

# Package ‘toxpiR’

December 16, 2025

**Type** Package

**Title** Create ToxPi Prioritization Models

**Version** 1.3.1

**Description** Enables users to build 'ToxPi' prioritization models and provides functionality within the grid framework for plotting ToxPi graphs. 'toxpiR' allows for more customization than the 'ToxPi GUI' ([<https://toxpi.github.io/>](https://toxpi.github.io/)) and integration into existing workflows for greater ease-of-use, reproducibility, and transparency. toxpiR package behaves nearly identically to the GUI; the package documentation includes notes about all differences. The vignettes download example files from [<https://github.com/ToxPi/ToxPi-example-files>](https://github.com/ToxPi/ToxPi-example-files).

**Imports** grDevices, methods, S4Vectors, grid, rlang, stats, BiocGenerics, tidyr, utils, ggplot2

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**Depends** R (>= 4.0)

**Suggests** rmarkdown, knitr, testthat (>= 3.0.0), covr, DBI

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**VignetteBuilder** knitr

**License** GPL (>= 3)

**URL** <https://github.com/ToxPi/toxpiR>, <https://toxpi.github.io/toxpiR/>

**BugReports** <https://github.com/ToxPi/toxpiR/issues>

**NeedsCompilation** no

**Author** Jonathon F Fleming [aut, cre] (ORCID: [<https://orcid.org/0000-0003-2447-3139>](https://orcid.org/0000-0003-2447-3139)),  
Dayne L Filer [aut, fnd] (ORCID: [<https://orcid.org/0000-0002-3443-5315>](https://orcid.org/0000-0002-3443-5315)),

Dillon T Lloyd [aut],  
 Preethi Thunga [aut] (ORCID: <<https://orcid.org/0000-0001-5447-0129>>),  
 Skylar W Marvel [aut],  
 Alison A Motsinger-Reif [fnd] (ORCID:  
 <<https://orcid.org/0000-0003-1346-2493>>),  
 David M Reif [aut, fnd] (ORCID:  
 <<https://orcid.org/0000-0001-7815-6767>>)

**Maintainer** Jonathon F Fleming <jffleming0129@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-12-16 15:20:02 UTC

## Contents

boxLegendGrob . . . . .	2
pieGridGrob . . . . .	3
pieGrob . . . . .	5
toxiR-datasets . . . . .	6
txpCalculateScores . . . . .	7
txpExportGui . . . . .	8
txpGenerics . . . . .	9
txpImportGui . . . . .	10
TxpModel-class . . . . .	11
TxpModelList-class . . . . .	13
TxpResult-class . . . . .	15
TxpResult-plot . . . . .	19
TxpResultList-class . . . . .	21
TxpResultParam-class . . . . .	22
TxpSlice-class . . . . .	23
TxpSliceList-class . . . . .	25
TxpTransFunc-class . . . . .	27
TxpTransFuncList-class . . . . .	28
<b>Index</b>	<b>30</b>

---

boxLegendGrob	<i>Create a filled-box legend</i>
---------------	-----------------------------------

---

### Description

Create a filled-box legend

### Usage

```
boxLegendGrob(labels, fills, name = NULL, vp = NULL, gp = NULL)
```

**Arguments**

labels	Character, the legend labels
fills	Colors to fill the slices
name, vp, gp	Passed to <a href="#">grid::frameGrob</a>

**Details**

Not yet exported. Need to break out the creation of viewports and grobs as done in the exported grobs. This will allow better grobEdit methods, which also needs to be created for the boxLegendGrob. Also need to do some input checks.

Also, if `grid::legendGrob` gets updated to use the 'has.fill' option this function should be removed and `grid::legendGrob` can be used instead.

---

pieGridGrob	<i>Make grid of pieGrobs</i>
-------------	------------------------------

---

**Description**

Make grid of pieGrobs

**Usage**

```
pieGridGrob(
  radMat,
  wts = NULL,
  fills = NULL,
  labels = NULL,
  showRadSum = FALSE,
  ncol = NULL,
  nrow = NULL,
  byrow = TRUE,
  name = NULL,
  gp = NULL,
  vp = NULL
)
```

```
grid.pieGridGrob(
  radMat,
  wts = NULL,
  fills = NULL,
  labels = NULL,
  showRadSum = FALSE,
  ncol = NULL,
  nrow = NULL,
  byrow = TRUE,
  name = NULL,
```

```

    gp = NULL,
    vp = NULL
  )

```

### Arguments

radMat	matrix(<numeric>), observations by slice radii
wts	vector(<numeric>), relative weights of each slice
fills	Vector of colors to fill slices
labels	vector(<character>), (optional) label for each observation
showRadSum	Logical scalar, when TRUE show the weighted sum of slices below the label
nrow, ncol	Integer scalar, number of rows and columns for the grid
byrow	Logical scalar, fill the grid by rows when TRUE
name, gp, vp	Passed to <a href="#">grid::gTree</a>

### Value

pieGrob [grid::grob](#) object

### Examples

```

library(grid)

s <- seq(0.2, 1, by = 0.1)
smat <- do.call("rbind", replicate(20, s, simplify = FALSE))
grid.newpage()
grid.pieGridGrob(radMat = smat)

rownames(smat) <- sprintf("obs%02d", 1:20)
grid.newpage()
grid.pieGridGrob(radMat = smat, wts = s)
grid.newpage()
grid.pieGridGrob(radMat = smat, wts = s, showRadSum = TRUE, labels = FALSE)
grid.newpage()
grid.pieGridGrob(radMat = smat, labels = "hello")
grid.newpage()
grid.pieGridGrob(radMat = smat, labels = 1:20)

## Can edit like normal grid objects
grid.newpage()
grid.pieGridGrob(radMat = smat, wts = s, showRadSum = TRUE)
grid.ls() ## shows grid elements
grid.edit("pie-20", fills = 1:9)
grid.edit("pie-19-label", gp = gpar(font = 2, col = "red"))
grid.edit("pie-1", wts = rep(1, 9), rads = rep(1, 9))
for (s in sprintf("pie-%d-radSum", 2:4)) {
  grid.edit(s, gp = gpar(font = 2, col = "blue"))
}

```

---

pieGrob	<i>Create a pie grob</i>
---------	--------------------------

---

## Description

Create a pie grob

## Usage

```
pieGrob(rads, fills = NULL, wts = NULL, name = NULL, vp = NULL, gp = NULL)
```

```
grid.pieGrob(rads, fills = NULL, wts = NULL, name = NULL, vp = NULL, gp = NULL)
```

## Arguments

rads	Numeric, radius values for each slice from 0 to 1
fills	Colors to fill the slices
wts	Numeric, the relative portion of each slice
name, vp, gp	Passed to <a href="#">grid::gTree</a>

## Details

The default coloring can be set with `options("txp.fills")`.

## Value

pieGrob [grid::grob](#) object

## Examples

```
library(grid)

s <- seq(0.2, 1, by = 0.1)
grid.newpage()
grid.pieGrob(rads = s)
grid.newpage()
grid.pieGrob(rads = s, wts = s)

curr_txp_fills <- options()$txp.fills
options(txp.fills = 1:8)
grid.newpage()
grid.pieGrob(rads = s)
options(txp.fills = curr_txp_fills)

## Can edit
grid.newpage()
grid.pieGrob(rads = s, name = "myPie")
grid.ls() ## show the grid elements
```

```
grid.edit("myPie", fills = 1:9, wts = 9:1)
```

---

toxiR-datasets	<i>toxiR data objects</i>
----------------	---------------------------

---

## Description

Objects included in the toxiR package, loaded with `utils::data`

## Usage

```
data(txp_example_input, package = "toxiR")
```

```
data(txp_example_model, package = "toxiR")
```

## txp\_example\_input

Small example input data to be used with `txpCalculateScores` in creating `TxpResult` objects. A `base::data.frame` with 10 rows and 9 variables

**name** Observation names

**metric#** Input data for ToxPi models

## txp\_example\_model

Example `TxpModel` object intended for `txp_example_data`; model with 4 slices.

## Source

<https://github.com/ToxPi/ToxPi-example-files>

## Examples

```
data(txp_example_input, package = "toxiR")
data(txp_example_model, package = "toxiR")
txp_example_input
txp_example_model

## Code to create txp_example_model
tf1 <- TxpTransFuncList(linear = function(x) x)
s1 <- TxpSliceList(s1 = TxpSlice(sprintf("metric%d", 1:2)),
                  s2 = TxpSlice("metric3"),
                  s3 = TxpSlice(sprintf("metric%d", 4:7),
                                tf1[rep("linear", 4)]),
                  s4 = TxpSlice("metric8", tf1))
tf2 <- TxpTransFuncList(NULL, linear = function(x) x, NULL, NULL)
TxpModel(txpSlices = s1, txpWeights = c(2, 1, 3, 2), txpTransFuncs = tf2)
```

---

txpCalculateScores      *Calculate ToxPi Scores for the given model and input data*

---

## Description

Calculate ToxPi Scores for the given model and input data

## Usage

```
txpCalculateScores(model, input, ...)  
  
## S4 method for signature 'TxpModel,data.frame'  
txpCalculateScores(  
  model,  
  input,  
  id.var = NULL,  
  rank.ties.method = c("average", "first", "last", "random", "max", "min"),  
  negative.value.handling = c("keep", "missing")  
)  
  
## S4 method for signature 'TxpModelList,data.frame'  
txpCalculateScores(  
  model,  
  input,  
  id.var = NULL,  
  rank.ties.method = c("average", "first", "last", "random", "max", "min"),  
  negative.value.handling = c("keep", "missing")  
)  
  
## S4 method for signature 'list,data.frame'  
txpCalculateScores(  
  model,  
  input,  
  id.var = NULL,  
  rank.ties.method = c("average", "first", "last", "random", "max", "min"),  
  negative.value.handling = c("keep", "missing")  
)
```

## Arguments

model	<a href="#">TxpModel</a> object or <a href="#">TxpModelList</a> object
input	data.frame object containing the model input data
...	Included for extendability; not currently used
id.var	Character scalar, column in 'input' to store in
rank.ties.method	Passed to rank.ties.method slot

negative.value.handling

Passed to negative.value.handling slot

### Details

txpCalculateScores is implemented as an S4 generic function with methods for [TxpModel](#) and [TxpModelList](#).

Ranks are calculated such that the highest ToxPi score has a rank of 1.

Missingness is determined after applying input-level transformations but before applying slice-level transformations.

### Value

[TxpResult](#) or [TxpResultList](#) object

### See Also

[TxpModel](#), [TxpResult](#), [TxpResultParam](#)

### Examples

```
## Load example dataset & model; see ?TxpModel for building model objects
data(txp_example_input, package = "toxpiR")
data(txp_example_model, package = "toxpiR")

## Calculate scores for single model; returns TxpResult object
res <- txpCalculateScores(model = txp_example_model,
                          input = txp_example_input,
                          id.var = "name")

## Calculate scores for list of models; returns TxpResultList object
txpCalculateScores(model = TxpModelList(m1 = txp_example_model,
                                       m2 = txp_example_model),
                  input = txp_example_input,
                  id.var = "name")
resLst <- txpCalculateScores(model = list(m1 = txp_example_model,
                                         m2 = txp_example_model),
                            input = txp_example_input,
                            id.var = "name")
```

---

txpExportGui

*Export comma-separated file intended for ToxPi GUI*

---

### Description

Export comma-separated file intended for ToxPi GUI



**Usage**

```
txpExportGui(
  fileName = "txpModel.csv",
  input,
  model,
  id.var = NULL,
  fills = NULL
)
```

**Arguments**

fileName	Character scalar, the path to the output file
input	data.frame object containing the model input data
model	<a href="#">TxpModel</a> object or <a href="#">TxpModelList</a> object
id.var	Character scalar, column in 'input' to store in
fills	Colors to fill the slices

**Details**

The GUI differs in two meaningful ways for exporting `toxpIR` models: (1) the GUI only allows for integer weights; (2) the GUI applies transformation functions differently.

`txpExportGui` will not work for models with non-integer weights.

The GUI only applies a single transformation function to every input within a slice, and only functions from a pre-determined list; `toxpIR` allows users to apply any valid function individually to each input, then a second transformation function on the summed slice values. Because of this complexity, any exported models with slice-level transformation functions will not export at the input level. In other words, the export will have only the final slice scores. Otherwise, all input-level transformations will be performed, and the export will contain transformed input-level data with the `linear(x)` GUI transformation.

---

txpGenerics

*toxpIR package generics*


---

**Description**

`toxpIR` package generics; see class man pages for associated methods

**Usage**

```
txpValueNames(x, ...)
txpValueNames(x, ...) <- value
txpTransFuncs(x, ...)
```

```
txpTransFuncs(x, ...) <- value
txpSlices(x, ...)
txpSlices(x, ...) <- value
txpWeights(x, ...)
txpWeights(x, ...) <- value
txpScores(x, ...)
txpSliceScores(x, ...)
txpModel(x, ...)
txpIDs(x, ...)
txpIDs(x, ...) <- value
txpRanks(x, ...)
txpMissing(x, ...)
txpResultParam(x, ...)
```

### Arguments

x	toxiR S4 object
...	Included for extendability; not currently used
value	Replacement value

### Value

See specific methods for details.

---

txpImportGui

*Import data file generated by ToxPi GUI*

---

### Description

Import data file generated by ToxPi GUI

### Usage

```
txpImportGui(guiDataFile)
```

**Arguments**

guiDataFile      Character scalar, the path to a 'data' export from the ToxPi GUI

**Details**

This function takes the '\_data.csv' files generated by the GUI. See <https://toxpi.github.io/> for more information.

Because of the way toxpiR implements transformation functions, there is not a way currently to use the GUI 'hitcount' function.

**Value**

list with \$model containing [TxpModel](#) object; \$input containing data.frame with input data; \$fills containing a vector of fill colors.

---

TxpModel-class	<i>ToxPi Model</i>
----------------	--------------------

---

**Description**

S4 class to store ToxPi models

**Usage**

```
TxpModel(txpSlices, txpWeights = NULL, txpTransFuncs = NULL)
```

```
## S4 method for signature 'TxpModel'
txpSlices(x)
```

```
## S4 replacement method for signature 'TxpModel'
txpSlices(x) <- value
```

```
## S4 method for signature 'TxpModel'
txpWeights(x, adjusted = FALSE)
```

```
## S4 replacement method for signature 'TxpModel'
txpWeights(x) <- value
```

```
## S4 method for signature 'TxpModel'
txpTransFuncs(x)
```

```
## S4 replacement method for signature 'TxpModel'
txpTransFuncs(x) <- value
```

```
## S4 method for signature 'TxpModel'
txpValueNames(x, simplify = FALSE)
```

```
## S4 method for signature 'TxpModel'
names(x)

## S4 replacement method for signature 'TxpModel'
names(x) <- value

## S4 method for signature 'TxpModel'
length(x)

## S4 method for signature 'TxpModel,TxpModel'
merge(x, y)
```

### Arguments

txpSlices	Passed to txpSlices slot
txpWeights	Passed to txpWeights slot
txpTransFuncs	Passed to txpTransFuncs slot
x, y	TxpModel object
value	Replacement value
adjusted	Scalar logical, should the returned weights be adjusted such that they sum to 1?
simplify	Scalar logical, when TRUE the returned list is simplified

### Functions

- txpSlices(TxpModel): Return txpSlices slot
- txpWeights(TxpModel): Return txpWeights slot
- txpTransFuncs(TxpModel): Return txpTransFuncs slot
- txpValueNames(TxpModel): Return list of txpValueNames slots for the contained [TxpSliceList](#) object, or vector when simplify = TRUE
- names(TxpModel): Return slice names; shortcut for names(txpSlices(x))
- length(TxpModel): Return number of slices in model; shortcut for length(txpSlices(x))
- merge(x = TxpModel, y = TxpModel): Merge two TxpModel objects into a single model

### Slots

txpSlices [TxpSliceList](#) object

txpWeights numeric vector specifying the relative weight of each slice; when NULL, defaults to 1 (equal weighting) for each slice

txpTransFuncs [TxpTransFuncList](#) object (or list of functions coercible to TxpTransFuncList)

**Examples**

```

## Create TxpSliceList & TxpTransFuncList objects
s1 <- list(S1 = TxpSlice("inpt1"), S2 = TxpSlice("inpt2"))
tf <- list(NULL, sqrt = function(x) sqrt(x))

## Create TxpModel object
m1 <- TxpModel(txpSlices = s1, txpWeights = 2:1, txpTransFuncs = tf)
m1

## Access TxpModel slots
txpSlices(m1)
txpWeights(m1)
txpWeights(m1, adjusted = TRUE)
txpTransFuncs(m1)

## length
length(m1) ## equal to length(txpSlices(m1))
length(m1) == length(txpSlices(m1))

## names
names(m1) ## equal to names(txpSlices(m1))
all(names(m1) == names(txpSlices(m1)))

## Replacement
m2 <- m1
txpSlices(m2) <- list(S3 = TxpSlice("inpt3"), S4 = TxpSlice("inpt4"))
m2
names(m2)[2] <- "hello"
names(m2)
txpTransFuncs(m2) <- NULL
m2
txpTransFuncs(m2)[[1]] <- function(x) x^2
names(txpTransFuncs(m2))[1] <- "sq"
m2

## merge
m3 <- merge(m1, m2)
m3

```

---

TxpModelList-class      *List of TxpModel objects*

---

**Description**

Extension of [S4Vectors::SimpleList](#) that holds only [TxpModel](#) objects.

**Usage**

```
TxpModelList(...)
```

```
## S4 method for signature 'TxpModelList'
duplicated(x)
```

```
as.TxpModelList(x)
```

### Arguments

```
...           TxpModel object to create TxpModelList object
x             TxpModelList object
```

### Functions

- `duplicated(TxpModelList)`: Returns logical vector of length(x), where TRUE indicates a duplicate model in the list; see [base::duplicated](#)
- `as.TxpModelList()`: Coerce list or [TxpModel](#) objects to TxpModelList

### Examples

```
## Create some TxpModel objects; see ?TxpModel for more details
s1 <- list(S1 = TxpSlice("inpt1"), S2 = TxpSlice("inpt2"))
tf <- list(NULL, sqrt = function(x) sqrt(x))
m1 <- TxpModel(txpSlices = s1, txpWeights = 2:1, txpTransFuncs = tf)
m2 <- m1
txpSlices(m2) <- list(S3 = TxpSlice("inpt3"), S4 = TxpSlice("inpt4"))
m3 <- merge(m1, m2)

## Build a TxpModelList object
TxpModelList(m1 = m1, m2 = m2, m3 = m3)

## Note: names are printed as '' when all are NULL
TxpModelList(m1, m2, m3)
names(TxpModelList(m1, m2, m3))

## Test for duplicates
duplicated(TxpModelList(m1 = m1, m2 = m2, m3 = m3))
duplicated(TxpModelList(m1 = m1, m2 = m1, m3 = m3))

## Coerce lists/TxpModel objects to TxpModelList
as(list(m1 = m1, m2 = m2, m3 = m3), "TxpModelList")
as.TxpModelList(list(m1 = m1, m2 = m2, m3 = m3))

as(m1, "TxpModelList")
as.TxpModelList(m1)
```

---

TxpResult-class	<i>ToxPi Result</i>
-----------------	---------------------

---

**Description**

S4 class to store ToxPi results

**Usage**

```
## S4 method for signature 'TxpResult'  
txpScores(x)  
  
## S4 method for signature 'TxpResult'  
txpSliceScores(x, adjusted = TRUE)  
  
## S4 method for signature 'TxpResult'  
txpRanks(x)  
  
## S4 method for signature 'TxpResult'  
txpMissing(x)  
  
## S4 method for signature 'TxpResult'  
txpResultParam(x)  
  
## S4 method for signature 'TxpResult'  
txpModel(x)  
  
## S4 method for signature 'TxpResult'  
txpIDs(x)  
  
## S4 replacement method for signature 'TxpResult'  
txpIDs(x) <- value  
  
## S4 method for signature 'TxpResult'  
txpWeights(x, adjusted = FALSE)  
  
## S4 method for signature 'TxpResult'  
txpSlices(x)  
  
## S4 method for signature 'TxpResult'  
txpTransFuncs(x, level, simplify = FALSE)  
  
## S4 method for signature 'TxpResult'  
txpValueNames(x, simplify = FALSE)  
  
## S4 method for signature 'TxpResult,logical,missing'  
x[i, j, ..., drop = FALSE]
```

```

## S4 method for signature 'TxpResult,integer,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'TxpResult,numeric,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'TxpResult,character,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'TxpResult'
length(x)

## S4 method for signature 'TxpResult'
sort(x, decreasing = TRUE, na.last = TRUE, ...)

## S4 method for signature 'TxpResult'
names(x)

## S4 replacement method for signature 'TxpResult'
names(x) <- value

## S4 method for signature 'TxpResult'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  id.name = "id",
  score.name = "score",
  rank.name = "rank",
  adjusted = FALSE
)

```

### Arguments

x	<a href="#">TxpResult</a> object
adjusted	Logical scalar, when TRUE the weights are adjusted to sum to 1 or the slice scores are scaled to their respective weight
value	Replacement value
level	c('model', 'slices'); indicates whether to retrieve txpTransFuncs slot from the model or underlying slices
simplify	Logical scalar, flatten txpValueNames or txpTransFunc slots when retrieving slice-level information
i	Subsetting index
j, drop, optional	Not currently implemented



... Passed to [base::data.frame](#) in `as.data.frame` or [base::sort](#) in `sort`  
 decreasing, na.last  
 Passed to [base::sort](#)  
 row.names Passed to [base::data.frame](#)  
 id.name, score.name, rank.name  
 Character scalar; when coercing to [base::data.frame](#), the name for the txpIDs,  
 txpScores, and txpRanks columns, respectively

## Functions

- `txpScores(TxpResult)`: Return txpScores slot
- `txpSliceScores(TxpResult)`: Return txpSliceScores slot; default `adjusted = TRUE`, i.e. return slice scores adjusted for weight
- `txpRanks(TxpResult)`: Return txpRanks slot
- `txpMissing(TxpResult)`: Return txpMissing slot
- `txpResultParam(TxpResult)`: Return txpResultParam slot
- `txpModel(TxpResult)`: Return txpModel slot
- `txpIDs(TxpResult)`: Return txpIDs slot
- `txpWeights(TxpResult)`: Return txpWeights slot from model – shortcut for `txpWeights(txpModel(x))`; default `adjusted = FALSE`, i.e. return unadjusted weights
- `txpSlices(TxpResult)`: Return txpSlices slot from model – shortcut for `txpSlices(txpModel(x))`
- `txpTransFuncs(TxpResult)`: Return txpTransFuncs slot from model – shortcut for `txpTransFuncs(txpModel(x))`
- `txpValueNames(TxpResult)`: Return txpValueNames slot from slices – shortcut for `txpValueNames(txpSlices(txpModel(x)))`
- `length(TxpResult)`: Return the number of observations; shortcut for `length(txpScores(x))`
- `sort(TxpResult)`: Sort the “TxpResult” object by their ranks
- `names(TxpResult)`: Returns IDs; equal to `txpIDs(x)`
- `as.data.frame(TxpResult)`: Coerce TxpResult to [base::data.frame](#) object with IDs, scores, ranks, and slice scores

## Slots

`txpScores` vector(<numeric>) of model scores  
`txpSliceScores` matrix(<numeric>), sample by slice matrix with individual slice scores  
`txpRanks` vector(<numeric>) with rank of scores  
`txpMissing` vector(<numeric>) with data missingness  
`txpModel` [TxpModel](#) object  
`txpIDs` vector(<character>) of observation IDs  
`txpResultParam` [TxpResultParam](#) object

## See Also

[txpCalculateScores](#), [plot](#), [TxpResultList](#)

**Examples**

```

## Load example dataset & model; see ?TxpModel for building model objects
data(txp_example_input, package = "toxiR")
data(txp_example_model, package = "toxiR")

## Calculate scores for single model; returns TxpResult object
res <- txpCalculateScores(model = txp_example_model,
                          input = txp_example_input,
                          id.var = "name")

## Accessors
txpScores(res)

txpSliceScores(res) ## adjusted for weight, by default
apply(txpSliceScores(res), 2, max, na.rm = TRUE)

txpSliceScores(res, adjusted = FALSE) ## each score should have maximum of 1
apply(txpSliceScores(res, adjusted = FALSE), 2, max, na.rm = TRUE)

txpRanks(res)

txpMissing(res)

txpModel(res)
identical(txpModel(res), txp_example_model)

txpIDs(res)
names(res) ## identical to txpIDs(res)
identical(txpIDs(res), names(res))

# Can access TxpModel slots directly
txpWeights(res)
txpWeights(res, adjusted = TRUE)
txpSlices(res)
# When retrieving transform functions, must specify level because both
# models and slices have transform functions
txpTransFuncs(res, level = "model")

# Can access TxpSliceList slots directly
txpValueNames(res)
txpValueNames(res, simplify = TRUE)
txpTransFuncs(res, level = "slices")
txpTransFuncs(res, level = "slices", simplify = TRUE)

## Subsetting
res[1]
res[c("chem01", "chem09")]
res[grepl("4|6", txpIDs(res))]
## Not run:
res[c(TRUE, FALSE)] ## gets recycled with warning

## End(Not run)

```

```

## length -- returns number of observations
length(res)
length(res[1:5])

## sort
names(res)
names(sort(res))

txpScores(res)
txpScores(sort(res))
txpScores(sort(res, decreasing = FALSE))

## as.data.frame
as.data.frame(res)
as.data.frame(res, id.name = "nm", score.name = "scr", rank.name = "rnk")

```

---

TxpResult-plot

*Plot TxpResult objects*


---

## Description

Plot [TxpResult](#) objects

## Usage

```

## S4 method for signature 'TxpResult,missing'
plot(
  x,
  package = c("grid", "ggplot2"),
  fills = NULL,
  showScore = TRUE,
  gp = NULL,
  vp = NULL,
  name = NULL,
  newpage = TRUE,
  ...,
  ncol = NULL,
  bgColor = "grey80",
  borderColor = "white",
  sliceBorderColor = "white",
  sliceValueColor = NULL,
  sliceLineColor = NULL,
  showMissing = TRUE,
  showCenter = TRUE
)

## S4 method for signature 'TxpResult,numeric'

```

```

plot(
  x,
  y,
  labels = NULL,
  newpage = TRUE,
  margins = c(4, 0, 1, 1),
  name = NULL,
  gp = NULL,
  vp = NULL,
  ...
)

```

### Arguments

x	<a href="#">TxpResult</a> object
package	Character scalar, choice of "grid" or "ggplot2" for plotting ToxPi profiles
fills	Vector of colors to fill slices. Set to NULL to use default
showScore	Logical scalar, overall score printed below the name when TRUE
gp, vp, name	Passed to <a href="#">grid::frameGrob</a> when creating the plotting area
newpage	Logical scalar, <a href="#">grid::grid.newpage</a> called prior to plotting when TRUE
...	Passed to <a href="#">pieGridGrob</a> when plotting ToxPi and to <a href="#">pointsGrob</a> when plotting ranks
ncol	Number of columns for ggplot2 ToxPi profiles
bgColor, sliceLineColor	borderColor, sliceBorderColor, sliceValueColor, Various color options when creating ggplot2 ToxPi profiles. Set to NULL for no color
showMissing	
showCenter	Boolean for showing inner circle in ggplot2 ToxPi profiles. When set to False overrides showMissing
y	Rank vector, i.e. <code>txpRanks(x)</code>
labels	Integer vector, indices of x to label in the rank plot
margins	Passed to <a href="#">grid::plotViewport</a> ; only affects the scatterplot region margins

### Details

It is strongly recommended to use a specific device (e.g., [grDevices::png](#), [grDevices::pdf](#)) when creating rank plots. Using a GUI device will likely lead to inaccurate labeling, and any changes to the device size WILL lead to inaccurate labeling.

The plotting is built on the [grid::grid-package](#), and can be adjusted or edited as such.

If the labels are running off the device, the top or bottom margins can be increased with the `margins` parameter.

ToxPi profiles can also be plotted using the `ggplot2` package.

**Value**

No return value when using grid; called for side effect (i.e. drawing in current graphics device).  
Will return ggplot2 object otherwise.

**Functions**

- `plot(x = TxpResult, y = missing)`: Plot ToxPi diagrams
- `plot(x = TxpResult, y = numeric)`: Plot ToxPi ranks

**Examples**

```
## Load example dataset & model; see ?TxpModel for building model objects
data(txp_example_input, package = "toxiR")
data(txp_example_model, package = "toxiR")

## Calculate scores for single model; returns TxpResult object
res <- txpCalculateScores(model = txp_example_model,
                          input = txp_example_input,
                          id.var = "name")

library(grid)
plot(res)
plot(res[order(txpRanks(res))[1:4]])

library(ggplot2)
plot(res, package = "gg")
plot(res[order(txpRanks(res))], package = "gg", ncol = 5) +
  theme(legend.position = "bottom")

plot(res, txpRanks(res))
plot(res, txpRanks(res), pch = 16, size = unit(0.75, "char"))

## Will likely make inaccurate labels within a GUI, e.g. RStudio
## use png, pdf, etc. to get accurate labels
## Not run:
tmpPdf <- tempfile()
pdf(tmpPdf)
plot(res, txpRanks(res), labels = c(10, 4, 2), pch = 16)
dev.off()

## End(Not run)
```

---

TxpResultList-class    *List of TxpResult objects*

---

**Description**

Extension of [S4Vectors::SimpleList](#) that holds only [TxpResult](#) objects.

**Usage**

```
TxpResultList(...)

## S4 method for signature 'TxpResultList'
duplicated(x)

as.TxpResultList(x)
```

**Arguments**

```
...      TxpResult object to create TxpResultList object
x        TxpResultList object
```

**See Also**

[TxpResult](#), [txpCalculateScores](#)

**Examples**

```
## Load example dataset & model; see ?TxpModel for building model objects
data(txp_example_input, package = "toxpiR")
data(txp_example_model, package = "toxpiR")

## Calculate scores for list of models; returns TxpResultList object
txpCalculateScores(model = TxpModelList(m1 = txp_example_model,
                                         m2 = txp_example_model),
                  input = txp_example_input,
                  id.var = "name")
resLst <- txpCalculateScores(model = list(m1 = txp_example_model,
                                         m2 = txp_example_model),
                           input = txp_example_input,
                           id.var = "name")

## duplicated
duplicated(resLst)

## Coercion
as(list(resLst[[1]], resLst[[2]]), "TxpResultList")
as.TxpResultList(list(res1 = resLst[[1]], res2 = resLst[[2]]))

as(resLst[[1]], "TxpResultList")
as.TxpResultList(resLst[[1]])
```

---

TxpResultParam-class    *ToxPi Result Parameters*

---

**Description**

S4 class to store ToxPi result calculation parameters

**Arguments**

`rank.ties.method`  
 Passed to `rank.ties.method` slot

`negative.value.handling`  
 Passed to `negative.value.handling` slot

**Details**

If more than one value is passed to TxoResultParam scalar options, e.g. `rank.ties.method`, only the first value is kept.

The `rank.ties.method` slot is passed to `base::rank` for calculating the ranks of observations, with the highest-scoring observation having the rank of 1.

`negative.value.handling` indicates how to handle negative values in the inputs. The ToxPi algorithm originally intended to accept non-negative potency values; the GUI, therefore, treats negative values in the input as missing. By default, `txpCalculateScores` keeps negative values (`negative.value.handling = "keep"`). To replicate the GUI behavior, users can set `negative.value.handling = "missing"`.

**Slots**

`rank.ties.method` Character scalar, method used to calculate score ranks passed to `base::rank`

`negative.value.handling` Character scalar, how negative values are handled, see details

**See Also**

[txpCalculateScores](#), [TxpResult](#)

---

TxpSlice-class

*ToxPi Slice*

---

**Description**

S4 class to store ToxPi slices

**Usage**

```
TxpSlice(txpValueNames, txpTransFuncs = NULL)

## S4 method for signature 'TxpSlice'
txpValueNames(x)

## S4 replacement method for signature 'TxpSlice'
txpValueNames(x) <- value

## S4 method for signature 'TxpSlice'
txpTransFuncs(x)
```

```
## S4 replacement method for signature 'TxpSlice'
txpTransFuncs(x) <- value

## S4 method for signature 'TxpSlice'
length(x)

## S4 method for signature 'TxpSlice,TxpSlice'
merge(x, y)
```

### Arguments

txpValueNames	Passed to txpValueNames slot
txpTransFuncs	Passed to txpTransFuncs slot
x, y	TxpSlice object
value	Replacement value

### Details

If the user supplies txpTransFuncs a single function/[TxpTransFunc](#) object, the given function will be recycled for each input with a warning.

### Functions

- txpValueNames(TxpSlice): Return txpValueNames slot
- txpTransFuncs(TxpSlice): Return txpTransFuncs slot
- length(TxpSlice): Return number of inputs in slice; shortcut for length(txpValueNames(x))
- merge(x = TxpSlice, y = TxpSlice): Merge two TxpSlice objects into a single slice

### Slots

txpValueNames vector(<character>) specifying the input columns to include in the slice  
txpTransFuncs [TxpTransFuncList](#) with one function per entry in txpValueNames or an object that can be coerced to TxpTransFuncList; when NULL, no transformation function applied

### Examples

```
## Create TxpSlice object
# Without transform functions
TxpSlice(txpValueNames = c("sqrData", "expData"))
# With transform functions
TxpSlice(txpValueNames = c("sqrData", "expData"),
         txpTransFuncs = c(sq = function(x) x^2, log = function(x) log(x)))

# Transformation function recycled with warning when single function given
TxpSlice(txpValueNames = c("sqrData", "expData"),
         txpTransFuncs = function(x) x^2)
```



```

## Access TpxSlice slots
s1 <- TpxSlice(txpValueNames = c("sqrData", "expData"),
              txpTransFuncs = c(sq = function(x) x^2,
                                log = function(x) log(x)))

txpValueNames(s1)
txpTransFuncs(s1)

## Replacement
txpValueNames(s1)[1] <- "hello"
s1

txpTransFuncs(s1)[[2]](exp(1))
txpTransFuncs(s1)[[2]] <- function(x) sqrt(x)
txpTransFuncs(s1)[[2]](exp(1))

# Note that replacing a single list element does NOT update the name
s1
names(txpTransFuncs(s1))[2] <- "sqrt"
s1

# Replacing the whole list DOES update the names
txpTransFuncs(s1) <- list(sqrt = function(x) sqrt(x),
                          log = function(x) log(x))
s1

## length -- returns number of inputs
length(TpxSlice(letters))

## merge
s1 <- TpxSlice("hello")
s2 <- TpxSlice("data")
merge(s1, s2)

# Note, input names still must be unique
## Not run: merge(s1, s1) ## produces error

```

---

TpxSliceList-class      *List of TpxSlice objects*

---

## Description

Extension of [S4Vectors::SimpleList](#) that requires uniquely-named elements and holds only [TpxSlice](#) objects.

## Usage

```

TpxSliceList(...)

## S4 method for signature 'TpxSliceList'
txpValueNames(x, simplify = FALSE)

```

```
## S4 method for signature 'TxpSliceList'
txpTransFuncs(x, simplify = FALSE)

## S4 method for signature 'TxpSliceList'
duplicated(x)

as.TxpSliceList(x)
```

### Arguments

...	<a href="#">TxpSlice</a> object to create <a href="#">TxpSliceList</a> object; MUST give unique names to each slice
x	<a href="#">TxpSliceList</a> object
simplify	Scalar logical, when TRUE the returned list is simplified to a vector/ <a href="#">TxpTransFuncList</a> object

### Details

Note, there is no coercion for [TxpSlice](#) to [TxpSliceList](#) because unique names are required.

### Functions

- `txpValueNames(TxpSliceList)`: Return list of `txpValueNames` slots for the contained [TxpSlice](#) objects, or vector when `simplify = TRUE`
- `txpTransFuncs(TxpSliceList)`: Return list of `txpTransFuncs` slots for the contained [TxpSlice](#) objects, or [TxpTransFuncList](#) when `simplify = TRUE`
- `duplicated(TxpSliceList)`: Returns logical vector of length(`x`), where TRUE indicates a duplicate slice in the list; see [base::duplicated](#)

### Examples

```
## Create TxpSlice objects
s1 <- TxpSlice("input1", list(linear = function(x) x))
s2 <- TxpSlice(c("input2", "input3"),
              list(log = function(x) log(x), sqrt = function(x) sqrt(x)))

## Create TxpSliceList
s1 <- TxpSliceList(s1 = s1, s2 = s2)

## Accessors
txpValueNames(s1)
txpValueNames(s1, simplify = TRUE)

txpTransFuncs(s1)
txpTransFuncs(s1, simplify = TRUE)

## Coercion
as(list(s1 = TxpSlice("hello"), s2 = TxpSlice("user")), "TxpSliceList")
as.TxpSliceList(c(s1 = TxpSlice("hello"), s2 = TxpSlice("user")))
```

```
## Concatenation
c(sl, TxpSliceList(s3 = TxpSlice("input4")))

## Reduce TxpSliceList to single slice
Reduce(merge, sl)
```

---

TxpTransFunc-class      *Numeric transformation function*

---

## Description

S4 class to store numeric transformation functions

## Usage

```
TxpTransFunc(x)
```

## Arguments

x                      function, see details

## Details

TxpTransFunc inherits from a standard R function, but specifies a single input and a numeric output of the same length.

Functions can be passed directly to TxpTransFuncList list and the functions will be coerced to TxpTransFunc.

We have an imperfect system for dealing with primitive functions (e.g., `base::sqrt`). To coerce primitives to TxpTransFunc's, we wrap them in another function `cal`; wrapping the primitives obscures the original function and requires the user to explore the function environment to understand the primitive called. We recommend wrapping primitives in separate functions to make the intent clear, e.g., `mysqrt <- function(x) sqrt(x)`.

## Examples

```
f1 <- function(x) "hello"
f2 <- function(x) 3
f3 <- function(x) x + 5
## Not run:
t1 <- TxpTransFunc(x = f1) ## Produces error
t2 <- TxpTransFunc(x = f2) ## Produces error

## End(Not run)
t3 <- TxpTransFunc(x = f3)

## TxpTransFunc objects act as any other function
body(t3)
```

```

formals(t3)
t3(1:10)

## Coercion from functions
## Not run:
TxpTransFuncList(f1, f2, f3) ## Produces error because f1, f3 not valid

## End(Not run)

```

---

TxpTransFuncList-class

*List of TxpTransFunc objects*

---

### Description

Extension of [S4Vectors::SimpleList](#) that holds only NULL or [TxpTransFunc](#) objects.

### Usage

```

TxpTransFuncList(...)

as.TxpTransFuncList(x)

```

### Arguments

... [TxpTransFunc](#) object or function to create TxpTransFuncList object  
x list, function, or [TxpTransFunc](#) object to coerce to TxpTransFuncList

### Details

When ... includes function objects, TxpTransFuncList will attempt to coerce them to [TxpTransFunc](#) and return an error if any of the elements cannot be coerced to [TxpTransFunc](#).

### Examples

```

## Create TxpTransFunc objects
tf1 <- TxpTransFunc(function(x) x)
tf2 <- TxpTransFunc(function(x) sqrt(x))

## Create TxpTransFuncList
tf1 <- TxpTransFuncList(linear = tf1, sqrt = tf2, cube = function(x) x^3)
tf1[[3]](3) == 27
tf1[["sqrt"]](4) == 2

## Concatenate
c(tf1, tf1)

## names
names(c(tf1, tf1))

```

```
# note: names are printed as '' when missing; NULL is printed when list item
# is NULL
names(TxpTransFuncList(function(x) x, NULL))
TxpTransFuncList(function(x) x, NULL)

## coercion
as(function(x) x, "TxpTransFuncList")
as.TxpTransFuncList(function(x) x)

as(TxpTransFunc(function(x) x), "TxpTransFuncList")
as.TxpTransFuncList(TxpTransFunc(function(x) x))

as(list(function(x) x, sqrt = function(x) sqrt(x)), "TxpTransFuncList")
as.TxpTransFuncList(list(function(x) x, sqrt = function(x) sqrt(x)))
```

# Index

- [,TxpResult,character,missing-method (TxpResult-class), [15](#)
- [,TxpResult,integer,missing-method (TxpResult-class), [15](#)
- [,TxpResult,logical,missing-method (TxpResult-class), [15](#)
- [,TxpResult,numeric,missing-method (TxpResult-class), [15](#)
  
- as.data.frame, TxpResult-method (TxpResult-class), [15](#)
- as.TxpModelList (TxpModelList-class), [13](#)
- as.TxpResultList (TxpResultList-class), [21](#)
- as.TxpSliceList (TxpSliceList-class), [25](#)
- as.TxpTransFuncList (TxpTransFuncList-class), [28](#)
  
- base::data.frame, [6](#), [17](#)
- base::duplicated, [14](#), [26](#)
- base::rank, [23](#)
- base::sort, [17](#)
- base::sqrt, [27](#)
- boxLegendGrob, [2](#)
  
- duplicated, TxpModelList-method (TxpModelList-class), [13](#)
- duplicated, TxpResultList-method (TxpResultList-class), [21](#)
- duplicated, TxpSliceList-method (TxpSliceList-class), [25](#)
  
- grDevices::pdf, [20](#)
- grDevices::png, [20](#)
- grid.pieGridGrob (pieGridGrob), [3](#)
- grid.pieGrob (pieGrob), [5](#)
- grid::frameGrob, [3](#), [20](#)
- grid::grid-package, [20](#)
- grid::grid.newpage, [20](#)
- grid::grob, [4](#), [5](#)
  
- grid::gTree, [4](#), [5](#)
- grid::plotViewport, [20](#)
  
- length, TxpModel-method (TxpModel-class), [11](#)
- length, TxpResult-method (TxpResult-class), [15](#)
- length, TxpSlice-method (TxpSlice-class), [23](#)
  
- merge, TxpModel, TxpModel-method (TxpModel-class), [11](#)
- merge, TxpSlice, TxpSlice-method (TxpSlice-class), [23](#)
  
- names, TxpModel-method (TxpModel-class), [11](#)
- names, TxpResult-method (TxpResult-class), [15](#)
- names<-, TxpModel-method (TxpModel-class), [11](#)
- names<-, TxpResult-method (TxpResult-class), [15](#)
  
- pieGridGrob, [3](#), [20](#)
- pieGrob, [5](#)
- plot, [17](#)
- plot (TxpResult-plot), [19](#)
- plot, TxpResult, missing-method (TxpResult-plot), [19](#)
- plot, TxpResult, numeric-method (TxpResult-plot), [19](#)
  
- S4Vectors::SimpleList, [13](#), [21](#), [25](#), [28](#)
- sort, TxpResult-method (TxpResult-class), [15](#)
  
- toxiR-datasets, [6](#)
- txp\_example\_input (toxiR-datasets), [6](#)
- txp\_example\_model (toxiR-datasets), [6](#)
- txpCalculateScores, [6](#), [7](#), [17](#), [22](#), [23](#)

- txpCalculateScores, list, data.frame-method (txpCalculateScores), 7
- txpCalculateScores, TxpModel, data.frame-method (txpCalculateScores), 7
- txpCalculateScores, TxpModelList, data.frame-method (txpCalculateScores), 7
- txpExportGui, 8
- txpGenerics, 9
- txpIDs (txpGenerics), 9
- txpIDs, TxpResult-method (TxpResult-class), 15
- txpIDs<- (txpGenerics), 9
- txpIDs<-, TxpResult-method (TxpResult-class), 15
- txpImportGui, 10
- txpMissing (txpGenerics), 9
- txpMissing, TxpResult-method (TxpResult-class), 15
- TxpModel, 6–9, 11, 13, 14, 17
- TxpModel (TxpModel-class), 11
- txpModel (txpGenerics), 9
- txpModel, TxpResult-method (TxpResult-class), 15
- TxpModel-class, 11
- TxpModel-txpSlices (TxpModel-class), 11
- TxpModelList, 7–9
- TxpModelList (TxpModelList-class), 13
- TxpModelList-class, 13
- txpRanks (txpGenerics), 9
- txpRanks, TxpResult-method (TxpResult-class), 15
- TxpResult, 6, 8, 16, 19–23
- TxpResult (TxpResult-class), 15
- TxpResult-class, 15
- TxpResult-plot, 19
- TxpResultList, 8, 17
- TxpResultList (TxpResultList-class), 21
- TxpResultList-class, 21
- TxpResultParam, 8, 17
- TxpResultParam (TxpResultParam-class), 22
- txpResultParam (txpGenerics), 9
- txpResultParam, TxpResult-method (TxpResult-class), 15
- TxpResultParam-class, 22
- txpScores (txpGenerics), 9
- txpScores, TxpResult-method (TxpResult-class), 15
- TxpSlice, 25, 26
- TxpSlice (TxpSlice-class), 23
- TxpSlice-class, 23
- TxpSliceList, 12
- TxpSliceList (TxpSliceList-class), 25
- TxpSliceList-class, 25
- txpSlices (txpGenerics), 9
- txpSlices, TxpModel-method (TxpModel-class), 11
- txpSlices, TxpResult-method (TxpResult-class), 15
- txpSlices<- (txpGenerics), 9
- txpSlices<-, TxpModel-method (TxpModel-class), 11
- txpSliceScores (txpGenerics), 9
- txpSliceScores, TxpResult-method (TxpResult-class), 15
- TxpTransFunc, 24, 28
- TxpTransFunc (TxpTransFunc-class), 27
- TxpTransFunc-class, 27
- TxpTransFuncList, 12, 24, 26
- TxpTransFuncList (TxpTransFuncList-class), 28
- TxpTransFuncList-class, 28
- txpTransFuncs (txpGenerics), 9
- txpTransFuncs, TxpModel-method (TxpModel-class), 11
- txpTransFuncs, TxpResult-method (TxpResult-class), 15
- txpTransFuncs, TxpSlice-method (TxpSlice-class), 23
- txpTransFuncs, TxpSliceList-method (TxpSliceList-class), 25
- txpTransFuncs<- (txpGenerics), 9
- txpTransFuncs<-, TxpModel-method (TxpModel-class), 11
- txpTransFuncs<-, TxpSlice-method (TxpSlice-class), 23
- txpValueNames (txpGenerics), 9
- txpValueNames, TxpModel-method (TxpModel-class), 11
- txpValueNames, TxpResult-method (TxpResult-class), 15
- txpValueNames, TxpSlice-method (TxpSlice-class), 23
- txpValueNames, TxpSliceList-method (TxpSliceList-class), 25
- txpValueNames<- (txpGenerics), 9

`txpValueNames<-`, `TxpSlice`-method  
    (`TxpSlice`-class), [23](#)  
`txpWeights` (`txpGenerics`), [9](#)  
`txpWeights`, `TxpModel`-method  
    (`TxpModel`-class), [11](#)  
`txpWeights`, `TxpResult`-method  
    (`TxpResult`-class), [15](#)  
`txpWeights<-` (`txpGenerics`), [9](#)  
`txpWeights<-`, `TxpModel`-method  
    (`TxpModel`-class), [11](#)  
  
`utils::data`, [6](#)