

# Package ‘toxEval’

July 22, 2025

**Type** Package

**Title** Exploring Biological Relevance of Environmental Chemistry  
Observations

**Version** 1.4.0

**Description** Data analysis package for estimating potential biological effects from chemical concentrations in environmental samples. Included are a set of functions to analyze, visualize, and organize measured concentration data as it relates to user-selected chemical-biological interaction benchmark data such as water quality criteria. The intent of these analyses is to develop a better understanding of the potential biological relevance of environmental chemistry data. Results can be used to prioritize which chemicals at which sites may be of greatest concern. These methods are meant to be used as a screening technique to predict potential for biological influence from chemicals that ultimately need to be validated with direct biological assays. A description of the analysis can be found in Blackwell (2017) <[doi:10.1021/acs.est.7b01613](https://doi.org/10.1021/acs.est.7b01613)>.

**License** CC0

**Copyright** This software is in the public domain because it contains materials that originally came from the United States Geological Survey, an agency of the United States Department of Interior. For more information, see the official USGS copyright policy at [https://www.usgs.gov/visual-id/credit\\_usgs.html#copyright](https://www.usgs.gov/visual-id/credit_usgs.html#copyright)

**Depends** R (>= 4.1.0)

**Imports** dplyr, tidyr, DT (>= 0.1.24), leaflet (>= 1.0.0), ggplot2 (>= 3.0.0), magrittr, shiny, shinydashboard, RColorBrewer, readxl, tools, shinyAce, shinycssloaders

**Suggests** rmarkdown, testthat, knitr, here, tcpl, openxlsx, covr

**BugReports** <https://github.com/DOI-USGS/toxEval/issues>

**VignetteBuilder** knitr

**BuildVignettes** true

**LazyLoad** yes

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Laura DeCicco [aut, cre] (ORCID:  
    <<https://orcid.org/0000-0002-3915-9487>>),  
Steven Corsi [aut] (ORCID: <<https://orcid.org/0000-0003-0583-5536>>),  
Daniel Villeneuve [aut] (ORCID:  
    <<https://orcid.org/0000-0003-2801-0203>>),  
Brett Blackwell [aut] (ORCID: <<https://orcid.org/0000-0003-1296-4539>>),  
Gerald Ankley [aut] (ORCID: <<https://orcid.org/0000-0002-9937-615X>>),  
Alison Appling [rev] (Reviewed for USGS),  
Dalma Martinovic [rev] (Reviewed for USGS)

**Maintainer** Laura DeCicco <ldecicco@usgs.gov>

**Repository** CRAN

**Date/Publication** 2024-11-22 21:20:02 UTC

Contents

as.toxEval . . . . .	3
clean_endPoint_info . . . . .	3
create_toxEval . . . . .	4
endpoint_hits_DT . . . . .	5
end_point_info . . . . .	7
explore_endpoints . . . . .	8
filter_groups . . . . .	8
flags . . . . .	9
get_ACC . . . . .	9
get_chemical_summary . . . . .	10
get_concentration_summary . . . . .	11
graph_chem_data . . . . .	13
hits_by_groupings_DT . . . . .	15
hits_summary_DT . . . . .	16
make_tox_map . . . . .	18
plot_chemical_boxplots . . . . .	19
plot_tox_endpoints . . . . .	22
plot_tox_endpoints2 . . . . .	24
plot_tox_heatmap . . . . .	25
plot_tox_stacks . . . . .	28
rank_sites_DT . . . . .	30
remove_flags . . . . .	32
summary.toxEval . . . . .	33
ToxCast_ACC . . . . .	33
tox_chemicals . . . . .	34
<b>Index</b>	<b>35</b>

---

as.toxEval	<i>toxEval helper functions</i>
------------	---------------------------------

---

**Description**

A small collection of helper functions for toxEval

**Usage**

```
as.toxEval(x, ...)
```

**Arguments**

x	list or toxEval object
...	data frames to override data within the original x list.

**Examples**

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
full_path <- file.path(path_to_tox, file_name)
tox_list <- create_toxEval(full_path)

# To over-ride one of the data frames:
chem_data <- data.frame(
  CAS = "21145-77-7",
  Value = 1,
  "Sample Date" = as.Date("2012-01-01"),
  SiteID = "USGS-04249000"
)
tox_list_new <- as.toxEval(tox_list, chem_data)
```

---

clean_endPoint_info	<i>clean_endPoint_info</i>
---------------------	----------------------------

---

**Description**

As of ToxCast 4.1, this function only helps clean up abbreviations found in the end\_point\_info data frame.

**Usage**

```
clean_endPoint_info(end_point_info)
```

## Arguments

end\_point\_info Data frame Endpoint information from ToxCast.

## Value

The returned data frame is based on end\_point\_info, but with some endPoints filtered out and some additional categories in intended\_target\_family and intended\_target\_family\_sub. The names in intended\_target\_family are revised to look more appealing in graphs and tables.

## Examples

```
end_point_info <- end_point_info
nrow(end_point_info)
cleaned_ep <- clean_endPoint_info(end_point_info)
nrow(cleaned_ep)
```

---

create_toxEval	<i>Load and check toxEval data</i>
----------------	------------------------------------

---

## Description

This function is used to load a data file for analysis in the form of a single Excel file. The Excel file should include 3 mandatory sheets named "Data", "Chemicals", and "Sites". Additionally there are 2 optional sheets: "Exclude" and "Benchmarks". This function creates a data frame for each sheet, perform basic checks on the data to assure that required columns are included for each sheet

## Usage

```
create_toxEval(excel_file_path, ...)
```

## Arguments

excel_file_path	Path to Excel file that contains at least 3 sheets: Data, Chemicals, and Sites, and could optionally contain Exclude and Benchmarks.
...	data frames to override data within the original x list.

## Details

Required columns in the Data sheet include "CAS", "SiteID", "Value", and "Sample Date". The "Value" column includes concentration measurements in units of  $\mu\text{g/L}$ . "Sample Date" can be either a date or date/time or an integer. Additional columns may be included for user purposes, but will not be used in toxEval.

Required columns in the Chemical sheet include "CAS", "Class". "CAS" values in this sheet must exactly match corresponding "CAS" values in the Data sheet. The "Class" designation allows data to be grouped in a user-specified way. For example, in a data set of multiple pesticides, it may

be valuable to explore differences and similarities to of insecticides, herbicides and fungicides. Additional columns may be included for user purposes, but will not be used in toxEval.

Required columns in the Sites sheet includes "SiteID", "Short Name", and for the Shiny application "dec\_lat", "dec\_lon". Values in the "SiteID" column in this sheet exactly match corresponding values in the "SiteID" column in the Data sheet. Additional columns may be included for user purposes, but will not be used in toxEval.

When using the optional sheet Exclude, columns required include "CAS" and "endPoint". These are used to exclude particular chemicals (via CAS), ToxCast endpoints (via endPoint), or a unique chemical/endpoint combination. Additional columns may be included for user purposes, but will not be used in toxEval.

When using the optional sheet Benchmarks, columns required include "CAS", "endPoint", "ACC\_value" and "chnm". This sheet is used to over-ride the functions using endpoints from the ToxCast database, allowing the user to import endpoint information from other sources. It could also be useful for reproducing results in the future (for example, if after ToxCast updates, analysis with an older version of ToxCast may be reproduced by including the selected ToxCast endpoint database in this sheet. Additional columns may be included for user purposes, but will not be used in toxEval.

For more information, see the "Prepare Data" vignette: `vignette("PrepareData", package = "toxEval")`.

All remaining toxEval functions use data from via the list that is returned from this function.

## Value

The object returned from this function contains a list of between three and five data frames. The minimum data frames returned are chem\_data (containing at least the columns: "CAS", "SiteID", "Value", "Sample Date"), chem\_info (containing at least the columns: "CAS", "Class"), and chem\_site (containing at least the columns: "SiteID", "Short Name". For the Shiny app, "dec\_lat" and "dec\_lon" are also required). The optional data frames are exclusions (containing at least the columns: "CAS" and "endPoint"), and benchmarks (containing at least the columns: "CAS", "endPoint", "ACC\_value" and "chnm")

## Examples

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
excel_file_path <- file.path(path_to_tox, file_name)
tox_list <- create_toxEval(excel_file_path)
```

---

endpoint_hits_DT	<i>Rank endpoints by category</i>
------------------	-----------------------------------

---

## Description

The endpoint\_hits\_DT (data.table (DT) option) and endpoint\_hits (data frame option) functions create tables with one row per endPoint, and one column per category("Biological", "Chemical", or "Chemical Class"). The values in the table are the number of sites where the EAR exceeded the user-defined EAR hit\_threshold in that endpoint/category combination. If the category "Chemical" is chosen, an "info" link is provided to the chemical information available in the "Comptox Dashboard" <https://comptox.epa.gov/dashboard/>.

**Usage**

```

endpoint_hits_DT(
  chemical_summary,
  category = "Biological",
  mean_logic = FALSE,
  sum_logic = TRUE,
  hit_threshold = 0.1,
  include_links = TRUE
)

endpoint_hits(
  chemical_summary,
  category = "Biological",
  mean_logic = FALSE,
  sum_logic = TRUE,
  hit_threshold = 0.1
)

```

**Arguments**

chemical_summary	Data frame from get_chemical_summary
category	Character. Either "Biological", "Chemical Class", or "Chemical".
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE indicates that EAR values are not considered to be additive and often will be a more appropriate choice for traditional benchmarks as opposed to ToxCast benchmarks.
hit_threshold	Numeric. EAR threshold defining a "hit".
include_links	Logical. whether or not to include a link to the ToxCast dashboard. Only needed for the "Chemical" category.

**Details**

The tables show slightly different results when choosing to explore data from a single site rather than all sites. The value displayed in this instance is the number of samples with hits rather than the number of sites with hits.

**Value**

data frame with one row per endpoint that had a hit (based on the hit\_threshold). The columns are based on the category.

**Examples**

```

# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")

```

```
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

hits_df <- endpoint_hits(chemical_summary, category = "Biological")
endpoint_hits_DT(chemical_summary, category = "Biological")
endpoint_hits_DT(chemical_summary, category = "Chemical Class")
endpoint_hits_DT(chemical_summary, category = "Chemical")
```

---

end\_point\_info

*Endpoint information from ToxCast*

---

## Description

See vignette("Setting up toxEval package data", package = "toxEval") for more information on how the data was aggregated.

## Value

data frame with 72 columns. The columns and definitions are discussed in the "ToxCast Assay Annotation Version 1.0 Data User Guide (PDF)" (see source). The column "Relevance Category" was included for consideration of grouping/filtering endpoints based on user goals.

## Source

[doi:10.23645/epacomptox.6062479.v3](https://doi.org/10.23645/epacomptox.6062479.v3)

## References

U.S. EPA. 2014. ToxCast Assay Annotation Data User Guide.

## Examples

```
end_point_info <- end_point_info
head(end_point_info[, 1:5])
```

---

explore_endpoints	<i>Explore data in the Shiny Application</i>
-------------------	--

---

### Description

Open an interactive app in a browser. Using this function is a quick and convenient way to explore data. For more customization, the R-code to produce each graph and table is displayed in the app. That is a good starting-point for a custom analysis.

### Usage

```
explore_endpoints(browse = TRUE)
```

### Arguments

browse	Logical. Use browser for running Shiny app.
--------	---

---

filter_groups	<i>Filter endPoints based on groups and assays.</i>
---------------	---

---

### Description

This function provides a mechanism to specify 3 levels of information in the supplied data frame `end_point_info` to be used in subsequent analysis steps. First, the user specifies the ToxCast assay annotation using the 'groupCol' argument, which is a column header in 'end\_point\_info'. Second, the user specifies the families of assays to exclude. Finally, the user can choose to remove specific group(s) from the category. The default is to remove 'Background Measurement' and 'Undefined'. Choices for this should be reconsidered based on individual study objectives.

### Usage

```
filter_groups(
  ep,
  groupCol = "intended_target_family",
  remove_assays = c("BSK"),
  remove_groups = c("Background Measurement", "Undefined")
)
```

### Arguments

ep	Data frame containing Endpoint information from ToxCast
groupCol	Character name of ToxCast annotation column to use as a group category
remove_assays	Vector of assays to EXCLUDE in the data analysis. By default, the "BSK" (BioSeek) assay is removed.
remove_groups	Vector of groups within the selected 'groupCol' to remove.



## Details

The default category ('groupCol') is 'intended\_target\_family'. Depending on the study, other categories may be more relevant. The best resource on these groupings is the "ToxCast Assay Annotation Data User Guide". It defines "intended\_target\_family" as "the target family of the objective target for the assay". Much more detail can be discovered in that documentation.

## Examples

```
end_point_info <- end_point_info
cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
head(filtered_ep)
```

---

flags

*ToxCast Chemical Information*

---

## Description

See vignette("Setting up toxEval package data", package = "toxEval") for more information on how the data was aggregated.

## Value

data frame

## Examples

```
head(flags)
```

---

get\_ACC

*Get the ACC values for a selection of chemicals*

---

## Description

The get\_ACC function retrieves the activity concentration at cutoff (ACC) values for specified chemicals.

## Usage

```
get_ACC(CAS)
```

## Arguments

CAS                      Vector of CAS.

**Details**

The function `get_ACC` will convert the ACC values in the ToxCast database from units of ( $\mu\text{M}$ ) to units of  $\mu\text{g/L}$ , and reformat the data as input to `toxEval`.

**Value**

data frame with columns CAS, chnm, flags, endPoint, ACC, MWt, and ACC\_value

**Examples**

```
CAS <- c("121-00-6", "136-85-6", "80-05-7", "84-65-1", "5436-43-1", "126-73-8")
ACC <- get_ACC(CAS)
head(ACC)
```

---

`get_chemical_summary`    *Compute EAR values*

---

**Description**

This function computes Exposure:Activity ratios using user-provided measured concentration data from the output of `create_toxEval`, and joins the data with the activity concentration at cutoff data provided by ToxCast. Data from ToxCast is included with this package, but alternative benchmark data can be provided to perform the same "toxEval" analysis.

**Usage**

```
get_chemical_summary(
  tox_list,
  ACC = NULL,
  filtered_ep = "All",
  chem_data = NULL,
  chem_site = NULL,
  chem_info = NULL,
  exclusion = NULL
)
```

**Arguments**

<code>tox_list</code>	List with data frames for <code>chem_data</code> , <code>chem_info</code> , <code>chem_site</code> , and optionally exclusions and benchmarks. Created with <code>create_toxEval</code> .
<code>ACC</code>	Data frame with columns: CAS, chnm, endPoint, and ACC_value for specific chemical/endpoint combinations generated using the <code>get_ACC</code> function. End-Points with specific data quality flags may optionally be removed using the <code>remove_flags</code> function.
<code>filtered_ep</code>	Data frame with columns: endPoints, groupCol. Default is "All", where no filtering occurs.

chem_data	<i>Optional</i> data frame with (at least) columns: CAS, SiteID, and Value. Default is NULL. The argument will over-ride what is in tox_list.
chem_site	<i>Optional</i> data frame with (at least) columns: SiteID, and Short Name. Default is NULL. The argument will over-ride what is in tox_list.
chem_info	<i>Optional</i> data frame with (at least) columns: CAS, and class. Default is NULL. The argument will over-ride what is in tox_list.
exclusion	<i>Optional</i> data frame with (at least) columns: CAS and endPoint. Default is NULL. The argument will over-ride what is in tox_list.

### Details

To use the data provided by the package, a sample workflow is shown below in the examples. The examples include retrieving the ToxCast (ACC) values that are used to calculate EARs, choosing endPoints that should be ignored based on data quality "flags" in the ToxCast database, and removing groups of endPoints that may not be important to the analysis at hand.

### Value

a data frame with the columns: CAS, chnm (chemical name as a factor), site, date, EAR, Bio\_category, shortName (of site), Class. The output of this function is where you find EAR values for every chemical/endpoint combination.

### Examples

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)

chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)
head(chemical_summary)
```

---

get\_concentration\_summary

*Create concentration summary*

---

### Description

Use this function to create a chemical\_summary, but instead of using any benchmarks, the EAR column is simply the concentration. The output of this function can be used in any of the plotting or table functions in the same way that the output of [get\\_chemical\\_summary](#).

**Usage**

```
get_concentration_summary(
  tox_list,
  chem_data = NULL,
  chem_site = NULL,
  chem_info = NULL,
  tox_names = FALSE
)
```

**Arguments**

tox_list	List with data frames for chem_data, chem_info, and chem_site. Created with <a href="#">create_toxEval</a> .
chem_data	<i>Optional</i> data frame with (at least) columns: CAS, SiteID, and Value. Default is NULL. The argument will over-ride what is in tox_list.
chem_site	<i>Optional</i> data frame with (at least) columns: SiteID, and Short Name. Default is NULL. The argument will over-ride what is in tox_list.
chem_info	<i>Optional</i> data frame with (at least) columns: CAS, and class. Default is NULL. The argument will over-ride what is in tox_list.
tox_names	Logical whether to use the provided chemical names from the ToxCast or not. If there is not a match by CAS, the function will look for a column "Chemical" in the "Chemical" tab. If that column doesn't exist, it will create a (not good!) name.

**Value**

a data frame with the columns: CAS, chnm (chemical name as a factor), site, date, EAR (which is just concentration), Bio\_category, shortName (of site), Class. The output of this function is where you find EAR values for every chemical/endpoint combination.

**Examples**

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

chemical_summary_conc <- get_concentration_summary(tox_list)
head(chemical_summary_conc)
plot_tox_boxplots(chemical_summary_conc,
  category = "Chemical",
  x_label = "Concentration [ug/L]"
)
```

---

graph_chem_data	<i>Prepare boxplot data</i>
-----------------	-----------------------------

---

## Description

A set of functions to prepare the data for boxplots. Often, these functions are used within the plotting functions. They are exported however to allow custom graphs to be created.

## Usage

```
graph_chem_data(
  chemical_summary,
  ...,
  manual_remove = NULL,
  mean_logic = FALSE,
  sum_logic = TRUE
)

tox_boxplot_data(
  chemical_summary,
  category = "Biological",
  manual_remove = NULL,
  mean_logic = FALSE,
  sum_logic = TRUE
)

side_by_side_data(
  gd_left,
  gd_right,
  left_title = "Left",
  right_title = "Right"
)
```

## Arguments

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
...	Additional group_by arguments. This can be handy for creating facet graphs.
manual_remove	Vector of categories to remove.
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
category	Character. Either "Biological", "Chemical Class", or "Chemical".
gd_left	Data frame that must include the columns chnm, Class, and either EAR or mean- EAR.

gd_right	Data frame that must include the columns chnm, Class, and either EAR or mean-EAR.
left_title	Character that will be associated with the "gd_left" data frame in a column named "guide_side".
right_title	Character that will be associated with the "gd_right" data frame in a column named "guide_side".

## Details

The function `side_by_side_data` will combine two data frames, either the output of `get_chemical_summary` or `graph_chem_data`, into a single data frame. The important work here is that the chemicals and classes factor levels are ordered primarily based on "gd\_left", but include "gd\_right" when the contents are mismatched.

## Examples

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)

chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)
# Let's say we want to compare 2 chemical summaries
# We'll look at one summing EARs, and with concentrations
# First, we need a chemical summary for concentrations:
chemical_summary_conc <- get_concentration_summary(tox_list)

gd_tox <- graph_chem_data(chemical_summary)
gd_conc <- graph_chem_data(chemical_summary_conc)

ch_combo <- side_by_side_data(gd_tox, gd_conc,
  left_title = "ToxCast",
  right_title = "Concentrations"
)
plot_chemical_boxplots(ch_combo, guide_side,
  x_label = ""
) +
  ggplot2::facet_grid(. ~ guide_side, scales = "free_x")
```

---

hits\_by\_groupings\_DT    *Biological hits per category*

---

## Description

The hits\_by\_groupings\_DT (DT option) and hits\_by\_groupings (data frame option) functions create tables with one row per category("Biological", "Chemical", or "Chemical Class"). The columns indicate the "Biological" groupings. The values in the table signify how many sites have samples with EARs that exceeded the hit\_threshold for that particular "Biological"/category combination. If the user chooses "Biological" as the category, it is a simple 2-column table of "Biological" groupings and number of sites (nSites).

## Usage

```
hits_by_groupings_DT(  
  chemical_summary,  
  category = "Biological",  
  mean_logic = FALSE,  
  sum_logic = TRUE,  
  hit_threshold = 0.1  
)
```

```
hits_by_groupings(  
  chemical_summary,  
  category,  
  mean_logic = FALSE,  
  sum_logic = TRUE,  
  hit_threshold = 0.1  
)
```

## Arguments

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
category	Character. Either "Biological", "Chemical Class", or "Chemical".
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
hit_threshold	Numeric threshold defining a "hit".

## Details

The tables result in slightly different results for a single site, displaying the number of samples with hits rather than the number of sites.

**Value**

data frame with one row per category, and one column per Biological grouping.

**Examples**

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

site_df <- hits_by_groupings(chemical_summary, category = "Biological")

hits_by_groupings_DT(chemical_summary, category = "Biological")
hits_by_groupings_DT(chemical_summary, category = "Chemical Class")
hits_by_groupings_DT(chemical_summary, category = "Chemical")
```

---

hits_summary_DT	<i>Summary of hits per site/category</i>
-----------------	--

---

**Description**

The hits\_summary\_DT (DT option) and hits\_summary (data frame option) functions create tables information on the number of hit\_threshold exceedances per site for each individual grouping. The table has one row per group per site that has hit\_threshold exceedances. For example, if "Biological" is the category, and a site has EAR levels above the specified hit\_threshold for "DNA Binding" and "Nuclear Receptors", that site will have 2 rows of data in this table.

**Usage**

```
hits_summary_DT(
  chemical_summary,
  category = "Biological",
  sum_logic = TRUE,
  hit_threshold = 0.1
)

hits_summary(chemical_summary, category, hit_threshold = 0.1, sum_logic = TRUE)
```



**Arguments**

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
category	Character. Either "Biological", "Chemical Class", or "Chemical".
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
hit_threshold	Numeric threshold defining a "hit".

**Details**

For each row, there are 4 columns. Site and category (as defined by the category argument) define the row. "Samples with hits" are how many samples exceeded the hit\_threshold for the specified category at the specified site. "Number of Samples" indicates how many samples were collected at an individual site based on unique date.

The tables contain slightly different results for evaluation of a single site. There are three columns (the Site column is dropped), and rather than one row per site/category, there is one row per category.

**Value**

data frame with with one row per unique site/category combination. The columns are site, category, Samples with Hits, and Number of Samples.

data frame with columns "Hits per Sample", "Individual Hits", "nSample", "site", and "category"

**Examples**

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

stats_group <- hits_summary(chemical_summary, "Biological")

hits_summary_DT(chemical_summary, category = "Biological")
hits_summary_DT(chemical_summary, category = "Chemical Class")
hits_summary_DT(chemical_summary, category = "Chemical")
```

---

make\_tox\_map*Create an interactive map of the data*

---

## Description

The function `make_tox_map` creates a [leaflet](#) map of the sites. This function places symbols at the location of each site in the data file that represent the magnitude of EAR (color) and the number of samples in the data set (size). This is the only function that requires "dec\_lon" and "dec\_lat" (decimal longitude and decimal latitude) in the data frame specified for the `chem_site` argument.

## Usage

```
make_tox_map(  
  chemical_summary,  
  chem_site,  
  category = "Biological",  
  mean_logic = FALSE,  
  sum_logic = TRUE  
)  
  
map_tox_data(  
  chemical_summary,  
  chem_site,  
  category = "Biological",  
  mean_logic = FALSE,  
  sum_logic = TRUE  
)
```

## Arguments

<code>chemical_summary</code>	Data frame from <a href="#">get_chemical_summary</a> .
<code>chem_site</code>	Data frame containing the columns <code>SiteID</code> , <code>site_grouping</code> , <code>Short Name</code> , <code>dec_lon</code> , and <code>dec_lat</code> .
<code>category</code>	Character. Either "Biological", "Chemical Class", or "Chemical".
<code>mean_logic</code>	Logical. TRUE displays the mean EAR from each site, FALSE displays the maximum EAR from each site.
<code>sum_logic</code>	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.

## Details

The function `map_tox_data` calculates the statistics for the map. It may be useful on its own.

## Examples

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)
tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

make_tox_map(chemical_summary, tox_list$chem_site, "Biological")
make_tox_map(chemical_summary, tox_list$chem_site, "Chemical Class")
make_tox_map(chemical_summary, tox_list$chem_site, "Chemical")
```

---

plot\_chemical\_boxplots

*Grouped Boxplots*

---

## Description

The `plot_tox_boxplots` function creates a set of boxplots representing EAR values computed with the `get_chemical_summary` function, and dependent on the choice of several input options. See "Summarizing the data" in the Introduction vignette: `vignette("Introduction", package = "toxEval")`. for a description of how the EAR values are computed, aggregated, and summarized. Choosing "Chemical Class" in the category argument will generate separate boxplots for each unique class. "Chemical" will generate boxplots for each individual chemical, and "Biological" will generate boxplots for each group in the selected ToxCast annotation.

## Usage

```
plot_chemical_boxplots(
  chemical_summary,
  ...,
  manual_remove = NULL,
  mean_logic = FALSE,
  sum_logic = TRUE,
  plot_ND = TRUE,
  font_size = NA,
  title = NA,
  x_label = NA,
  palette = NA,
  hit_threshold = NA,
```

```

    site_counts = "count"
  )

plot_tox_boxplots(
  chemical_summary,
  category = "Biological",
  manual_remove = NULL,
  mean_logic = FALSE,
  sum_logic = TRUE,
  plot_ND = TRUE,
  font_size = NA,
  title = NA,
  x_label = NA,
  palette = NA,
  hit_threshold = NA
)

```

### Arguments

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
...	Additional group_by arguments. This can be handy for creating facet graphs.
manual_remove	Vector of categories to remove.
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
plot_ND	Logical. Whether or not to plot "Biological" groupings, "Chemical Class" groupings, or "Chemical" that do not have any detections.
font_size	Numeric value to adjust the axis font size.
title	Character title for plot. Default is NA which produces no title.
x_label	Character for x label. Default is NA which produces an automatic label.
palette	Vector of color palette for boxplot fill. Can be a named vector to specify specific colors for specific categories.
hit_threshold	Numeric threshold defining a "hit".
site_counts	This describes how to calculate the left-side counts. Default is "count" which is number of sites with detections. "frequency" would calculate a percentage of detections. This assumes all non-detects have been included as 0 for a value. "none" is the final option which can be to remove those labels. Custom labels could then be constructed after this function is run.
category	Character. Either "Biological", "Chemical Class", or "Chemical".

### Details

It is also possible to display a threshold line using the `hit_threshold` argument. The graph will then include the number of sites with detections, the threshold line, and the number of "hits" indicating how many sites that have EAR values exceeding the `hit_threshold`.

The graph shows a slightly different result for a single site. For a single site graph, the number of chemicals that were detected and have associated endpoint ACCs represented are displayed.

The functions `plot_tox_boxplots` and `graph_chem_data` are functions that perform the statistical calculations to create the plot. `graph_chem_data` is specific to the "Chemical" plot, and `plot_tox_boxplots` is for "Biological" and "Chemical Class".

Box plots are standard Tukey representations. See "Box plot details" in the Basic Workflow vignette: `vignette("basicWorkflow", package = "toxEval")` for more information.

## Examples

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)
ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)
plot_tox_boxplots(chemical_summary, "Biological")

plot_tox_boxplots(chemical_summary, "Chemical Class")
plot_tox_boxplots(chemical_summary, "Chemical")

cbPalette <- c(
  "#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
  "#0072B2", "#D55E00", "#CC79A7"
)
graphData <- tox_boxplot_data(
  chemical_summary = chemical_summary,
  category = "Biological"
)
cbValues <- colorRampPalette(cbPalette)(length(levels(graphData$category)))
names(cbValues) <- levels(graphData$category)

plot_tox_boxplots(chemical_summary,
  hit_threshold = 0.1,
  category = "Biological",
  palette = cbValues,
  title = "Maximum EAR per site, grouped by biological activity groupings"
)

plot_tox_boxplots(chemical_summary,
  category = "Chemical", x_label = "EAR"
)
single_site <- dplyr::filter(chemical_summary, site == "USGS-04024000")
```

```

plot_tox_boxplots(single_site,
  category = "Biological"
)
plot_tox_boxplots(single_site,
  category = "Chemical", hit_threshold = 0.001
)

```

---

plot_tox_endpoints	<i>EndPoint boxplots</i>
--------------------	--------------------------

---

## Description

The `plot_tox_endpoints` function creates a set of boxplots representing EAR values for each `endPoint` based on the selected data. A subset of data is first chosen by specifying a group in the `filterBy` argument. The `filterBy` argument must match one of the unique options in the category. For example, if the category is "Chemical Class", then the `filterBy` argument must be one of the defined "Chemical Class" options such as "Herbicide". A boxplot is generated for each `endPoint`. The EAR values that are used to create the boxplots are the mean or maximum (as defined by `mean_logic`) for each site as described in "Summarizing the data" in the Introduction vignette: `vignette("Introduction", package = "toxEval")`.

## Usage

```

plot_tox_endpoints(
  chemical_summary,
  category = "Biological",
  filterBy = "All",
  manual_remove = NULL,
  hit_threshold = NA,
  mean_logic = FALSE,
  sum_logic = TRUE,
  font_size = NA,
  title = NA,
  x_label = NA,
  palette = NA,
  top_num = NA
)

```

## Arguments

<code>chemical_summary</code>	Data frame from <code>get_chemical_summary</code> .
<code>category</code>	Either "Biological", "Chemical Class", or "Chemical".
<code>filterBy</code>	Character. Either "All" or one of the filtered categories.
<code>manual_remove</code>	Vector of categories to remove.
<code>hit_threshold</code>	Numeric threshold defining a "hit".

mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
font_size	Numeric to adjust the axis font size.
title	Character title for plot.
x_label	Character for x label. Default is NA which produces an automatic label.
palette	Vector of color palette for fill. Can be a named vector to specify specific color for specific categories.
top_num	Integer number of endpoints to include in the graph. If NA, all endpoints will be included.

### Details

Box plots are standard Tukey representations. See "Box plot details" in the Basic Workflow vignette: `vignette("basicWorkflow", package = "toxEval")` for more information.

### Examples

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)
ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

plot_tox_endpoints(chemical_summary,
  filterBy = "Cell Cycle",
  top_num = 10
)

plot_tox_endpoints(chemical_summary,
  filterBy = "Cell Cycle",
  top_num = 10,
  x_label = "EAR"
)

plot_tox_endpoints(chemical_summary,
  category = "Chemical Class", filterBy = "PAHs",
  top_num = 10, hit_threshold = 0.001
)

plot_tox_endpoints(chemical_summary, category = "Chemical", filterBy = "Atrazine")
```

```
plot_tox_endpoints(chemical_summary, category = "Chemical", top_num = 10)
single_site <- dplyr::filter(chemical_summary, site == "USGS-04024000")
plot_tox_endpoints(single_site, category = "Chemical", top_num = 10)
```

---

plot\_tox\_endpoints2     *EndPoint boxplots with faceting option*

---

## Description

The `plot_tox_endpoints2` function creates a set of boxplots representing EAR values for each `EndPoint` based on the selected data. A subset of data is first chosen by specifying a group in the `filterBy` argument. The `filterBy` argument must match one of the unique options in the category. For example, if the category is "Chemical Class", then the `filterBy` argument must be one of the defined "Chemical Class" options such as "Herbicide".

## Usage

```
plot_tox_endpoints2(
  cs,
  ...,
  category = "Chemical",
  filterBy = "All",
  manual_remove = NULL,
  hit_threshold = NA,
  mean_logic = FALSE,
  sum_logic = TRUE,
  font_size = NA,
  title = NA,
  x_label = NA,
  palette = NA,
  top_num = NA
)
```

## Arguments

<code>cs</code>	Data.frame from <a href="#">get_chemical_summary</a> .
<code>...</code>	Additional <code>group_by</code> arguments. This can be handy for creating facet graphs.
<code>category</code>	Either "Biological", "Chemical Class", or "Chemical".
<code>filterBy</code>	Character. Either "All" or one of the filtered categories.
<code>manual_remove</code>	Vector of categories to remove.
<code>hit_threshold</code>	Numeric threshold defining a "hit".
<code>mean_logic</code>	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
<code>sum_logic</code>	logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.



font_size	Numeric to adjust the axis font size.
title	Character title for plot.
x_label	Character for x label. Default is NA which produces an automatic label.
palette	Vector of color palette for fill. Can be a named vector to specify specific color for specific categories.
top_num	Integer number of endpoints to include in the graph. If NA, all endpoints will be included.

### Details

The difference between this function and the [plot\\_tox\\_endpoints](#) is that the ... arguments allow for customized faceting. To include this in the original toxEval function, backward compatibility would be broken.

### Examples

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)
ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
cs <- get_chemical_summary(tox_list, ACC, filtered_ep)
cs$guide_side <- "A"

cs2 <- cs
cs2$guide_side <- "B"

cs_double <- rbind(cs, cs2)

plot_tox_endpoints2(cs_double, guide_side,
  top_num = 10
) +
  ggplot2::facet_grid(. ~ guide_side, scales = "free_x")
```

## Description

The `plot_tox_heatmap` function creates a heat (tile) map with sites on the x-axis, a specified grouping on the y-axis (defined by the `category` argument), and color shading defining the mean or maximum EAR. See "Summarizing the data" in the Introduction vignette: `vignette("Introduction", package = "toxEval")` for a description on how the EAR values are computed, aggregated, and summarized. The y-axis grouping can be "Biological", "Chemical Class", or "Chemical". When specifying the "Chemical" option, a secondary y-axis is automatically included to group chemicals into chemical class. The function computes default breaks for the color scale to match the spread of the data, but breaks can also be customized with the `breaks` argument. This is a function where it may be ideal to create a custom order to the sites (for example, west-to-east). See the above section "Custom configuration" `vignette("Introduction", package = "toxEval")` for instructions on how to convert the character vector sites to a factor with ordered levels.

## Usage

```
plot_tox_heatmap(
  chemical_summary,
  chem_site,
  category = "Biological",
  breaks = c(1e-05, 1e-04, 0.001, 0.01, 0.1, 1, 10),
  manual_remove = NULL,
  mean_logic = FALSE,
  sum_logic = TRUE,
  plot_ND = TRUE,
  font_size = NA,
  title = NA,
  legend_lab = NA
)
```

## Arguments

<code>chemical_summary</code>	Data frame from <code>get_chemical_summary</code> .
<code>chem_site</code>	Data frame with columns SiteID, site_grouping, and Short Name.
<code>category</code>	Either "Biological", "Chemical Class", or "Chemical".
<code>breaks</code>	Numerical vector to define data bins and legend breaks.
<code>manual_remove</code>	Vector of categories to remove.
<code>mean_logic</code>	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
<code>sum_logic</code>	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
<code>plot_ND</code>	Logical. Whether or not to plot "Biological" groupings, "Chemical Class" groupings, or "Chemical" that do not have any detections.
<code>font_size</code>	Numeric value to adjust the axis font size.
<code>title</code>	Character title for plot.
<code>legend_lab</code>	Character label for legend. Default is NA which produces an automatic label.

## Details

If there are site/parameters (chemical/chemical class/biological grouping) combinations that don't have data, those areas are represented by an "X". If there are 0 values, they are considered "non-detects", and represented with a distinct color.

## Examples

```
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)

chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

# Order the site_groupings:
tox_list$chem_site$site_grouping <- factor(tox_list$chem_site$site_grouping,
  levels = c(
    "Lake Superior",
    "Lake Michigan",
    "Lake Huron",
    "Lake Erie",
    "Lake Ontario"
  )
)

# Order sites:
sitesOrdered <- c(
  "StLouis", "Nemadji", "WhiteWI", "Bad", "Montreal",
  "PresqueIsle", "Ontonagon", "Sturgeon", "Tahquamenon", "Burns",
  "IndianaHC", "StJoseph", "PawPaw", "Kalamazoo", "GrandMI",
  "Milwaukee", "Muskegon", "WhiteMI", "PereMarquette", "Manitowoc",
  "Manistee", "Fox", "Oconto", "Peshtigo", "Menominee",
  "Indian", "Cheboygan", "Ford", "Escanaba", "Manistique",
  "ThunderBay", "AuSable", "Rifle", "Saginaw", "BlackMI",
  "Clinton", "Rouge", "HuronMI", "Raisin", "Maumee",
  "Portage", "Sandusky", "HuronOH", "Vermilion", "BlackOH",
  "Rocky", "Cuyahoga", "GrandOH", "Cattaraugus", "Tonawanda",
  "Genesee", "Oswego", "BlackNY", "Oswegatchie", "Grass",
  "Raquette", "StRegis"
)

tox_list$chem_site$`Short Name` <- factor(tox_list$chem_site$`Short Name`,
  levels = sitesOrdered
)
```

```

plot_tox_heatmap(chemical_summary,
                  tox_list$chem_site,
                  category = "Chemical Class")

plot_tox_heatmap(chemical_summary,
                  tox_list$chem_site,
                  category = "Chemical",
                  legend_lab = "EAR"
)

single_site <- dplyr::filter(chemical_summary, site == "USGS-04024000")

plot_tox_heatmap(
  chemical_summary = single_site,
  chem_site = dplyr::filter(tox_list$chem_site,
                           SiteID == "USGS-04024000"),
  category = "Chemical Class"
)
plot_tox_heatmap(
  chemical_summary = single_site,
  chem_site = dplyr::filter(tox_list$chem_site, SiteID == "USGS-04024000"),
  category = "Chemical"
)

```

---

plot\_tox\_stacks

*Plot stacked bar charts*


---

## Description

The `plot_tox_stacks` function creates a set of boxplots representing EAR values computed with the `get_chemical_summary` function, and dependent on the choice of several input options. See "Summarizing the data" in the Introduction vignette: `vignette("Introduction", package = "toxEval")` for a description on how the EAR values are computed, aggregated, and summarized. Choosing "Chemical Class" in the category argument will generate separate stacked bars for each unique class. "Chemical" will generate stacked bars for each individual chemical, and "Biological" will generate stacked bars for each group in the selected ToxCast annotation. The legend can optionally be turned on or off using the `include_legend` argument. It may be impractical for instance to show the legend for "Chemical" if there are hundreds of chemicals.

## Usage

```

plot_tox_stacks(
  chemical_summary,
  chem_site,
  category = "Biological",
  mean_logic = FALSE,
  sum_logic = TRUE,
  manual_remove = NULL,

```

```

    include_legend = TRUE,
    font_size = NA,
    title = NA,
    y_label = NA,
    top_num = NA
  )

```

## Arguments

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
chem_site	Data frame with at least columns SiteID, site_grouping, and Short Name.
category	Character. Either "Biological", "Chemical Class", or "Chemical".
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
manual_remove	Vector of categories to remove.
include_legend	Logical. Used to include legend or not.
font_size	Numeric value to adjust the axis font size.
title	Character title for plot.
y_label	Character for x label. Default is NA which produces an automatic label.
top_num	Integer number to include in the graph. If NA, all data will be included.

## Details

The graph displays a slightly different result for a single site. Providing data with only one site displays each individual sample as a stacked bar rather than the mean or maximum for a site.

This is a function where it may be ideal to create a custom order to the sites (for example, west-to-east). See the above section "Custom configuration" [vignette\("Introduction", package = "toxEval"\)](#) for instructions on how to convert the character vector sites to a factor with ordered levels.

## Examples

```

# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)

```

```

filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

plot_tox_stacks(chemical_summary, tox_list$chem_site,
                "Biological", top_num = 5)

plot_tox_stacks(chemical_summary, tox_list$chem_site, "Chemical Class")
plot_tox_stacks(chemical_summary, tox_list$chem_site, "Chemical", include_legend = FALSE)
plot_tox_stacks(chemical_summary, tox_list$chem_site, "Chemical", top_num = 5, y_label = "EAR")

single_site <- dplyr::filter(chemical_summary, site == "USGS-04024000")
plot_tox_stacks(single_site, tox_list$chem_site, "Chemical", top_num = 5)
plot_tox_stacks(single_site,
                chem_site = tox_list$chem_site,
                category = "Chemical", top_num = 5, y_label = "EAR"
)

```

---

rank\_sites\_DT

*Rank sites by EAR*


---

## Description

The rank\_sites\_DT (DT option) and rank\_sites (data frame option) functions create tables with one row per site. Columns represent the maximum or mean EAR (depending on the mean\_logic argument) for each category ("Chemical Class", "Chemical", or "Biological") and the frequency of the maximum or mean EAR exceeding a user specified hit\_threshold.

## Usage

```

rank_sites_DT(
  chemical_summary,
  category = "Biological",
  mean_logic = FALSE,
  sum_logic = TRUE,
  hit_threshold = 0.1
)

rank_sites(
  chemical_summary,
  category,
  hit_threshold = 0.1,
  mean_logic = FALSE,
  sum_logic = TRUE
)

```

**Arguments**

chemical_summary	Data frame from <a href="#">get_chemical_summary</a> .
category	Character. Either "Biological", "Chemical Class", or "Chemical".
mean_logic	Logical. TRUE displays the mean sample from each site, FALSE displays the maximum sample from each site.
sum_logic	Logical. TRUE sums the EARs in a specified grouping, FALSE does not. FALSE may be better for traditional benchmarks as opposed to ToxCast benchmarks.
hit_threshold	Numeric threshold defining a "hit".

**Details**

The tables show slightly different results for a single site. Rather than multiple columns for categories, there is now 1 row per category (since the site is known).

**Value**

data frame with one row per site, and the max or mean EAR and frequency of hits based on the category.

**Examples**

```
# This is the example workflow:
path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"

full_path <- file.path(path_to_tox, file_name)

tox_list <- create_toxEval(full_path)

ACC <- get_ACC(tox_list$chem_info$CAS)
ACC <- remove_flags(ACC)

cleaned_ep <- clean_endPoint_info(end_point_info)
filtered_ep <- filter_groups(cleaned_ep)
chemical_summary <- get_chemical_summary(tox_list, ACC, filtered_ep)

stats_df <- rank_sites(chemical_summary, "Biological")

rank_sites_DT(chemical_summary, category = "Biological")
rank_sites_DT(chemical_summary, category = "Chemical Class")
rank_sites_DT(chemical_summary, category = "Chemical")
```

remove\_flags

*Remove endpoints with specific data quality flags from data***Description**

Through the ToxCast program quality assurance procedures, information is examined and at times, it is necessary to assign a data quality flag to a specific chemical/assay result. A toxEval user may want to include or exclude assay results with certain flags depending on the objectives of a given study. Assay results with specific data quality flags assigned to them can be removed based on their designated flag with the `remove_flags` function. The flags included in ToxCast, and the associated flagsShort value (used in the `remove_flags` function) are as follows:

<b>flag_id</b>	<b>Full Name</b>
5*	Model directionality questionable
6*	Only highest conc above baseline, active
7	Only one conc above baseline, active
8	Multiple points above baseline, inactive
9	Bmd > ac50, indication of high baseline variability
10	Noisy data
11*	Borderline
15*	Gain AC50 < lowest conc & loss AC50 < mean conc
17	Less than 50% efficacy
18*	AC50 less than lowest concentration tested
13	Average number of replicates per conc is less than 2
14	Number of concentrations tested is less than 4
19	Cell viability assay fit with gnls winning model

Asterisks indicate flags removed in the function as default.

**Usage**

```
remove_flags(ACC, flag_id = c(5, 6, 11, 15, 18))
```

**Arguments**

ACC                      data frame with columns: casn, chnm, endPoint, and ACC\_value  
 flag\_id                  vector of flags to to trigger REMOVAL

**Examples**

```
CAS <- c("121-00-6", "136-85-6", "80-05-7", "84-65-1", "5436-43-1", "126-73-8")
ACC <- get_ACC(CAS)
nrow(ACC)

# See available flags and associated ids:
```



```

flags

ACC <- remove_flags(ACC)
nrow(ACC)

```

---

summary.toxEval	<i>Summary of tox_list</i>
-----------------	----------------------------

---

## Description

A "tox\_list" object is created from create\_toxEval. It is a list of 5 data frames: chem\_data, chem\_info, chem\_site, exclusions, and benchmarks. This function returns a message with how many chemicals have ToxCast information, and returns a vector of which chemicals do not have ToxCast information.

## Usage

```

## S3 method for class 'toxEval'
summary(object, ...)

```

## Arguments

object	toxEval object with "chem_info" data frame included.
...	additional parameters

## Examples

```

path_to_tox <- system.file("extdata", package = "toxEval")
file_name <- "OWC_data_fromSup.xlsx"
excel_file_path <- file.path(path_to_tox, file_name)
tox_list <- create_toxEval(excel_file_path)
summary(tox_list)

```

---

ToxCast_ACC	<i>ACC values included with toxEval. See vignette("Setting up toxEval package data", package = "toxEval") for more information on how the data was aggregated.</i>
-------------	--

---

## Description

Downloaded on September 2023 from ToxCast. See also <https://www.frontiersin.org/articles/10.3389/ftox.2023.1275980/full>.

## Value

data frame with columns CAS, chnm (chemical name), flags, endPoint, and ACC (value).

**Source**

<https://www.epa.gov/comptox-tools/exploring-toxcast-data>

**References**

U.S. EPA. 2023. ToxCast & Tox21 Summary Files. Retrieved from <https://www.epa.gov/comptox-tools/exploring-toxcast-data> on September 2023.

**Examples**

```
head(ToxCast_ACC)
```

---

tox_chemicals	<i>ToxCast Chemical Information</i>
---------------	-------------------------------------

---

**Description**

See vignette("Setting up toxEval package data", package = "toxEval") for more information on how the data was aggregated.

**Value**

data frame

**Examples**

```
head(tox_chemicals)
```

# Index

- \* **datasets**
  - end\_point\_info, [7](#)
  - flags, [9](#)
  - tox\_chemicals, [34](#)
  - ToxCast\_ACC, [33](#)
- as.toxEval, [3](#)
- clean\_endPoint\_info, [3](#)
- create\_toxEval, [4](#), [10](#), [12](#)
- end\_point\_info, [7](#), [8](#)
- endpoint\_hits(endpoint\_hits\_DT), [5](#)
- endpoint\_hits\_DT, [5](#)
- explore\_endpoints, [8](#)
- filter\_groups, [8](#)
- flags, [9](#)
- get\_ACC, [9](#), [10](#)
- get\_chemical\_summary, [10](#), [11](#), [13–15](#),  
[17–20](#), [22](#), [24](#), [26](#), [28](#), [29](#), [31](#)
- get\_concentration\_summary, [11](#)
- graph\_chem\_data, [13](#), [14](#)
- hits\_by\_groupings
  - (hits\_by\_groupings\_DT), [15](#)
- hits\_by\_groupings\_DT, [15](#)
- hits\_summary(hits\_summary\_DT), [16](#)
- hits\_summary\_DT, [16](#)
- leaflet, [18](#)
- make\_tox\_map, [18](#)
- map\_tox\_data(make\_tox\_map), [18](#)
- plot\_chemical\_boxplots, [19](#)
- plot\_tox\_boxplots
  - (plot\_chemical\_boxplots), [19](#)
- plot\_tox\_endpoints, [22](#), [25](#)
- plot\_tox\_endpoints2, [24](#)
- plot\_tox\_heatmap, [25](#)
- plot\_tox\_stacks, [28](#)
- rank\_sites(rank\_sites\_DT), [30](#)
- rank\_sites\_DT, [30](#)
- remove\_flags, [10](#), [32](#)
- side\_by\_side\_data(graph\_chem\_data), [13](#)
- summary.toxEval, [33](#)
- tox\_boxplot\_data(graph\_chem\_data), [13](#)
- tox\_chemicals, [34](#)
- ToxCast\_ACC, [33](#)