Package: tidytensor (via r-universe)

July 3, 2024

Type Package

Title TidyTensor

Version 1.0.0

Date 2018-12-05

Description Provides functions for working with tensors (vectors, matrices, or multidimensional arrays) as named hierarchical data structures.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, reticulate, stringr, keras, dplyr, covr, knitr, markdown, ggplot2, tidyr

Imports abind, purrr, rstackdeque, magrittr, rlang, tidyselect

RoxygenNote 7.1.1

VignetteBuilder knitr

Repository https://r-multiverse.r-universe.dev

RemoteUrl https://github.com/oneilsh/tidytensor

RemoteRef v1.0.0

RemoteSha 14f5b87d2dfae20eb35cfd974adf660a4fd89980

Contents

`as.data.frame.tidytensor`

Convert a tidytensor to a data.frame representation.

Description

Given a tidytensor, returns a data.frame, with each rank of the tensor being represented by a column. Produces an error if the resulting data.frame would have more than 10 million entries and `allow_huge = FALSE`.

Usage

```
## S3 method for class 'tidytensor'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	input to convert to a data.frame
`row.names`	NULL (default) or character vector giving the new row names for the data frame (included for method compatibility with base `as.data.frame`).
`optional`	Ignored (included for method compatibility with base `as.data.frame`)
`...`	additional arguments to be passed to or from methods (ignored).

Details

Note that this produces a row for each value in the tensor, and a column for each rank; data.frames are a much less efficient representation, but can be useful for e.g. visualization purposes. This method thus produces an error if the resulting data.frame would have more than 10 million entries and `allow_huge = FALSE` is set (default is `TRUE`). If dimnames() are set (naming each dimension withina rank), then the columns will be factors, rather than integer indices.

If the tidytensor ranks are not named, columns will be named `index_1`, `index_2`, etc., otherwise they will be set to ranknames. Tensor values will be in a column named `value`.

Value

a data.frame

See Also

ranknames.

Examples

```
# From an array (representing e.g. 30 26x26 images (30 sets of 26 rows of 26 pixels))
a <- array(rnorm(30 * 26 * 26), dim = c(30, 26, 26))
t <- as.tidytensor(a)
ranknames(t) <- c("sample", "row", "pixel")
df <- as.data.frame(t)
print(head(df))

# Example with named dimensions:
dimnames(t)[[1]] <- paste("sample", 1:30, sep = "_")
dimnames(t)[[2]] <- paste("row", 1:26, sep = "_")
dimnames(t)[[3]] <- paste("pixel", 1:26, sep = "_")
# or with a list:
dimnames(t) <- list(paste("sample", 1:30, sep = "_"),
                    paste("row", 1:26, sep = "_"),
                    paste("pixel", 1:26, sep = "_"))

print(head(as.data.frame(t)))
```

---

as.list.tidytensor *Convert a tidytensor into a nested list of tensors.*

Description

Convert a tidytensor into a nested list of tensors, nested down to level specified in `rank`. If `flatten = TRUE`, returns a flattens the structure to a list of tensors (not nested).

Usage

```
## S3 method for class 'tidytensor'
as.list(x, rank = 1, flatten = TRUE, state = NULL, ...)
```

Arguments

`x`	the tidytensor to convert.
`rank`	an indicator of the rank defining the contained tensors.
`flatten`	whether to return a nested list (`FALSE`) or a flattened list (`TRUE`).
`state`	an internally used parameter for tracking build state-do not set manually.
`...`	additional arguments passed to methods (unusued).

Details

The `state` parameter is for internal use, and needn't be set during normal usage.

Value

a list.

See Also

as.data.frame.tidytensor

Examples

```
# Three tidytensors of the same shape
t1 <- as.tidytensor(array(100 * 1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t1) <- c("sample", "row", "col")
l1 <- as.list(t1)
str(l1)
```

---

as.tidytensor *Convert a vector, matrix, or array to a tidytensor type.*

Description

Given a vector, matrix, or array, returns a tidytensor. If given a vector, converts to a 1-d array supporting dim(), matrices are left as matrices, and in all cases the class 'tidytensor' is added.

Usage

```
as.tidytensor(x, ...)
```

Arguments

x	input to convert to a tidytensor.
...	additional arguments to be passed to or from methods (ignored).

Details

Matrices are synonymous with 2-d arrays, so these are left as is. Vectors are converted to 1-d arrays so that they can support dim().

Value

a new tidytensor.

See Also

tt, ranknames.

Examples

```
# From an array (representing e.g. 30 26x26 images (30 sets of 26 rows of 26 pixels))
a <- array(rnorm(30 * 26 * 26), dim = c(30, 26, 26))
t <- as.tidytensor(a)
ranknames(t) <- c("sample", "row", "pixel")
print(t)

# From a matrix (representing e.g. a 26x26 image (26 rows of 26 pixels))
m <- matrix(rnorm(26 * 26), nrow = 26, ncol = 26)
t <- as.tidytensor(m)
ranknames(t) <- c("row", "pixel")
print(t)

# From a vector (representing e.g. 26 pixel values)
v <- rnorm(26)
t <- as.tidytensor(v)
ranknames(t) <- c("pixel")
print(t)
```

---

bind *Bind two or more tidytensors to create a new one with a new rank.*

Description

Given multiple tidytensors, or a list of tidytensors, binds them together to create a tidytensor of higher rank. For example, bind(x, y, z) where x, y, and z have shape [2, 3, 5] returns a new tidytensor of shape [3, 2, 3, 5].

Usage

```
bind(..., new_rank_name = NULL)
```

Arguments

...	one or more tidytensors, or a single list of them, to bind
new_rank_name	a name (length-1 character vector) for the newly created rank.

Details

All input tidytensors must have the same shape. It's also possible to set a new rankname for the newly created dimension; if ranknames were prevously unset lower ranknames are set to NA. If the input ranknames conflict, only those of the first input tidytensor will be used, and a warning will be generated.

Value

a new tidytensor.

See Also

ranknames

Examples

```
# Three tidytensors of the same shape
t1 <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
t2 <- as.tidytensor(array(10 * 1:(3 * 4 * 5), dim = c(3, 4, 5)))
t3 <- as.tidytensor(array(100 * 1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t1) <- c("sample", "row", "col")
ranknames(t2) <- c("sample", "row", "col")
ranknames(t3) <- c("sample", "row", "col")
t4 <- bind(t1, t2, t3, new_rank_name = "batch")
print(t4)
```

---

combine_ranks *Combine multiple ranks of a tensor into a single rank*

Description

Combine multiple ranks of a tensor into a single rank, for example for use in data augmentation.

Usage

```
combine_ranks(x, ..., new_rank_name = NULL, .dots = NULL)
```

Arguments

x	the tidytensor to combine ranks for.
...	ranknames or integers to combine (quoted or unquoted).
new_rank_name	Name to give the newly combined rank; by default the new rank name is constructed from the names of the combined ranks.
.dots	character or integer vector of ranknames.

Details

If all ranks being combined have dimension names, the dimension names of the newly produced rank will be combinations of those specified.

It is only possible to combine consecutive ranks; use `permute()` to first organize ranks.

Value

a new tidytensor.

See Also

permute, bind

Examples

```
# shape [5, 20, 26, 26] for 5 batches of 20 26x26 "images"
t <- as.tidytensor(array(rnorm(5 * 20 * 26 * 26), dim = c(5, 20, 26, 26)))
ranknames(t) <- c("batch", "image", "row", "col")

# given an image tidytensor (26x26), return a set of replicates with noise added
make_noisy_images <- function(t2) {
  res <- bind(t2,
              t2 + rnorm(length(t2)),
              t2 + rnorm(length(t2)),
              t2 + rnorm(length(t2)), new_rank_name = "replicate")
}

# augment the original data by replacing each image with a set of
# noisy replicates
t <- tt_apply(t, image, make_noisy_images)

# now t is shape (5, 20, 4, 26, 26)
# with ranknames (batch, image, replicate, row, col)
# let's set some dimension names

# setting to "1", "2", "3", ...
t <- set_dimnames_for_rank(t, image, .dots = 1:20)

# setting to "original", "rep1", "rep2", "rep3"
t <- set_dimnames_for_rank(t, replicate, original, rep1, rep2, rep3)

# to make it compatible with the original shape we
# combine images and replicates
t2 <- combine_ranks(t, image, replicate)

print(t2)

# since the combined ranks both have dimension names, the newly
# created rank does as well and we can verify contents
# here we see that the second batch, image 3, replicate 2 is indeed the same
print(t[2, "3", "rep2", , ])
print(t2[2, "3_rep2", , ])
```

`partition` *Partition a tidytensor into a list of smaller tidytensors of the same rank*

Description

Partitions a tensor into pieces of sizes relative to `sizes`; e.g. a tensor with shape (24, 50, 50, 3) partitioned with `partition(sizes = c(0.5, 0.5))` results in a list of two tensors of shape (12, 50, 50, 3).

Ranknames are respected for both inputs and return values.

Usage

```
partition(x, sizes = c(0.5, 0.5))
```

Arguments

x	the tidytensor to apply over.
sizes	relative sizes of partitions

Details

Entries in `sizes` are treated as relative, so `sizes = c(2, 1, 1)` is equivalent to `sizes = c(0.5, 0.25, 0.25)`. Non-integer parition boundaries are rounded down, and this may result in entries with shape (0, ...), but only when the size of the first rank is smaller than the number of partitions requested.

Value

a list of tidytensors.

See Also

`c`, `permute`

Examples

```
# shape [100, 26, 26]
t <- as.tidytensor(array(rnorm(100 * 26 * 26), dim = c(100, 26, 26)))
ranknames(t) <- c("sample", "row", "col")
print(t)

partitions <- partition(t, c(0.2, 0.8))
print(partitions)
```

`permute` *Permute the ranks of a tensor*

Description

Permute the ranks of a tensor, for example to convert between "channels first" and "channels last" representations.

Ranknames are respected for both inputs and return values.

Usage

```
permute(tensor, ..., .dots = NULL)
```

Arguments

tensor	the tidytensor permute.
...	ranknames or integers to permute by (quoted or unquoted).
.dots	character or integer vector to permute by.

Details

The rank parameter may be an integer numeric vector (for permuting by index), or character vector (for permuting by rankname).

Value

a new tidytensor.

Examples

```
# shape [20, 26, 26]
t <- as.tidytensor(array(rnorm(20 * 26 * 26), dim = c(20, 26, 26)))
ranknames(t) <- c("sample", "row", "col")
print(t)

t2 <- permute(t, col, sample, row)
t2 <- permute(t, 3, 1, 2)
t2 <- permute(t, .dots = c(3, 1, 2))
t2 <- permute(t, .dots = c("col", "sample", "row"))
```

---

print.tidytensor *Print a tidytensor.*

Description

Prints a summary of a tidytensor as a nested hierarchy of tensors of lower rank.

Usage

```
## S3 method for class 'tidytensor'
print(
  x,
  show_dimnames = FALSE,
  max_per_level = 1,
  base_rank = NULL,
  max_rows = 6,
  max_cols = 6,
  max_depth = 3,
  signif_digits = 3,
  indent = 0,
  ...
)
```

Arguments

x	a tidytensor to summarize.
show_dimnames	show the dimension names, if present, or dimension indices if not in base-level prints.
max_per_level	only show this many sub-tensors per level.
base_rank	either NULL, 1, 2, or 3 - specifies whether the inner/bottom-most tensors should be represented as rank 1, 2, or 3 in a grid (NULL for autodetect based on tensor shape, see details).
max_rows	limit the base-level prints to include this many rows (also applies to 1d prints).
max_cols	limit the base-level prints to include this many columns.
max_depth	in 3d representation, limit the base-level prints to include this many entries of the last rank.
signif_digits	number of significant digits to print for numeric tensors.
indent	indent the printout by this much (used internally).
...	additional arguments to be passed to or from methods (ignored).

Details

The base_rank argument specifies whether the lowest ranks of the tensor (displayed as a grid) should be shown as rank 2 tensors, rank 3 tensors, or rank 1 tensors; the default of NULL will select 3 if the last rank is of size 3 or 1 (assuming an image and a "channels-last" convention), 2 if the 3rd-to-last rank is length 3 or 1 (assuming an image and a "channels-first" convention) or if there are only two ranks or if the last two ranks are equal (assuming an image channel of some kind), and otherwise will default to 1.

max_per_level indicates how many replicates

See Also

print.tidytensor

Examples

```
t <- as.tidytensor(array(1:(2 * 3 * 4 * 5), dim = c(2, 3, 4, 5)))
ranknames(t) <- c("samples", "batches", "rows", "cols")
print(t, base_rank = 2)

t <- as.tidytensor(array(1:(2 * 3 * 40 * 50 * 3), dim = c(2, 3, 40, 50, 3)))
ranknames(t) <- c("sample", "batch", "row", "pixel", "channel")
print(t, max_rows = 6, max_cols = 6, max_depth = 3, show_dimnames = TRUE, base_rank = 3)
```

`ranknames` *Get ranknames of a tidytensor.*

Description

A tidytensor t may have ranknames(t); this is a character vector of the same length as dim(t) for future use. Note that ranknames(t) is independent of names(t) or dimnames(t); we are not naming elements, or the dimension names for each rank, but rank names themselves. Like names() and dimnames(), unset ranknames() are NULL.

Usage

```
ranknames(x, ...)
```

Arguments

`x`	input tidytensor to get ranknames for.
`...`	additional arguments to be passed to or from methods (ignored).

Details

Ranknames for a tidytensor t are stored as the names() attribute of dimnames(t). If dimnames(t) happens to be null, before setting ranknames() we create valid dimnames() filled with NA values. The tidytensor package also provides a specialized dimnames() which preserves ranknames when setting dimnames().

Value

character vector of the same length as dim(x), or NULL if unset.

See Also

`set_ranknames`, `ranknames<-`

Examples

```
t <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t) <- c("sample", "row", "col")
print(ranknames(t))
```

`ranknames<-` *Assign ranknames to a tidytensor.*

Description

A tidytensor t may have ranknames(t); this is a character vector of the same length as dim(t) for future use. Note that ranknames(t) is independent of names(t) or dimnames(t); we are not naming elements, or the dimension names for each rank, but rank names themselves. Like names() and dimnames(), unset ranknames() are NULL.

Usage

```
ranknames(x) <- value
```

Arguments

`x`	input tidytensor to set ranknames on.
`value`	what to store in ranknames(x).

Details

Ranknames for a tidytensor t are stored as the names() attribute of dimnames(t). If dimnames(t) happens to be null, before setting ranknames() we create valid dimnames() filled with NA values. The tidytensor package also provides a specialized dimnames() which preserves ranknames when setting dimnames().

See Also

`set_ranknames`, `dimnames<-`

Examples

```
t <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t) <- c("sample", "row", "col")
print(t)

# works like names():
t <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t) <- c("sample", "row", "col")
print(ranknames(t))
ranknames(t)[3] <- "pixel"
print(t)
```

`set_dimnames` *Set dimnames() via a standard function call.*

Description

Since tidytensors are arrays, they support dimnames(). The usuall syntax dimnames(x) <- works; this function provides a Magritte-compatible regular function, set_dimnames(x, newnames) which returns a new tidytensor.

Usage

```
set_dimnames(x, newnames, ...)
```

Arguments

`x`	input tidytensor to set dimnames on.
`newnames`	list of dimnames to assign.
`...`	additional arguments to be passed to or from methods (ignored).

Details

Setting dimnames with set_dimnames() preserves any ranknames present.

Value

a tidytensor with dimnames set.

See Also

`ranknames<-`, `dimnames`

Examples

```
t <- as.tidytensor(array(1:(3 * 2), dim = c(3, 2)))
t <- set_dimnames(t, list(c("sample1", "sample2", "sample3"), c("valset1", "valset2")))
print(t)

# We can also assign ranknames:
ranknames(t) <- c("sample", "valset")
print(t)
```

`set_dimnames_for_rank` *Set dimnames() via a standard function call, for a particular rank.*

Description

Sets the dimensions names for a particular rank, without requiring dimnames for the other ranks.

Usage

```
set_dimnames_for_rank(x, rank, ..., .dots = NULL)
```

Arguments

`x`	input tidytensor to set dimnames on.
`rank`	rank to set the dimnames on.
`...`	dimnames to assign (quoted or unquoted).
`.dots`	character vector of dimnames to assign (quoted or unquoted).

Details

If all dimnames are unset, they will be set to NA for the other ranks, otherwise they will be left alone.

Value

a tidytensor with dimnames set.

See Also

`ranknames<-`, `dimnames`, `set_dimnames`

Examples

```
t <- as.tidytensor(array(1:(3 * 2), dim = c(3, 2)))
t <- set_dimnames_for_rank(t, 2, valset1, valset2)
t <- set_dimnames_for_rank(t, 2, .dots = c("valset1", "valset2"))
print(t)
```

set_ranknames *Assign ranknames to a tidytensor via a standard function call.*

Description

A tidytensor t may have ranknames(t); this is a character vector of the same length as dim(t) for future use. Note that ranknames(t) is independent of names(t) or dimnames(t); we are not naming elements, or the dimension names for each rank, but rank names themselves. Like names() and dimnames(), unset ranknames() are NULL.

Usage

```
set_ranknames(x, ..., .dots = NULL)
```

Arguments

x	input tidytensor to set ranknames on.
...	new ranknames to assign (quoted or unquoted).
.dots	character vector of new ranknames to assign.

Details

Ranknames for a tidytensor t are stored as the names() attribute of dimnames(t). If dimnames(t) happens to be null, before setting ranknames() we create valid dimnames() filled with NA values. The tidytensor package also provides a specialized dimnames() which preserves ranknames when setting dimnames().

Value

a tidytensor with ranknames set.

See Also

ranknames<-

Examples

```
t <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
t <- set_ranknames(t, sample, row, col)
t <- set_ranknames(t, .dots = c("sample", "row", "col"))
print(t)
```

shuffle *Shuffle a tidytensor in the first rank.*

Description

Shuffle's the entries in the first rank of a tensor. For example, if x has shape (3, 5, 5), it may be indexed as `x[c(2, 3, 1), , ]`. It's possible to set a custom seed for repeatable shuffling (amongst tensors with the same size in the first rank).

Usage

```
shuffle(t, seed = NULL)
```

Arguments

t	the tidytensor to apply over.
seed	random seed to be used for shuffling.

Details

Since tidytensor consider tensors as representing hierarchical "set of" relationships, shuffling in any rank other than the first would permute lower entities across set boundaries in higher ranks. For example, in a set of color images of shape (500, 28, 28, 3), shuffling the last rank would re-order the channels, but identically for all the images. See `tt_apply` for applying functions (such as shuffle) over lower ranks of a tensor.

Value

a tidytensor of the same shape.

See Also

`tt_apply`, `permute`

Examples

```
# shape [100, 26, 26]
t <- as.tidytensor(array(rnorm(100 * 26 * 26), dim = c(100, 26, 26)))
ranknames(t) <- c("sample", "row", "col")
print(t)

t <- shuffle(t, seed = 42)
```

stitch *Concatenate two or more tidytensors to create a new one with the same rank*

Description

Given multiple tidytensors of the same shape except the first rank, concatenates them together to create a tidytensor of the same shape, but larger in the first. For example, c(x, y, z) where x and have shape [2, 3, 5] and z has shape [10, 3, 5] returns a new tidytensor of shape [14, 3, 5].

Usage

```
stitch(...)
```

Arguments

...	a number of tidytensors of the same shape, or a single list of them.

Details

All input tidytensors must have the same shape except for the first rank. If the input ranknames conflict, only those of the first input tidytensor will be used, and a warning will be generated.

Value

a new tidytensor.

See Also

bind

Examples

```
# Three tidytensors of the same shape
t1 <- as.tidytensor(array(1:(3 * 4 * 5), dim = c(3, 4, 5)))
t2 <- as.tidytensor(array(10 * 1:(3 * 4 * 5), dim = c(3, 4, 5)))
t3 <- as.tidytensor(array(100 * 1:(3 * 4 * 5), dim = c(3, 4, 5)))
ranknames(t1) <- c("sample", "row", "col")
ranknames(t2) <- c("sample", "row", "col")
ranknames(t3) <- c("sample", "row", "col")
t4 <- stitch(t1, t2, t3)
print(t4)

list_example <- list(t1, t2, t3)
t5 <- stitch(list_example)
print(t5)
```

`subset.tidytensor` *Subset dimensions of a tidytensor*

Description

A functional form of e.g. `tensor[1:10, 3, ]`, supporting selecting by ranknames, usage with indexing when the rank is unknown.

Usage

```
## S3 method for class 'tidytensor'
subset(x, ..., drop = TRUE)
```

Arguments

`x`	the tidytensor to apply over.
`...`	named or unnamed parameters specifying subsetting criteria (see examples)
`drop`	whether to drop ranks with size 1 (similar to `x[..., drop = TRUE]`)

Details

Subsetting a tidytensor with `subset()` as opposed to `[]` allows for subsetting even when the number of ranks of the input is unknown; see examples.

Value

a tidytensor

See Also

`shuffle`, `permute`

Examples

```
# shape [100, 26, 26, 3]
t <- as.tidytensor(array(rnorm(100 * 26 * 26 * 3), dim = c(100, 26, 26, 3)))
ranknames(t) <- c("sample", "row", "col", "channel")
t <- set_dimnames_for_rank(t, channel, R, G, B)
print(t)

t2 <- subset(t, row = 1:10, sample = 27, drop = FALSE)
print(t2)

# same thing, but without named ranks (not a good idea to mixes named
# subsetting and non-named subsetting)
t2 <- subset(t, 27, 1:10, drop = FALSE)
print(t2)
```

```
# equiv of t3[1:20, , , c("G", "R", "B")] # since the last rank has dimnames()
# note the re-ordering of channel levels
t3 <- subset(t, sample = 1:20, channel = c("G", "R", "B"), drop = FALSE)
print(t3)
```

`tt` *Convert a vector, matrix, or array to a tidytensor type.*

Description

`tt()` is a convenience shorthand for `as.tidytensor()`. Given a vector, matrix, or array, returns a tidytensor. If given a vector, converts to a 1-d array supporting `dim()`, matrices are left as matrices, and in all cases the class 'tidytensor' is added.

Usage

```
tt(x, ...)
```

Arguments

`x`	input to convert to a tidytensor.
`...`	additional arguments to be passed to or from methods (ignored).

Details

Matrices are synonymous with 2-d arrays, so these are left as is. Vectors are converted to 1-d arrays so that they can support `dim()`.

Value

a new tidytensor.

See Also

`print.tidytensor`, `ranknames`.

Examples

```
# From an array (representing e.g. 30 26x26 images (30 sets of 26 rows of 26 pixels))
a <- array(rnorm(30 * 26 * 26), dim = c(30, 26, 26))
t <- tt(a)
ranknames(t) <- c("sample", "row", "pixel")
print(t)

# From a matrix (representing e.g. a 26x26 image (26 rows of 26 pixels)) using %>%
library(magrittr)
t <- matrix(rnorm(26 * 26), nrow = 26, ncol = 26) %>% tt()
ranknames(t) <- c("row", "pixel")
```

```
print(t)

# From a vector (representing e.g. 26 pixel values)
v <- rnorm(26)
t <- tt(rnorm(26))
ranknames(t) <- c("pixel")
print(t)
```

`tt_apply` *Apply a function over lower ranks of a tidytensor*

Description

Applies a function over the lower ranks of a tidytensor, collecting the results into a tidytensor. For example, if `FUN` is a function that takes a tidytensor of shape [26, 26] and returns a tidytensor of shape [13, 13], then we could apply `FUN` on a tidytensor of shape [3, 100, 26, 26] starting at rank 2 to get back one with shape [3, 100, 13, 13]. If `flatten = TRUE`, the higher ranks are collapsed to produce shape [300, 26, 26]

Ranknames are respected for both inputs and return values.

Usage

```
tt_apply(x, rank = 1, FUN, flatten = FALSE, drop_final_1 = TRUE, ...)
```

Arguments

`x`	the tidytensor to apply over.
`rank`	an indicator of the rank to apply over (see details).
`FUN`	the function to apply
`flatten`	whether to preserve the higher-rank structure, or collapse into a single rank (see description).
`drop_final_1`	If FUN returns a rank-0 tensor (length-1 vector), should it be collapsed? E.g. if final shape is (10, 10, 1), adjusts shape to (10, 10)
`...`	additional arguments passed to FUN.

Details

The `rank` argument should specify a single rank to apply over; if `ranknames(t) <- c("sample", "rows", "cols", "channels")` then `rank = 2`, `rank = "rows"`, and `rank = c(FALSE, TRUE, FALSE, FALSE)` all indicate that `FUN` will be called on tidytensors with ranknames `c("rows", "cols", "channels")`.

Value

a new tidytensor.

See Also

permute

Examples

```
# shape [20, 26, 26]
t <- as.tidytensor(array(rnorm(20 * 26 * 26), dim = c(20, 26, 26)))
ranknames(t) <- c("sample", "row", "col")
print(t)

# compute the deviation from median for each sample
dev_median <- function(t) {
  return(t - median(t))
}

median_deviations <- tt_apply(t, sample, dev_median)
print(median_deviations)
```

Index