save the .pdf file in
hitc.all	If FALSE, only plots with hitc==1 will be displayed

### Details

tcplMakeAeidMultiPlts provides a wrapper for [tcplMultiplot](#), allowing the user to produce PDFs with the curve plots without having to separately load all of the data and establish the PDF device.

If 'fname' is NULL, a default name is given by concatenating together assay information.

---

tcplMakeAeidPlts	<i>Create a .pdf with dose-response plots</i>
------------------	---

---

### Description

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

### Usage

```
tcplMakeAeidPlts(
  aeid,
  compare = F,
  lvl = 4L,
  fname = NULL,
  odir = getwd(),
  ordr.fitc = TRUE,
  cnst = NULL
)
```

### Arguments

aeid	Integer of length 1 or 2, the assay endpoint id
compare	Boolean to for comparison of aeids if length(aeid)>1
lvl	Integer of length 1, the data level to use (4-7). Only level 5-6 valid for compare aeids.
fname	Character, the filename
odir	The directory to save the .pdf file in
ordr.fitc	Logical, should the fits be ordered by fit category?
cnst	Constant hline to draw on plot

**Details**

tcplMakeAeidPlts provides a wrapper for [tcplPlotFits](#), allowing the user to produce PDFs with the curve plots without having to separately load all of the data and establish the PDF device.

If 'fname' is NULL, a default name is given by concatenating together assay information.

Note, the default value for `ordr.fitc` is TRUE in `tcplMakeAeidPlts`, but FALSE in `tcplPlotFits`

Note, only level 5 or level 6 is valid for comparing 2 auids.

**Examples**

```
## Not run:
## Will produce the same result as the example for tcplPlotFits
tcplMakeAeidPlts(aeid = 1, lvl = 6, ordr.fitc = FALSE)

## End(Not run)

## Not run:
## Compare two auids on same plots
tcplMakeAeidPlts(aeid = c(1,2), compare=T, lvl = 6)

## End(Not run)
```

---

tcplMakeChidMultiPlts *Create a .pdf with all dose-response plots for a given chid, 6 per page*

---

**Description**

tcplMakeChidMultiPlts Create a .pdf with all dose-response plots for a given chid

**Usage**

```
tcplMakeChidMultiPlts(
  chid,
  lvl = 4L,
  fname = NULL,
  odir = getwd(),
  hitc.all = TRUE
)
```

**Arguments**

chid	Integer of length 1, the chemical id
lvl	Integer of length 1, the data level to use (4-7)
fname	Character, the filename
odir	The directory to save the .pdf file in
hitc.all	If FALSE, only plots with <code>hitc==1</code> will be displayed

**Details**

tcplMakeChidMultiPlts provides a wrapper for [tcplMultiplot](#), allowing the user to produce PDFs with the curve plots without having to separately load all of the data and establish the PDF device.

If 'fname' is NULL, a default name is given by concatenating together assay information.

---

 tcplMultiplot

*Plot summary fits based on fit and dose-response data*


---

**Description**

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

**Usage**

```
tcplMultiplot(dat, agg, flg = NULL, boot = NULL, browse = FALSE, hitc.all)
```

**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.
boot	data.table, level 7 data, see details.
browse	Logical, should browser() be called after every plot?
hitc.all	Logical, if FALSE, only plots with hitc==1 will be displayed

**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.

If dat contains only one aeid, plots will be ordered by chemical name (chnm). Otherwise, plots are ordered by assay endpoint name (aenm). ## While it is most likely the user will want to just save all of the plots ## to view in a PDF, the 'browse' parameter can be used to quickly view ## some plots.

---

tcplPlot	#----- <i>Plotting Function for tcpl</i>	<i>Generic</i>
----------	---	----------------

---

## Description

tcplPlot queries the tcpl databases and returns a plot for the given level and data type.

## Usage

```
tcplPlot(
  dat = NULL,
  type = "mc",
  fld = "m4id",
  val = NULL,
  compare.val = NULL,
  by = NULL,
  output = c("console", "pdf", "png", "jpg", "svg", "tiff"),
  fileprefix = paste0("tcplPlot_", Sys.Date()),
  multi = NULL,
  verbose = FALSE,
  nrow = NULL,
  ncol = NULL,
  dpi = 600,
  flags = FALSE,
  yuniform = FALSE,
  yrange = c(NA, NA)
)
```

## Arguments

dat	data.table containing plot-prepared data, used for stand-alone (non-ToxCast data like other tcplfit2-fit data) or advanced plotting (generating comparison plots across multiple database configurations) and not required. See tcplPlotLoadData.
type	Character of length 1, the data type, "sc" or "mc".
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'.
compare.val	List, vectors of values for each field to query on to compare with val. Must be in the same order as 'fld'. Must have the same length as val (1:1 comparison). Must be set to compare plots; otherwise leave NULL
by	Parameter to divide files into e.g. "aeid".
output	How should the plot be presented. To work with the plot in environment, use "ggplot"; to interact with the plot in application, use "console"; or to save as a file type, use "pdf", "jpg", "png", "svg", or "tiff".

fileprefix	Prefix of file when saving.
multi	Boolean, by default TRUE for "pdf". If multi is TRUE, output by default 4 plots per page for 'verbose' = TRUE and 6 plots per page for 'verbose' = FALSE.
verbose	Boolean, by default FALSE. If TRUE, a table with fitting parameters is included with the plot.
nrow	Integer, number of rows in multiplot. By default 2.
ncol	Integer, number of columns in multiplot. By default 3, 2 if verbose.
dpi	Integer, image print resolution. By default 600.
flags	Boolean, by default FALSE. If TRUE, level 6 flags are displayed below annotations on plot
yuniform	Boolean, by default FALSE. If TRUE, all plots will have uniform y axis scaling, automatically determined.
yrange	Integer of length 2, for directly setting the y-axis range, c(<min>,<max>). By default, c(NA,NA).

### 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.

Leaving fld NULL will return all data.

### Examples

```
## Not run:
tcplPlot(fld = "m4id", val = c(18609966)) ## Create a level 4 plot

## End(Not run)
```

---

tcplPlotFitc	<i>Plot the fit category tree</i>
--------------	-----------------------------------

---

### 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**

Suggested device size (inches): width = 10, height = 7.5, pointsize = 9

**Examples**

```
## Not run:
## Plot visualization of fit categories for all level 5 data
tcpIPlotFitc(fitc = tcpILoadData(5)$fitc)

## End(Not run)
```

---

tcpIPlotFits

*Plot summary fits based on fit and dose-response data*


---

**Description**

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

**Usage**

```
tcpIPlotFits(
  dat,
  agg,
  flg = NULL,
  boot = NULL,
  ordr.fitc = FALSE,
  browse = FALSE,
  cnst = NULL,
  orig.aeid = NULL,
  compare = F
)
```

**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.
boot	data.table, level 7 data, see details.
ordr.fitc	Logical, should the fits be ordered by fit category?
browse	Logical, should browser() be called after every plot?
cnst	Constant hline to draw on plot
orig.aeid	Original aeid list from tcpIMakeAeidPlts to maintain order
compare	boolean to determine if aeids should be compared on same 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.

Supplying level 4 data for the 'dat' parameter will result in level 4 plots. Similarly, supp

If fits are not ordered by fit category, they will be ordered by chemical ID. Inputs with multiple assay endpoints will first be ordered by assay endpoint ID.

## Examples

```
## Not run:
## tcplPlotFits needs data.tables supplying the concentration/response
## data stored in mc4_agg, as well as the fit information from mc4 or mc5.
## Additionally, tcplPlotFits can take level 6 data from mc6 and add the
## flag information to the plots. The following shows how to make level 5
## plots. Adding the 'flg' parameter would result in level 6 plots, and
## loading level 4, rather than level 5 data, would result in level 4 plots.

15 <- tcplLoadData(lvl = 5, fld = "m4id", val = 18609966)
14_agg <- tcplLoadData(lvl = "agg", fld = "m4id", val = 18609966)

pdf(file = "tcplPlotFits.pdf", height = 6, width = 10, pointsize = 10)
tcplPlotFits(dat = 15, agg = 14_agg)
graphics.off()

## While it is most likely the user will want to just save all of the plots
## to view in a PDF, the 'browse' parameter can be used to quickly view
## some plots.

## Start by identifying some sample IDs to plot, then call tcplPlotFits with
## a subset of the data. This browse function is admittedly clunky.
bpa <- tcplLoadChem(field = "chnm", val = "Bisphenol A")[ , spid]
15_sub <- 15[spid %in% bpa]
tcplPlotFits(dat = 15_sub,
             agg = 14_agg[m4id %in% 15_sub$m4id],
             browse = TRUE)

## End(Not run)
```

---

`tcplPlotLoadData`

*Utility function to load data for tcplPlot*

---

## Description

`tcplPlotLoadData` queries the `tcpl` databases and returns a `data.table` with data for the given field, value, level, and data type prepared in a format `tcplPlot` can use to generate plots.

**Usage**

```
tcplPlotLoadData(type = "mc", fld = "m4id", val, flags = FALSE)
```

**Arguments**

type	Character of length 1, the data type, "sc" or "mc"
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'.
flags	Boolean, by default FALSE. If TRUE, level 6 flags are loaded for use in tcplPlot. Must be set to TRUE if tcplPlot 'flags' also is/will be set to TRUE

**Details**

This utility function is used by tcplPlot to load and prepare data from tcplLoadData for use in generating plots. It is exported for use in advanced comparison plots where users create plots using multiple data sources. After saving the response from tcplPlotLoadData, switch data source config and pass the data to tcplPlot dat parameter.

The data type can be either 'mc' for multiple concentration data, or 'sc' for single concentration data.

**Value**

A data.table containing plot-ready data for the given fields.

**See Also**

[tcplPlot](#)

**Examples**

```
## Not run:
## load mc plot data for an entire endpoint
dat <- tcplPlotLoadData(fld = "aeid", val = 703)

## load sc plot data for an entire endpoint
dat <- tcplPlotLoadData(type = "sc", fld = "aeid", val = 703)

## load plot data for two endpoint-samples and include loading of flags
## flags must equal TRUE if tcplPlot will/does
dat <- tcplPlotLoadData(fld = c("spid", "aeid"),
                       val = list(c("TP0000269F11", "TP0000395A09"),703),
                       flags = TRUE)

## if desired, switch connections
tcplConf()

## use dat in tcplPlot
tcplPlot(dat = dat,
```

```

fld = c("spid", "aeid"),
val = list(c("TP0000269F11", "TP0000395A09"),703),
compare.val = list(c("LEGTV002B01", "LEGTV003A06"),703),
output = "pdf",
flags = TRUE,
fileprefix="example")

## End(Not run)

```

---

tcplPlotlyPlot	<i>tcplPlotlyPlot</i>
----------------	-----------------------

---

### Description

tcplPlotlyPlot

### Usage

```
tcplPlotlyPlot(dat, lvl = 5)
```

### Arguments

dat	data table with all required conc/resp data
lvl	integer level of data that should be plotted level 2 - for 'sc' plotting level 5 - for 'mc' plotting, all fit models and winning model with hitcall

### Value

A plotly plot

---

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

**Details**

A level 4 plot ('lvl' = 4) will plot the concentration series and the applicable curves, without an indication of the activity call or the winning model. Level 4 plots can be created without having done subsequent processing.

Level 5 plots include the level 4 information with the activity call and model selection. The winning model will be highlighted red in the side panel containing the summary statistics. Level 6 plots, in addition to all of the level 4 and 5 information, include the positive flag IDs. If the flag has an associated value, the value will be in parentheses following the flag ID. Level 7 plots in addition to all of the level 4, 5, and 6 information, include the AC50 confidence interval and hit percentage information from bootstrapping.

**See Also**

[tcplPlotFits](#), [tcplMakeAeidPlts](#)

**Examples**

```
## Not run:
tcplPlotM4ID(m4id = 18609966, lvl = 4) ## Create a level 4 plot
tcplPlotM4ID(m4id = 18609966, lvl = 5) ## Create a level 5 plot
tcplPlotM4ID(m4id = 18609966, lvl = 6) ## Create a level 6 plot

## End(Not run)
```

---

tcplPlotPlate

*Plot plate heatmap*

---

**Description**

tcplPlotPlate generates a heatmap of assay plate data

**Usage**

```
tcplPlotPlate(dat, apid, id = NULL, quant = c(0.001, 0.999))
```

**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.
quant	Numeric vector, the range of data to include in the legend

**Details**

The legend represents the range of the data supplied to `dat`, for the applicable ID. The additional horizontal lines on the legend indicate the range of the plotted plate, to show the relation of the plate to the assay as a whole. A plot with a legend specific for the given `apid` can be created by only supplying the data for the `apid` of interest to `'dat'`.

The `quant` parameter, by default including 99.8 allows for extreme outliers without losing resolution. Outliers in either direction will be highlighted with a dark ring, as seen in the example. A NULL value for `'quant'` will not restrict the data at all, and will use the full range for the legend.

Wells with a well quality of 0 (only applicable for level 1 plots), will have an "X" through their center.

**Note**

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

**Examples**

```
## Not run:
d1 <- tcplLoadData(lvl = 1, fld = "acid", val = 1)
tcplPlotPlate(dat = d1, apid = "09Apr2014.Plate.17")

## End(Not run)
```

---

<code>tcplPlotSetYRange</code>	<i>tcplPlotSetYRange</i>
--------------------------------	--------------------------

---

**Description**

`tcplPlotSetYRange`

**Usage**

```
tcplPlotSetYRange(dat, yuniform, yrange, type)
```

**Arguments**

<code>dat</code>	dataset
<code>yuniform</code>	should the yrange be uniform
<code>yrange</code>	length 2 of the yrange
<code>type</code>	mc or sc

**Value**

yrange of the data

---

tcplPlotValidate	<i>tcplPlotValidate</i>
------------------	-------------------------

---

**Description**

tcplPlotValidate

**Usage**

```
tcplPlotValidate(
  type = "mc",
  flags = NULL,
  output = "none",
  multi = NULL,
  verbose = FALSE
)
```

**Arguments**

type	string of mc or sc indicating if it is single or multi conc
flags	bool - should we return flags
output	how should the plot be formatted
multi	are there multiple plots
verbose	should the plot return a table with parameters

**Value**

a list of validated parameters for plotting

---

tcplPrep0tpt	<i>Map assay/chemical ID values to annotation information</i>
--------------	---

---

**Description**

tcplPrep0tpt queries the chemical and assay information from the tcpl database, and maps the annotation information to the given data.

**Usage**

```
tcplPrep0tpt(dat, ids = NULL)
```

**Arguments**

dat	data.table, output from <a href="#">tcplLoadData</a>
ids	Character, (optional) a subset of ID fields to map

## Details

tcplPrep0tpt is used to map chemical and assay identifiers to their respective names and annotation information to create a human-readable table that is more suitable for an export/output.

By default the function will map sample ID (spid), assay component id (acid), and assay endpoint ID (aeid) values. However, if 'ids' is not null, the function will only attempt to map the ID fields given by 'ids.'

## Value

The given data.table with chemical and assay information mapped

## Examples

```
## Not run:
## Load some example data
d1 <- tcplLoadData(1)

## Check for chemical name in 'dat'
"chnm" %in% names(d1) ## FALSE

#' ## Map all annotations
d2 <- tcplPrep0tpt(d1) ##
"chnm" %in% names(d2) ## TRUE
"acnm" %in% names(d2) ## TRUE

## Map chemical annotation only
d3 <- tcplPrep0tpt(d1, ids = "spid")
"chnm" %in% names(d3) ## TRUE
"acnm" %in% names(d3) ## FALSE

## End(Not run)
```

---

tcplRun

*Perform data processing*

---

## Description

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

## Usage

```
tcplRun(
  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 endpoint 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.

---

 tcp1SubsetChid

*Subset level 5 data to a single sample per chemical*


---

### Description

tcp1SubsetChid subsets level 5 data to a single tested sample per chemical. In other words, if a chemical is tested more than once (a chid has more than one spid) for a given assay endpoint, the function uses a series of logic to select a single "representative" sample.



**Usage**

```
tcplSubsetChid(dat, flag = TRUE, type = "mc", export_ready = FALSE)
```

**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
type	Character of length 1, the data type, "sc" or "mc"
export_ready	Boolean, default FALSE, should only export ready 1 values be included in calculation

**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 AC50 (or 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](#)

**Examples**

```
## Not run:
## Load the example level 5 data
d1 <- tcplLoadData(lvl = 5, fld = "aeid", val = 797)
d1 <- tcplPrep0tpt(d1)

## Subset to an example of a duplicated chid
d2 <- d1[chid == 20182]
d2[, list(m4id, hitc, fitc, modl_ga)]

## Here the consensus hit-call is 1 (active), and the fit categories are
## all equal. Therefore, if the flags are ignored, the selected sample will
## be the sample with the lowest modl_ga.
tcplSubsetChid(dat = d2, flag = FALSE)[, list(m4id, modl_ga)]

## End(Not run)
```

---

 tcpIVarMat

*Create chemical by assay matrices*


---

## Description

tcpIVarMat creates chemical by assay matrices.

## Usage

```
tcpIVarMat(
  dsstox_substance_id = NULL,
  aeid = NULL,
  add.vars = NULL,
  flag = TRUE
)
```

## Arguments

dsstox_substance_id	Integer, chemical ID values to subset on
aeid	Integer, assay endpoint ID values to subset on
add.vars	Character, mc4 or mc5 field(s) not included in the standard list to add additional matrices
flag	Integer or Logical of length 1, passed to <a href="#">tcpISubsetChid</a>

## Details

The tcpIVarMat function is used to create chemical by assay matrices for different parameters. The standard list of matrices returned includes:

1. "ac50" – The active concentration at 50 the winning model.
2. "ac50\_verbose" – The AC50 for the winning model, with text describing some situations.
3. "acc" – The active concentration at user-defined cutoff for the winning model.
4. "acc\_verbose" – The ACC for the winning model, with text describing some situations.
5. "mc\_hitc" – The hit-call for the winning model in multiple-concentration (mc) screening.
6. "sc\_hitc" – The hit-call in single concentration (sc) screening.

tcpIVarMat produces matrices of combined sc-mc output. For the ac50 and acc matrices specifically, values are inserted in place to show complete views of what was tested and what the results were. ac50 and acc values are:

- set to 1e6 when the chemical is tested but negative in mc. In \_verbose matrices, these are indicated as "MC neg".

- set to 1e7 when the chemical is not tested in mc but was screened in sc with a positive hitcall for the same aeid. In `_verbose` matrices, these are indicated as "SC pos, No MC".
- set to 1e8 when the chemical is not tested in mc but was screened in sc with a negative hitcall for the same aeid. In `_verbose` matrices, these are indicated as "SC neg, No MC"
- not changed when chemical is tested in mc and positive, or not tested in either mc or sc

sc and mc data both are currently required to be included for these calculations. As a result, the "API" driver is not currently supported since it does not return sc data.

To add additional matrices, the `'add.vars'` parameter can be used to specify the fields from the mc4 or mc5 tables to create matrices for.

When more than one sample is included for a chemical/assay pair, `tcplVarMat` aggregates multiple samples to a chemical level call utilizing `tcplSubsetChid`. The input for the `tcplVarMat` `'flag'` parameter is passed to the `tcplSubsetChid` call and used to parse down the data to create the matrices.

## Value

A list of chemical by assay matrices (`data.tables`) where the rows are given by the `dsstox_substance_id` and corresponding `chnm` (chemical name) columns and the colnames are given by assay endpoint name (`aenm`).

## See Also

[tcplSubsetChid](#)

## Examples

```
## Not run:
## Demonstrate the returned values.
varmat <- tcplVarMat()

## Other changes can be made
aeids <- c(80)
dtxsid <- c("DTXSID4034653", "DTXSID2032683", "DTXSID6032358",
"DTXSID0032651", "DTXSID8034401")
varmat <- tcplVarMat(aeid = aeids, dsstox_substance_id = dtxsid)
varmat <- tcplVarMat(aeid = aeids, add.vars = c("m4id", "resp_max", "max_med"))

## To save output to file
library(writexl)
write_xlsx(varmat, path = "varmat_output.xlsx")

## End(Not run)
```

---

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 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.

### Note

This function is not exported and is not intended 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](#)

---

test_api	<i>List containing ids used for different automated tests of tcpl integration with the CTX APIs, randomly selected from what is available via API.</i>
----------	--

---

## Description

List containing ids used for different automated tests of tcpl integration with the CTX APIs, randomly selected from what is available via API.

## Usage

```
test_api
```

## Format

A list with 7 items:

- acid** Randomly selected assay component endpoint id
- acid** Assay component id associated with the above acid
- aid** Assay id associated with the above acid

**asid** Assay source id associated with the above aeid  
**dtxsid** dsstox substance id of one sample from the above aeid  
**spid** Sample id of one (the same) sample from the above aeid  
**m4id** Level 4 id of one (the same) sample from the above aeid

**Source**

CTX Bioactivity API

---

write_lvl_4	<i>Write level 4 with updated schema</i>
-------------	--

---

**Description**

Write level 4 with updated schema

**Usage**

write\_lvl\_4(dat)

**Arguments**

dat                    output of tcplfit2 that has been unnested into a data.table

# Index

- \* **data processing functions**
  - tcplRun, [95](#)
- \* **datasets**
  - invitrodb\_dd, [12](#)
  - load\_data\_columns, [14](#)
  - mc\_test, [32](#)
  - mc\_vignette, [35](#)
  - mcdat, [31](#)
  - mthd\_list\_defaults, [47](#)
  - sc\_test, [57](#)
  - sc\_vignette, [59](#)
  - smdat, [57](#)
  - test\_api, [101](#)
- \* **multiple-concentration**
  - mc1, [16](#)
  - mc2, [17](#)
  - mc3, [19](#)
  - mc4, [24](#)
  - mc5, [26](#)
  - mc6, [29](#)
- \* **single-concentration**
  - sc1, [51](#)
  - sc2, [54](#)
- \* **tcpl abbreviations**
  - is.odd, [12](#)
  - lu, [15](#)
  - lw, [16](#)
  - sink.reset, [61](#)
- .buildAssayQ, [4](#)
- .convertNames, [5](#)
- .load6DR, [5](#)
- .plateHeat, [6](#)
- .prepField, [6](#)
- AIC, [63](#)
- blineShift, [7](#)
- check\_tcpl\_db\_schema, [7](#)
- Configure functions, [8](#)
- constrOptim, [44, 71](#)
- data.table, [79–82](#)
- flareFunc, [9](#)
- Hill model utilites, [10](#)
- interlaceFunc, [11](#)
- invitrodb\_dd, [12](#)
- is.odd, [12, 15, 16, 61](#)
- length, [16](#)
- Load assay information, [13](#)
- load\_data\_columns, [14](#)
- lu, [13, 15, 16, 61](#)
- lw, [13, 15, 16, 61](#)
- mc1, [16, 18, 20, 24, 26, 30](#)
- mc2, [17, 17, 19, 20, 24, 26, 30](#)
- MC2\_Methods, [18, 18](#)
- mc2\_mthds (MC2\_Methods), [18](#)
- mc3, [7, 17, 18, 19, 24, 26, 30](#)
- MC3\_Methods, [20, 20](#)
- mc3\_mthds, [7](#)
- mc3\_mthds (MC3\_Methods), [20](#)
- mc4, [17, 18, 20, 24, 26, 30](#)
- MC4\_Methods, [25](#)
- mc4\_mthds (MC4\_Methods), [25](#)
- mc5, [17, 18, 20, 24, 26, 29, 30](#)
- MC5\_Methods, [26, 27](#)
- mc5\_mthds (MC5\_Methods), [27](#)
- mc6, [10, 12, 17, 18, 20, 24, 26, 29, 31](#)
- MC6\_Methods, [10, 12, 30, 30](#)
- mc6\_mthds (MC6\_Methods), [30](#)
- mc\_test, [32](#)
- mc\_vignette, [35](#)
- mcdat, [31](#)
- mclapply, [96](#)
- Method functions, [43](#)
- Models, [24, 44, 62](#)

- mthd\_list\_defaults, 47
- Query functions, 47
- Register/update annotation, 49
- registerMthd, 50
- sc1, 51, 53, 54
- SC1\_Methods, 51, 52
- sc1\_mthds (SC1\_Methods), 52
- sc2, 51, 54, 56
- SC2\_Methods, 54, 55
- sc2\_mthds (SC2\_Methods), 55
- sc\_test, 57
- sc\_vignette, 59
- smdat, 57
- sink, 61
- sink.number, 61
- sink.reset, 13, 15, 16, 61
- Startup, 9
- Sys.getenv, 9
- tcplAddModel, 62
- tcplAICProb, 63
- tcplAppend, 64, 100, 101
- tcplCascade, 16, 17, 19, 24, 26, 29, 51, 54, 64, 100, 101
- tcplCode2CASN, 65
- tcplConf (Configure functions), 8
- tcplConfDefault (Configure functions), 8
- tcplConfList (Configure functions), 8
- tcplConfLoad (Configure functions), 8
- tcplConfReset (Configure functions), 8
- tcplConfSave (Configure functions), 8
- tcplCytPt, 66
- tcpldbStats, 68
- tcplDefine, 68
- tcplDelete, 69
- tcplFit, 24, 44, 63, 70
- tcplFit2, 71
- tcplFit2\_nest, 72
- tcplFit2\_unnest, 72
- tcplGetAeid, 73
- tcplggplot, 73
- tcplggplotCompare, 74
- tcplHillACXX (Hill model utilites), 10
- tcplHillConc (Hill model utilites), 10
- tcplHillVal (Hill model utilites), 10
- tcplHit2, 75
- tcplLegacyPlot, 75
- tcplListFlds, 76
- tcplLoadAcid (Load assay information), 13
- tcplLoadAeid (Load assay information), 13
- tcplLoadAid (Load assay information), 13
- tcplLoadAsid (Load assay information), 13
- tcplLoadChem, 76
- tcplLoadChemList, 77
- tcplLoadConcUnit, 79
- tcplLoadData, 79, 85, 89, 94
- tcplLoadUnit, 81
- tcplLvlCount, 81
- tcplMakeAeidMultiPlts, 82
- tcplMakeAeidPlts, 83, 92
- tcplMakeChidMultiPlts, 84
- tcplMthdAssign (Method functions), 43
- tcplMthdClear (Method functions), 43
- tcplMthdList (Method functions), 43
- tcplMthdLoad (Method functions), 43
- tcplMultiplot, 83, 85, 85
- tcplObjCnst, 71
- tcplObjCnst (Models), 44
- tcplObjGnls, 71
- tcplObjGnls (Models), 44
- tcplObjHill, 71
- tcplObjHill (Models), 44
- tcplPlot, 86, 90
- tcplPlotFitc, 87
- tcplPlotFits, 62, 84, 88, 91, 92
- tcplPlotLoadData, 89
- tcplPlotlyPlot, 91
- tcplPlotM4ID, 91
- tcplPlotPlate, 92
- tcplPlotSetYRange, 93
- tcplPlotValidate, 94
- tcplPrepOtpt, 94, 97
- tcplQuery, 79–82
- tcplQuery (Query functions), 47
- tcplQueryAPI (Query functions), 47
- tcplRegister (Register/update annotation), 49
- tcplRun, 16, 17, 19, 24, 26, 29, 51, 54, 95
- tcplSendQuery, 70
- tcplSendQuery (Query functions), 47
- tcplSubsetChid, 66, 96, 98, 99



tcplUpdate (Register/update  
annotation), [49](#)

tcplVarMat, [98](#)

tcplWriteData, [100](#)

tcplWriteLvl0, [100](#), [100](#)

test\_api, [101](#)

unique, [15](#)

which, [15](#), [16](#)

write\_lvl\_4, [102](#)