

# Package ‘BaseSet’

January 20, 2025

**Title** Working with Sets the Tidy Way

**Version** 0.9.0

**Description** Implements a class and methods to work with sets, doing intersection, union, complementary sets, power sets, cartesian product and other set operations in a “tidy” way. These set operations are available for both classical sets and fuzzy sets. Import sets from several formats or from other several data structures.

**License** MIT + file LICENSE

**URL** <https://github.com/ropensci/BaseSet>,  
<https://docs.ropensci.org/BaseSet/>

**BugReports** <https://github.com/ropensci/BaseSet/issues>

**Depends** R (>= 4.0.0)

**Imports** dplyr (>= 1.0.0), magrittr, methods, rlang, utils

**Suggests** Biobase, covr, forcats, ggplot2, GO.db, GSEABase, knitr, org.Hs.eg.db, reactome.db, rmarkdown, spelling, testthat (>= 2.1.0)

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.2.3

**Collate** 'validity.R' 'AllClasses.R' 'AllGenerics.R'  
'BaseSet-package.R' 'GMT.R' 'GeneSetCollection.R' 'activate.R'  
'add.R' 'add\_column.R' 'add\_relation.R' 'adjacency.R'  
'arrange.R' 'c.R' 'cardinality.R' 'cartesian.R' 'complement.R'  
'data\_frame.R' 'deactivate.R' 'droplevels.R' 'elements.R'  
'extract.R' 'filter.R' 'group.R' 'group\_by.R' 'head.R'  
'incidence.R' 'independent.R' 'operations.R' 'intersection.R'  
'length.R' 'list.R' 'move\_to.R' 'mutate.R' 'names.R' 'naming.R'  
'nested.R' 'obo.R' 'power\_set.R' 'print.R' 'pull.R'  
'relations.R' 'remove.R' 'remove\_column.R' 'rename.R'  
'select.R' 'set.R' 'size.R' 'subtract.R' 'tidy-set.R' 'union.R'  
'union\_closed.R' 'utils-pipe.R' 'utils.R' 'zzz.R'

**NeedsCompilation** no

**Author** Lluís Revilla Sancho [aut, cre, cph]  
 (<<https://orcid.org/0000-0001-9747-2570>>),  
 Zebulun Arendsee [rev],  
 Jennifer Chang [rev]

**Maintainer** Lluís Revilla Sancho <lluis.revilla@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-08-22 11:30:05 UTC

## Contents

activate	3
add_column	5
add_elements	6
add_relation	7
add_relations	8
add_sets	9
adjacency	10
arrange.TidySet	11
as.data.frame.TidySet	12
as.list.TidySet	12
c.TidySet-method	13
cardinality	14
cartesian	14
complement	16
complement_element	17
complement_set	18
droplevels.TidySet	20
elements	21
element_size	22
extract-TidySet	23
filter.TidySet	25
getGAF	26
getGMT	27
getOBO	28
group	28
group_by.TidySet	30
incidence	31
independent	32
intersection	32
is.fuzzy	34
is_nested	35
length.TidySet	36
lengths.TidySet-method	37
length_set	37
move_to	38

multiply_probabilities . . . . .	39
mutate.TidySet . . . . .	40
name_elements . . . . .	41
name_elements<- . . . . .	42
name_sets . . . . .	43
name_sets<- . . . . .	44
naming . . . . .	45
nElements . . . . .	46
nRelations . . . . .	47
nSets . . . . .	48
power_set . . . . .	48
pull.TidySet . . . . .	49
relations . . . . .	50
remove_column . . . . .	52
remove_element . . . . .	53
remove_relation . . . . .	54
remove_set . . . . .	55
rename_elements . . . . .	56
rename_set . . . . .	57
select.TidySet . . . . .	58
sets . . . . .	59
set_size . . . . .	61
set_symbols . . . . .	62
show,TidySet-method . . . . .	62
size . . . . .	63
subtract . . . . .	64
tidy . . . . .	65
tidySet . . . . .	66
TidySet-class . . . . .	68
union . . . . .	69
union_closed . . . . .	70
union_probability . . . . .	71

**Index****73**

---

**activate***Determine the context of subsequent manipulations.*

---

**Description**

Functions to help to perform some action to just some type of data: elements, sets or relations. activate: To table the focus of future manipulations: elements, sets or relations. active: To check the focus on the TidySet. deactivate: To remove the focus on a specific TidySet-

**Usage**

```
activate(.data, what)
```

```
active(.data)
```

```
deactivate(.data)
```

**Arguments**

<code>.data</code>	A TidySet object.
<code>what</code>	Either "elements", "sets" or "relations"

**Value**

A TidySet object.

**See Also**

Other methods: [TidySet-class](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is.nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
a <- tidySet(relations)
elements(a) <- cbind(elements(a),
  type = c(rep("Gene", 4), rep("lncRNA", 2))
)
# Filter in the whole TidySet
filter(a, elements == "a")
filter(a, elements == "a", type == "Gene")
# Equivalent to filter_elements
filter_element(a, type == "Gene")
a <- activate(a, "elements")
active(a)
filter(a, type == "Gene")
a <- deactivate(a)
active(a)
filter(a, type == "Gene")
```

---

add_column	<i>Add column</i>
------------	-------------------

---

**Description**

Add column to a slot of the TidySet object.

**Usage**

```
add_column(object, slot, columns)

## S4 method for signature 'TidySet,character'
add_column(object, slot, columns)
```

**Arguments**

object	A TidySet object.
slot	A TidySet slot.
columns	The columns to add.

**Value**

A TidySet object.

**Methods (by class)**

- `add_column(object = TidySet, slot = character)`: Add a column to any slot

**See Also**

[rename\\_set\(\)](#)

Other column: [remove\\_column\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
```

```
TS <- tidySet(relations)
add_column(TS, "relations", data.frame(well = c(
  "GOOD", "BAD", "WORSE",
  "UGLY", "FOE", "HEY"
)))
```

---

add\_elements

*Add elements to a TidySet*

---

## Description

Functions to add elements. If the elements are new they are added, otherwise they are omitted.

## Usage

```
add_elements(object, elements, ...)
```

## Arguments

object	A <a href="#">TidySet</a> object
elements	A character vector of the elements.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

## Value

A [TidySet](#) object with the new elements.

## Note

add\_element doesn't set up any other information about the elements. Remember to add/modify them if needed with [mutate](#) or [mutate\\_element](#)

## See Also

Other add\_\*: [add\\_relations\(\)](#), [add\\_sets\(\)](#)

## Examples

```
x <- list("a" = letters[1:5], "b" = LETTERS[3:7])
a <- tidySet(x)
b <- add_elements(a, "fg")
elements(b)
```

---

add_relation	<i>Add relations</i>
--------------	----------------------

---

**Description**

Given a TidySet adds new relations between elements and sets.

**Usage**

```
add_relation(object, relations, ...)
```

```
## S4 method for signature 'TidySet,data.frame'
add_relation(object, relations)
```

**Arguments**

object	A TidySet object
relations	A data.frame object
...	Placeholder for other arguments that could be passed to the method. Currently not used.

**Value**

A TidySet object.

**Methods (by class)**

- `add_relation(object = TidySet, relations = data.frame)`: Adds relations

**See Also**

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is.nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
relations <- data.frame(
```

```
sets = c(rep("A2", 5), "B2"),
elements = letters[seq_len(6)],
fuzzy = runif(6),
new = runif(6)
)
add_relation(TS, relations)
```

---

add\_relations

*Add relations to a TidySet*

---

### Description

Adds new relations to existing or new sets and elements. If the sets or elements do not exist they are added.

### Usage

```
add_relations(object, elements, sets, fuzzy, ...)
```

### Arguments

object	A <a href="#">TidySet</a> object
elements	A character vector of the elements.
sets	A character vector of sets to be added.
fuzzy	The strength of the membership.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

### Value

A [TidySet](#) object with the new relations.

### Note

add\_relations doesn't set up any other information about the relationship. Remember to add/modify them if needed with [mutate](#) or [mutate\\_relation](#)

### See Also

[add\\_relation\(\)](#) to add relations with new sets or/and new elements.

Other add\_\*: [add\\_elements\(\)](#), [add\\_sets\(\)](#)



## Examples

```
x <- list("a" = letters[1:5], "b" = LETTERS[3:7])
a <- tidySet(x)
add_relations(a, elements = c("a", "b", "g"), sets = "d")
add_relations(a, elements = c("a", "b"), sets = c("d", "g"))
add_relations(a, elements = c("a", "b"), sets = c("d", "g"), fuzzy = 0.5)
add_relations(a,
  elements = c("a", "b"), sets = c("d", "g"),
  fuzzy = c(0.5, 0.7)
)
```

---

add\_sets

*Add sets to a TidySet*

---

## Description

Functions to add sets. If the sets are new they are added, otherwise they are omitted.

## Usage

```
add_sets(object, sets, ...)
```

## Arguments

object	A <a href="#">TidySet</a> object
sets	A character vector of sets to be added.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

## Value

A [TidySet](#) object with the new sets.

## Note

add\_sets doesn't set up any other information about the sets. Remember to add/modify them if needed with [mutate](#) or [mutate\\_set](#)

## See Also

Other add\_\*: [add\\_elements\(\)](#), [add\\_relations\(\)](#)

## Examples

```
x <- list("a" = letters[1:5], "b" = LETTERS[3:7])
a <- tidySet(x)
b <- add_sets(a, "fg")
sets(b)
```

---

adjacency

*Adjacency*

---

### Description

Are two elements connected ?

### Usage

```
## S3 method for class 'TidySet'  
adjacency(object)  
  
adjacency_element(object)  
  
adjacency_set(object)  
  
## S3 method for class 'TidySet'  
adjacency(object)
```

### Arguments

object            A TidySet object

### Value

A square matrix, 1 if two nodes are connected, 0 otherwise.

### See Also

[incidence\(\)](#)

### Examples

```
x <- list("SET1" = letters[1:5], "SET2" = LETTERS[3:7])  
a <- tidySet(x)  
adjacency_element(a)  
adjacency_set(a)
```

---

arrange.TidySet	<i>Arrange the order of a TidySet</i>
-----------------	---------------------------------------

---

## Description

Use `arrange` to extract the columns of a `TidySet` object. You can use `activate` with `filter` or use the specific function. The S3 method filters using all the information on the `TidySet`.

## Usage

```
## S3 method for class 'TidySet'
arrange(.data, ...)

arrange_set(.data, ...)

arrange_element(.data, ...)

arrange_relation(.data, ...)
```

## Arguments

<code>.data</code>	The <code>TidySet</code> object
<code>...</code>	Comma separated list of variables names or expressions integer column position to be used to reorder the <code>TidySet</code> .

## Value

A `TidySet` object

## See Also

[dplyr::arrange\(\)](#) and [activate\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```
relations <- data.frame(
  sets = c(rep("A", 5), "B", rep("A2", 5), "B2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
```

```

a <- tidySet(relations)
a <- mutate_element(a,
  type = c(rep("Gene", 4), rep("lncRNA", 2))
)

b <- arrange(a, desc(type))
elements(b)
b <- arrange_element(a, elements)
elements(b)
# Arrange sets
arrange_set(a, sets)

```

---

as.data.frame.TidySet *Transforms a TidySet to a data.frame*

---

### Description

Flattens the three slots to a single big table

### Usage

```

## S3 method for class 'TidySet'
as.data.frame(x, ...)

```

### Arguments

x	The TidySet object.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

### Value

A data.frame table.

---

as.list.TidySet *Convert to list*

---

### Description

Converts a TidySet to a list.

### Usage

```

## S3 method for class 'TidySet'
as.list(x, ...)

```

**Arguments**

x                    A TidySet object to be coerced to a list.  
...                   Placeholder for other arguments that could be passed to the method. Currently not used.

**Value**

A list.

**Examples**

```
r <- data.frame(sets = c("A", "A", "A", "B", "C"),
  elements = c(letters[1:3], letters[2:3]),
  fuzzy = runif(5),
  info = rep_len(c("important", "very important"), 5))
TS <- tidySet(r)
TS
as.list(TS)
```

---

c,TidySet-method

*Combine Values into a Vector or List*

---

**Description**

This method combines TidySets. It only works if the first element is a TidySet.

**Usage**

```
## S4 method for signature 'TidySet'
c(x, ...)
```

**Arguments**

x                    A TidySet object.  
...                   Objects to be concatenated. All NULL entries are dropped.

**Examples**

```
TS <- tidySet(list(A = letters[1:5], B = letters[6]))
TS2 <- c(TS, data.frame(sets = "C", elements = "gg"))
```

---

cardinality	<i>Cardinality or membership of sets</i>
-------------	--

---

**Description**

Calculates the membership of sets according to the logic defined in FUN.

**Usage**

```
cardinality(object, sets = NULL, ...)

## S4 method for signature 'TidySet'
cardinality(object, sets, FUN = "sum", ...)
```

**Arguments**

object	A TidySet object.
sets	Character vector with the name of the sets.
...	Other arguments passed to FUN.
FUN	Function that returns a single numeric value given a vector of fuzzy values.

**Methods (by class)**

- `cardinality(TidySet)`: Cardinality of sets

**See Also**

[size\(\)](#)

**Examples**

```
rel <- list(A = letters[1:3], B = letters[1:2])
TS <- tidySet(rel)
cardinality(TS, "A")
```

---

cartesian	<i>Create the cartesian product of two sets</i>
-----------	---

---

**Description**

Given two sets creates new sets with one element of each set

**Usage**

```
cartesian(object, set1, set2, name = NULL, ...)
```

```
## S3 method for class 'TidySet'
cartesian(
  object,
  set1,
  set2,
  name = NULL,
  keep = TRUE,
  keep_relations = keep,
  keep_elements = keep,
  keep_sets = keep,
  ...
)
```

**Arguments**

object	A TidySet object.
set1, set2	The name of the sets to be used for the cartesian product
name	The name of the new set.
...	Placeholder for other arguments that could be passed to the method. Currently not used.
keep	A logical value if you want to keep.
keep_relations	A logical value if you want to keep old relations.
keep_elements	A logical value if you want to keep old elements.
keep_sets	A logical value if you want to keep old sets.

**Value**

A TidySet object with the new set

**See Also**

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b"),
  elements = letters[seq_len(6)]
```

```
)
TS <- tidySet(relations)
cartesian(TS, "a", "b")
```

---

complement

*Complement TidySet*


---

### Description

Use complement to find elements or sets the TidySet object. You can use activate with complement or use the specific function. You must specify if you want the complements of sets or elements.

### Usage

```
complement(.data, ...)
```

### Arguments

.data            The TidySet object  
...                Other arguments passed to either [complement\\_set\(\)](#) or [complement\\_element\(\)](#).

### Value

A TidySet object

### See Also

[activate\(\)](#)

Other complements: [complement\\_element\(\)](#), [complement\\_set\(\)](#), [subtract\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

### Examples

```
rel <- data.frame(
  sets = c("A", "A", "B", "B", "C", "C"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(rel)
TS %>%
  activate("elements") %>%
  complement("a")
```



```

TS %>%
  activate("elements") %>%
  complement("a", "C_a", keep = FALSE)
TS %>%
  activate("set") %>%
  complement("A")
TS %>%
  activate("set") %>%
  complement("A", keep = FALSE)
TS %>%
  activate("set") %>%
  complement("A", FUN = function(x){abs(x - 0.2)}, keep = FALSE)

```

---

complement\_element      *Complement of elements*

---

## Description

Return the objects without the elements listed

## Usage

```

complement_element(object, elements, ...)

## S4 method for signature 'TidySet,characterORfactor'
complement_element(
  object,
  elements,
  name = NULL,
  FUN = NULL,
  keep = TRUE,
  keep_relations = keep,
  keep_elements = keep,
  keep_sets = keep
)

```

## Arguments

object	A TidySet object.
elements	The set to look for the complement.
...	Placeholder for other arguments that could be passed to the method. Currently not used.
name	Name of the new set. By default it adds a "C".
FUN	A function to be applied when performing the union. The standard union is the "max" function, but you can provide any other function that given a numeric vector returns a single number.
keep	Logical value to keep all the other sets.

keep\_relations A logical value if you wan to keep old relations.  
 keep\_elements A logical value if you wan to keep old elements.  
 keep\_sets A logical value if you wan to keep old sets.

**Value**

A TidySet object.

**Methods (by class)**

- complement\_element(object = TidySet, elements = characterORfactor): Complement of the elements.

**See Also**

Other complements: [complement\\_set\(\)](#), [complement\(\)](#), [subtract\(\)](#)

Other methods that create new sets: [complement\\_set\(\)](#), [intersection\(\)](#), [subtract\(\)](#), [union\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is.nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c("A", "A", "B", "B", "C", "C"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
complement_element(TS, "a", "C_a")
complement_element(TS, "a", "C_a", keep = FALSE)
```

---

complement\_set

*Complement of a set*

---

**Description**

Return the complement for a set

**Usage**

```
complement_set(object, sets, ...)

## S4 method for signature 'TidySet,characterORfactor'
complement_set(
  object,
  sets,
  name = NULL,
  FUN = NULL,
  keep = TRUE,
  keep_relations = keep,
  keep_elements = keep,
  keep_sets = keep
)
```

**Arguments**

object	A TidySet object.
sets	The name of the set to look for the complement.
...	Placeholder for other arguments that could be passed to the method. Currently not used.
name	Name of the new set. By default it adds a "C".
FUN	A function to be applied when performing the union. The standard union is the "max" function, but you can provide any other function that given a numeric vector returns a single number.
keep	Logical value to keep all the other sets.
keep_relations	A logical value if you wan to keep old relations.
keep_elements	A logical value if you wan to keep old elements.
keep_sets	A logical value if you wan to keep old sets.

**Value**

A TidySet object.

**Methods (by class)**

- `complement_set(object = TidySet, sets = characterORfactor)`: Complement of the sets.

**See Also**

[filter\(\)](#)

Other complements: [complement\\_element\(\)](#), [complement\(\)](#), [subtract\(\)](#)

Other methods that create new sets: [complement\\_element\(\)](#), [intersection\(\)](#), [subtract\(\)](#), [union\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#),

```
group_by.TidySet(), group(), incidence(), intersection(), is.fuzzy(), is_nested(), move_to(),
mutate.TidySet(), nElements(), nRelations(), nSets(), name_elements<-(), name_sets<-(),
name_sets(), power_set(), pull.TidySet(), relations(), remove_column(), remove_element(),
remove_relation(), remove_set(), rename_elements(), rename_set(), select.TidySet(),
set_size(), sets(), subtract(), union()
```

### Examples

```
relations <- data.frame(
  sets = c("A", "A", "B", "B", "C", "C"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
complement_set(TS, "A")
```

---

droplevels.TidySet      *Drop unused elements and sets*

---

### Description

Drop elements and sets without any relation.

### Usage

```
## S3 method for class 'TidySet'
droplevels(x, elements = TRUE, sets = TRUE, relations = TRUE, ...)
```

### Arguments

x	A TidySet object.
elements	Logical value: Should elements be dropped?
sets	Logical value: Should sets be dropped?
relations	Logical value: Should sets be dropped?
...	Other arguments, currently ignored.

### Value

A TidySet object.

### Examples

```
rel <- list(A = letters[1:3], B = character())
TS <- tidySet(rel)
TS
sets(TS)
TS2 <- droplevels(TS)
TS2
sets(TS2)
```

---

elements	<i>Elements of the TidySet</i>
----------	--------------------------------

---

## Description

Given TidySet retrieve the elements or substitute them.

## Usage

```
elements(object)

elements(object) <- value

## S4 method for signature 'TidySet'
elements(object)

## S4 replacement method for signature 'TidySet'
elements(object) <- value

replace_elements(object, value)

## S4 method for signature 'TidySet'
nElements(object)
```

## Arguments

object	A TidySet object.
value	Modification of the elements.

## Value

A data.frame with information about the elements

## Methods (by class)

- `elements(TidySet)`: Retrieve the elements
- `elements(TidySet) <- value`: Modify the elements
- `nElements(TidySet)`: Return the number of elements

## See Also

[nElements\(\)](#)

Other slots: [relations\(\)](#), [sets\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#),

```
is_nested(), move_to(), mutate.TidySet(), nElements(), nRelations(), nSets(), name_elements<-(),
name_sets<-(), name_sets(), power_set(), pull.TidySet(), relations(), remove_column(),
remove_element(), remove_relation(), remove_set(), rename_elements(), rename_set(),
select.TidySet(), set_size(), sets(), subtract(), union()
```

## Examples

```
TS <- tidySet(list(A = letters[1:5], B = letters[2:10]))
elements(TS)
elements(TS) <- data.frame(elements = letters[10:1])
TS2 <- replace_elements(TS, data.frame(elements = letters[1:11]))
nElements(TS)
nElements(TS2)
```

---

element_size	<i>Calculates the size of the elements</i>
--------------	--

---

## Description

Assuming that the fuzzy values are probabilities, calculates the probability of being of different sizes for a given set.

## Usage

```
element_size(object, elements = NULL)
```

```
## S4 method for signature 'TidySet'
element_size(object, elements = NULL)
```

## Arguments

object	A TidySet object.
elements	The element from which the length is calculated.

## Value

A list with the size of the elements or the probability of having that size.

## Methods (by class)

- `element_size(TidySet)`: Calculates the number of sets an element appears with `length_set()`

**See Also**

cardinality

Other sizes: `set_size()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is.nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- data.frame(
  sets = c(rep("A", 5), "B", "C"),
  elements = c(letters[seq_len(6)], letters[6]),
  fuzzy = runif(7)
)
a <- tidySet(relations)
element_size(a)
```

---

extract-TidySet

*Extract*

---

**Description**

Operators acting on TidySet to extract or replace parts. They are designed to resemble the basic operators.

**Usage**

```
## S4 method for signature 'TidySet'
x$name

## S4 replacement method for signature 'TidySet'
x$name <- value

## S4 method for signature 'TidySet'
x[i, j, k, ..., drop = TRUE]

## S4 replacement method for signature 'TidySet'
x[i, j, k, ...] <- value

## S4 method for signature 'TidySet'
x[[i, j, ..., exact = TRUE]]

## S4 replacement method for signature 'TidySet'
x[[i]] <- value
```

**Arguments**

x	A TidySet object.
name	The data about the TidySet object to extract.
value	The value to overwrite
i	Which rows do you want to keep? By default all.
j	Which slot do you want to extract? One of "sets", "elements" or "relations".
k	Which columns do you want to extract. By default all.
...	Other arguments currently ignored.
drop	Remove remaining elements, sets and relations? Passed to all arguments of <code>droplevels()</code> .
exact	A logical value. FALSE if fuzzy matching is wanted. Add values to the TidySet. Allows to control to which slot is it added.

**Value**

Always returns a valid [TidySet](#).

**Examples**

```

TS <- tidySet(list(A = letters[1:5], B = letters[6]))
TS[, "sets", "origin"] <- sample(c("random", "non-random"), 2, replace = TRUE)
TS[, "sets", "type"] <- c("Fantastic", "Wonderful")
# This produces a warning
# Better to be explicit:
TS[, "relations", "description"] <- c("What", "can", "I", "say", "now", "?")
relations(TS)
TS[, "elements", "description"] <- rev(c("What", "can", "I", "say", "now", "?"))
elements(TS)
# Which will be deleted?
# TS$description <- NULL
TS$type
TS$origin <- c("BCN", "BDN")
# Different subsets
TS[1, "elements"]
TS[1, "sets"]
# Always print
TS
TS[, "sets", c("type", "origin")] # Same
TS[, "sets", "origin"] # Drop column type
is(TS[, "sets", "origin"])
TS[, "sets"]
TS[["A"]]
TS[["B"]]
TS[["C"]] # Any other set is the empty set

```



---

filter.TidySet	<i>Filter TidySet</i>
----------------	-----------------------

---

## Description

Use filter to subset the TidySet object. You can use activate with filter or use the specific function. The S3 method filters using all the information on the TidySet.

## Usage

```
## S3 method for class 'TidySet'
filter(.data, ...)

filter_set(.data, ...)

filter_element(.data, ...)

filter_relation(.data, ...)
```

## Arguments

.data	The TidySet object.
...	The logical predicates in terms of the variables of the sets.

## Value

A TidySet object.

## See Also

`dplyr::filter()` and `activate()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

## Examples

```
relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12),
  type = c(rep("Gene", 4), rep("lncRNA", 2))
)
```

```
TS <- tidySet(relations)
TS <- move_to(TS, from = "relations", to = "elements", column = "type")
filter(TS, elements == "a")
# Equivalent to filter_relation
filter(TS, elements == "a", sets == "a")
filter_relation(TS, elements == "a", sets == "a")
# Filter element
filter_element(TS, type == "Gene")
# Filter sets and by property of elements simultaneously
filter(TS, sets == "b", type == "lncRNA")
# Filter sets
filter_set(TS, sets == "b")
```

---

getGAF

*Read a GAF file*

---

## Description

Read a GO Annotation File (GAF) formatted file

## Usage

```
getGAF(x)
```

## Arguments

x                    A file in GAF format

## Value

A TidySet object

## References

The format is defined [here](#).

## See Also

Other IO functions: [getGMT\(\)](#), [getOBO\(\)](#)

## Examples

```
gafFile <- system.file(
  package = "BaseSet", "extdata",
  "go_human_rna_valid_subset.gaf"
)
gs <- getGAF(gafFile)
head(gs)
```

---

`getGMT`*Import GMT (Gene Matrix Transposed) files*

---

**Description**

The GMT (Gene Matrix Transposed) file format is a tab delimited file format that describes groups of genes. In this format, each row represents a group. Each group is described by a name, a description, and the genes in it.

**Usage**

```
getGMT(con, sep = "\t", ...)
```

**Arguments**

<code>con</code>	File name of the GMT file.
<code>sep</code>	GMT file field separator, by default tabs.
<code>...</code>	Other arguments passed to <code>readLines</code> .

**Value**

A `TidySet` object.

**References**

The file format is defined by the Broad Institute [here](#)

**See Also**

Other IO functions: [getGAF\(\)](#), [getOBO\(\)](#)

**Examples**

```
gmtFile <- system.file(  
  package = "BaseSet", "extdata",  
  "hallmark.gene.symbol.gmt"  
)  
gs <- getGMT(gmtFile)  
nRelations(gs)  
nElements(gs)  
nSets(gs)
```

---

`getOBO`*Read an OBO file*

---

**Description**

Read an Open Biological and Biomedical Ontologies (OBO) formatted file

**Usage**

```
getOBO(x)
```

**Arguments**

`x` Path to a file in OBO format.

**Value**

A TidySet object.

**References**

The format is described [here](#)

**See Also**

Other IO functions: [getGAF\(\)](#), [getGMT\(\)](#)

**Examples**

```
oboFile <- system.file(  
  package = "BaseSet", "extdata",  
  "go-basic_subset.obo"  
)  
gs <- getOBO(oboFile)  
head(gs)
```

---

`group`*Create a new set from existing elements*

---

**Description**

It allows to create a new set given some condition. If no element meet the condition an empty set is created.

**Usage**

```
group(object, name, ...)

## S3 method for class 'TidySet'
group(object, name, ...)
```

**Arguments**

object	A TidySet object.
name	The name of the new set.
...	A logical condition to subset some elements.

**Value**

A TidySet object with the new set.

**See Also**

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
x <- list("A" = c("a" = 0.1, "b" = 0.5), "B" = c("a" = 0.2, "b" = 1))
TS <- tidySet(x)
TS1 <- group(TS, "C", fuzzy < 0.5)
TS1
sets(TS1)
TS2 <- group(TS, "D", fuzzy < 0)
sets(TS2)
r <- data.frame(
  sets = c(rep("A", 5), "B", rep("A2", 5), "B2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12),
  type = c(rep("Gene", 2), rep("Protein", 2), rep("lncRNA", 2))
)
TS3 <- tidySet(r)
group(TS3, "D", sets %in% c("A", "A2"))
```

---

group_by.TidySet	<i>group_by TidySet</i>
------------------	-------------------------

---

## Description

Use `group_by` to group the TidySet object. You can use activate with `group_by` or with the whole data.

## Usage

```
## S3 method for class 'TidySet'
group_by(.data, ...)
```

## Arguments

<code>.data</code>	The TidySet object
<code>...</code>	The logical predicates in terms of the variables of the sets

## Value

A grouped data.frame (See The dplyr help page)

## See Also

[dplyr::group\\_by\(\)](#) and [activate\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```
relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
a <- tidySet(relations)
elements(a) <- cbind(elements(a),
  type = c(rep("Gene", 4), rep("lncRNA", 2))
)
group_by(a, elements)
```

---

incidence	<i>Incidence</i>
-----------	------------------

---

### Description

Check which elements are in which sets.

### Usage

```
incidence(object)

## S4 method for signature 'TidySet'
incidence(object)
```

### Arguments

object            Object to be coerced or tested.

### Value

A matrix with elements in rows and sets in columns where the values indicate the relationship between the element and the set.

### Methods (by class)

- `incidence(TidySet)`: Incidence of the TidySet

### See Also

[adjacency\(\)](#), [tidySet\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

### Examples

```
x <- list("a" = letters[1:5], "b" = LETTERS[3:7])
a <- tidySet(x)
incidence(a)
```

---

independent	<i>Independence of the sets</i>
-------------	---------------------------------

---

**Description**

Checks if the elements of the sets are present in more than one set.

**Usage**

```
independent(object, sets)
```

**Arguments**

object	A <code>TidySet</code> object.
sets	A character vector with the names of the sets to analyze.

**Value**

A logical value indicating if the sets are independent (TRUE) or not.

**Examples**

```
x <- list("A" = letters[1:5], "B" = letters[3:7], "C" = letters[6:10])
TS <- tidySet(x)
independent(TS)
independent(TS, c("A", "B"))
independent(TS, c("A", "C"))
independent(TS, c("B", "C"))
```

---

intersection	<i>Intersection of two or more sets</i>
--------------	---

---

**Description**

Given a `TidySet` creates a new set with the elements on the both of them following the logic defined on `FUN`.

**Usage**

```
intersection(object, sets, ...)

## S4 method for signature 'TidySet,character'
intersection(
  object,
  sets,
  name = NULL,
```



```

    FUN = "min",
    keep = FALSE,
    keep_relations = keep,
    keep_elements = keep,
    keep_sets = keep,
    ...
  )

```

### Arguments

object	A TidySet object.
sets	The character of sets to be intersect.
...	Other named arguments passed to FUN.
name	The name of the new set. By defaults joins the sets with an $\cup$ .
FUN	A function to be applied when performing the union. The standard intersection is the "min" function, but you can provide any other function that given a numeric vector returns a single number.
keep	A logical value if you want to keep originals sets.
keep_relations	A logical value if you wan to keep old relations.
keep_elements	A logical value if you wan to keep old elements.
keep_sets	A logical value if you wan to keep old sets.

### Details

#' The default uses the min function following the [standard fuzzy definition](#), but it can be changed.

### Value

A TidySet object.

### Methods (by class)

- `intersection(object = TidySet, sets = character)`: Applies the standard intersection

### See Also

Other methods that create new sets: [complement\\_element\(\)](#), [complement\\_set\(\)](#), [subtract\(\)](#), [union\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```

rel <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = c("a", "b", "c", "d", "f", "f")
)
TS <- tidySet(rel)
intersection(TS, c("A", "B")) # Default Name
intersection(TS, c("A", "B"), "C") # Set the name
# Fuzzy set
rel <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = c("a", "b", "c", "d", "f", "f"),
  fuzzy = runif(6)
)
TS2 <- tidySet(rel)
intersection(TS2, c("A", "B"), "C")
intersection(TS2, c("A", "B"), "C", FUN = function(x){max(sqrt(x))})

```

---

is.fuzzy

*Check if a TidySet is fuzzy.*


---

**Description**

Check if there are fuzzy sets. A fuzzy set is a set where the relationship between elements is given by a probability (or uncertainty).

**Usage**

```

is.fuzzy(object)

## S4 method for signature 'TidySet'
is.fuzzy(object)

```

**Arguments**

object            Object to be coerced or tested.

**Value**

A logical value.

**Methods (by class)**

- `is.fuzzy(TidySet)`: Check if it is fuzzy

**See Also**

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
TS <- tidySet(list(A = letters[1:5], B = letters[2:10]))
is.fuzzy(TS)
```

---

is\_nested

*Are some sets as elements of other sets?*

---

**Description**

Check if some elements are also sets of others. This is also known as hierarchical sets.

**Usage**

```
is_nested(object)

## S3 method for class 'TidySet'
is_nested(object)
```

**Arguments**

object            A TidySet object.

**Value**

A logical value: TRUE if there are some sets included as elements of others.

**See Also**

adjacency

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- list(A = letters[1:3], B = c(letters[4:5]))
TS <- tidySet(relations)
is_nested(TS)
TS2 <- add_relation(TS, data.frame(elements = "A", sets = "B"))
# Note that A is both a set and an element of B
TS2
is_nested(TS2)
```

---

length.TidySet	<i>Length of the TidySet</i>
----------------	------------------------------

---

**Description**

Returns the number of sets in the object.

**Usage**

```
## S3 method for class 'TidySet'
length(x)
```

**Arguments**

x                    A TidySet object.  
No replacement function is available, either delete sets or add them.

**Value**

A numeric value.

**See Also**

[dim\(\)](#), [ncol\(\)](#) and [nrow\(\)](#). Also look at [lengths\(\)](#) for the number of relations of sets.

**Examples**

```
TS <- tidySet(list(A = letters[1:5], B = letters[6]))
length(TS)
```

---

lengths, TidySet-method

*Lengths of the TidySet*

---

### Description

Returns the number of relations of each set in the object.

### Usage

```
## S4 method for signature 'TidySet'  
lengths(x, use.names = TRUE)
```

### Arguments

x                    A TidySet object.  
use.names           A logical value whether to inherit names or not.

### Value

A vector with the number of different relations for each set.

### See Also

[length\(\)](#), Use [set\\_size\(\)](#) if you are using fuzzy sets.

### Examples

```
TS <- tidySet(list(A = letters[1:5], B = letters[6]))  
lengths(TS)
```

---

length\_set

*Calculates the probability*

---

### Description

Given several probabilities it looks for how probable is to have a vector of each length

### Usage

```
length_set(probability)
```

### Arguments

probability        A numeric vector of probabilities.

**Value**

A vector with the probability of each set.

**See Also**

[length\\_probability\(\)](#) to calculate the probability of a specific length.

**Examples**

```
length_set(c(0.5, 0.1, 0.3, 0.5, 0.25, 0.23))
```

---

move\_to

*Move columns between slots*

---

**Description**

Moves information from one slot to other slots. For instance from the sets to the relations.

**Usage**

```
move_to(object, from, to, columns)

## S4 method for signature
## 'TidySet,characterORfactor,characterORfactor,character'
move_to(object, from, to, columns)
```

**Arguments**

object	A TidySet object.
from	The name of the slot where the content is.
to	The name of the slot to move the content.
columns	The name of the columns that should be moved.

**Value**

A TidySet object where the content is moved from one slot to other.

**Methods (by class)**

- `move_to(object = TidySet, from = characterORfactor, to = characterORfactor, columns = character)`: Move columns

**See Also**

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
x <- list("A" = c("a" = 0.1, "b" = 0.5), "B" = c("a" = 0.2, "b" = 1))
TS <- tidySet(x)
TS <- mutate_element(TS, b = runif(2))
TS2 <- move_to(TS, from = "elements", to = "relations", "b")
# Note that apparently we haven't changed anything:
TS2
```

---

multiply\_probabilities

*Probability of a vector of probabilities*

---

**Description**

Calculates the probability that all probabilities happened simultaneously. `independent_probabilities()` just multiply the probabilities of the index passed.

**Usage**

```
multiply_probabilities(p, i)
```

```
independent_probabilities(p, i)
```

**Arguments**

`p` Numeric vector of probabilities.  
`i` Numeric integer index of the complementary probability.

**Value**

A numeric value of the probability.

**See Also**

[length\\_probability\(\)](#)

**Examples**

```
multiply_probabilities(c(0.5, 0.1, 0.3, 0.5, 0.25, 0.23), c(1, 3))
independent_probabilities(c(0.5, 0.1, 0.3, 0.5, 0.25, 0.23), c(1, 3))
```

---

mutate.TidySet	<i>Mutate</i>
----------------	---------------

---

**Description**

Use mutate to alter the TidySet object. You can use activate with mutate or use the specific function. The S3 method filters using all the information on the TidySet.

**Usage**

```
## S3 method for class 'TidySet'
mutate(.data, ...)

mutate_set(.data, ...)

mutate_element(.data, ...)

mutate_relation(.data, ...)
```

**Arguments**

.data	The TidySet object.
...	The logical predicates in terms of the variables of the sets.

**Value**

A TidySet object

**See Also**

[dplyr::mutate\(\)](#) and [activate\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)



**Examples**

```

relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
a <- tidySet(relations)
a <- mutate_element(a, Type = c(rep("Gene", 4), rep("lncRNA", 2)))
a
b <- mutate_relation(a, Type = sample(c("PPI", "PF", "MP"), 12,
  replace = TRUE
))

```

---

name_elements	<i>Name elements</i>
---------------	----------------------

---

**Description**

Retrieve the name of the elements.

**Usage**

```

name_elements(object)

## S4 method for signature 'TidySet'
name_elements(object)

## S4 replacement method for signature 'TidySet,characterORfactor'
name_elements(object) <- value

```

**Arguments**

object            A TidySet object.  
value             A character with the new names for the elements.

**Value**

A TidySet object.

**Methods (by class)**

- name\_elements(TidySet): Name elements
- name\_elements(object = TidySet) <- value: Rename elements

**See Also**

Other names: [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
name_elements(TS)
```

---

name_elements<-	<i>Rename elements</i>
-----------------	------------------------

---

**Description**

Rename elements.

**Usage**

```
name_elements(object) <- value
```

**Arguments**

object	A TidySet object.
value	A character with the new names for the elements.

**Value**

A TidySet object.

**See Also**

[rename\\_elements\(\)](#)

Other names: [name\\_elements\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
TS
name_elements(TS) <- letters[1:6]
```

---

name_sets	<i>Name sets</i>
-----------	------------------

---

## Description

Retrieve the name of the sets.

## Usage

```
name_sets(object)

## S4 method for signature 'TidySet'
name_sets(object)

## S4 replacement method for signature 'TidySet,characterORfactor'
name_sets(object) <- value
```

## Arguments

**object**            A TidySet object.

**value**             A character with the new names for the sets.

## Value

A TidySet object.

## Methods (by class)

- name\_sets(TidySet): Name sets
- name\_sets(object = TidySet) <- value: Rename sets

**See Also**

Other names: `name_elements<-()`, `name_elements()`, `name_sets<-()`, `rename_elements()`, `rename_set()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
name_sets(TS)
```

---

name\_sets<-

*Rename sets*


---

**Description**

Rename sets.

**Usage**

```
name_sets(object) <- value
```

**Arguments**

`object` A TidySet object.  
`value` A character with the new names for the sets.

**Value**

A TidySet object.

**See Also**

[rename\\_set\(\)](#)

Other names: [name\\_elements<-\(\)](#), [name\\_elements\(\)](#), [name\\_sets\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
TS
name_sets(TS) <- LETTERS[1:2]
```

---

naming

*Name an operation*


---

**Description**

Helps setting up the name of an operation.

**Usage**

```
naming(
  start = NULL,
  sets1,
  middle = NULL,
  sets2 = NULL,
  collapse_symbol = "union"
)
```

**Arguments**

`start, middle` Character used as a start symbol or to divide `sets1` and `sets2`.  
`sets1, sets2` Character of sets  
`collapse_symbol` Name of the symbol that joins the sets on `sets1` and `sets2`.

**Value**

A character vector combining the sets

**See Also**

[set\\_symbols\(\)](#)

**Examples**

```
naming(sets1 = c("a", "b"))
naming(sets1 = "a", middle = "union", sets2 = "b")
naming(sets1 = "a", middle = "intersection", sets2 = c("b", "c"))
naming(sets1 = "a", middle = "intersection", sets2 = c("b", "c"))
naming(
  start = "complement", sets1 = "a", middle = "intersection",
  sets2 = c("b", "c"), collapse_symbol = "intersection"
)
```

---

nElements

*Number of elements*

---

**Description**

Check the number of elements of the TidySet.

**Usage**

```
nElements(object)
```

**Arguments**

object            Object to be coerced or tested.

**Value**

A numeric value with the number of elements.

**See Also**

Other count functions: [nRelations\(\)](#), [nSets\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```
TS <- tidySet(list(A = letters[1:2], B = letters[5:7]))
nElements(TS)
```

---

nRelations

*Number of relations*

---

## Description

Check the number of relations of the TidySet.

## Usage

```
nRelations(object)
```

## Arguments

object            Object to be coerced or tested.

## Value

A numeric value with the number of the relations.

## See Also

Other count functions: [nElements\(\)](#), [nSets\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```
TS <- tidySet(list(A = letters[1:2], B = letters[5:7]))
nRelations(TS)
```

---

nSets	<i>Number of sets</i>
-------	-----------------------

---

**Description**

Check the number of sets of the TidySet

**Usage**

```
nSets(object)
```

**Arguments**

object            Object to be coerced or tested.

**Value**

The number of sets present.

**See Also**

Other count functions: [nElements\(\)](#), [nRelations\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
TS <- tidySet(list(A = letters[1:2], B = letters[5:7]))
nSets(TS)
```

---

power_set	<i>Create the power set</i>
-----------	-----------------------------

---

**Description**

Create the power set of the object: All the combinations of the elements of the sets.

**Usage**

```
power_set(object, set, name, ...)
```



**Arguments**

object	A TidySet object.
set	The name of the set to be used for the power set, if not provided all are used.
name	The root name of the new set, if not provided the standard notation "P()" is used.
...	Other arguments passed down if possible.

**Value**

A TidySet object with the new set.

**See Also**

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b"),
  elements = letters[seq_len(6)]
)
TS <- tidySet(relations)
power_set(TS, "a", name = "power_set")
```

---

pull.TidySet

*Pull from a TidySet*

---

**Description**

Use pull to extract the columns of a TidySet object. You can use activate with filter or use the specific function. The S3 method filters using all the information on the TidySet.

**Usage**

```
## S3 method for class 'TidySet'
pull(.data, var = -1, name = NULL, ...)

pull_set(.data, var = -1, name = NULL, ...)

pull_element(.data, var = -1, name = NULL, ...)

pull_relation(.data, var = -1, name = NULL, ...)
```

**Arguments**

<code>.data</code>	The TidySet object
<code>var</code>	The literal variable name, a positive integer or a negative integer column position.
<code>name</code>	Column used to name the output.
<code>...</code>	Currently not used.

**Value**

A TidySet object

**See Also**

`dplyr::pull()` and `activate()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
a <- tidySet(relations)
a <- mutate_element(a, type = c(rep("Gene", 4), rep("lncRNA", 2)))
pull(a, type)
# Equivalent to pull_relation
b <- activate(a, "relations")
pull_relation(b, elements)
pull_element(b, elements)
# Filter element
pull_element(a, type)
# Filter sets
pull_set(a, sets)
```

---

relations

*Relations of the TidySet*

---

**Description**

Given TidySet retrieve the relations or substitute them. `TidySet()` object

**Usage**

```
relations(object)

relations(object) <- value

## S4 method for signature 'TidySet'
relations(object)

replace_relations(object, value)

## S4 replacement method for signature 'TidySet'
relations(object) <- value

## S4 method for signature 'TidySet'
nRelations(object)
```

**Arguments**

object	Object to be coerced or tested.
value	Modification of the relations.

**Value**

A data frame with information about the relations between elements and sets.

**Methods (by class)**

- `relations(TidySet)`: Retrieve the relations
- `relations(TidySet) <- value`: Modify the relations
- `nRelations(TidySet)`: Return the number of unique relations

**See Also**

[nRelations\(\)](#)

Other slots: [elements\(\)](#), [sets\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
TS <- tidySet(list(A = letters[1:2], B = letters[5:7]))
relations(TS)
```

---

remove_column	<i>Remove column</i>
---------------	----------------------

---

## Description

Removes column from a slot of the TidySet object.

## Usage

```
remove_column(object, slot, column_names)

## S4 method for signature 'TidySet,character,character'
remove_column(object, slot, column_names)
```

## Arguments

object	A TidySet object.
slot	A TidySet slot.
column_names	The name of the columns.

## Value

A TidySet object.

## Methods (by class)

- `remove_column(object = TidySet, slot = character, column_names = character)`: Remove columns to any slot

## See Also

[rename\\_set\(\)](#)

Other column: [add\\_column\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
x <- data.frame(sets = c(rep("A", 5), rep("B", 5)),
                elements = c(letters[1:5], letters[3:7]),
                extra = sample(c("YES", "NO"), 10, replace = TRUE))
TS <- tidySet(x)
TS
remove_column(TS, "relations", "extra")
```

---

remove_element	<i>Remove elements</i>
----------------	------------------------

---

**Description**

Given a TidySet remove elements and the related relations and if required also the sets.

**Usage**

```
remove_element(object, elements, ...)

## S4 method for signature 'TidySet,characterORfactor'
remove_element(object, elements)
```

**Arguments**

object	A TidySet object.
elements	The elements to be removed.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

**Value**

A TidySet object.

**Methods (by class)**

- `remove_element(object = TidySet, elements = characterORfactor)`: Removes everything related to an element

**See Also**

Other remove functions: [remove\\_relation\(\)](#), [remove\\_set\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
remove_element(TS, "c")
```

---

remove_relation	<i>Remove a relation</i>
-----------------	--------------------------

---

**Description**

Given a TidySet removes relations between elements and sets

**Usage**

```
remove_relation(object, elements, sets, ...)
```

```
## S4 method for signature 'TidySet,characterORfactor,characterORfactor'
remove_relation(object, elements, sets)
```

**Arguments**

object	A TidySet object
elements	The elements of the sets.
sets	The name of the new set.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

**Value**

A TidySet object.

**Methods (by class)**

- `remove_relation(object = TidySet, elements = characterORfactor, sets = characterORfactor)`: Removes a relation between elements and sets.

**See Also**

Other remove functions: [remove\\_element\(\)](#), [remove\\_set\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#),

```
name_elements<-(), name_sets<-(), name_sets(), power_set(), pull.TidySet(), relations(),
remove_column(), remove_element(), remove_set(), rename_elements(), rename_set(), select.TidySet(),
set_size(), sets(), subtract(), union()
```

## Examples

```
relations <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
remove_relation(TS, "A", "a")
```

---

remove_set	<i>Remove sets</i>
------------	--------------------

---

## Description

Given a TidySet remove sets and the related relations and if required also the elements

## Usage

```
remove_set(object, sets, ...)

## S4 method for signature 'TidySet,characterORfactor'
remove_set(object, sets)
```

## Arguments

object	A TidySet object.
sets	The sets to be removed.
...	Placeholder for other arguments that could be passed to the method. Currently not used.

## Value

A TidySet object.

## Methods (by class)

- `remove_set(object = TidySet, sets = characterORfactor)`: Removes everything related to a set

**See Also**

Other remove functions: `remove_element()`, `remove_relation()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `rename_elements()`, `rename_set()`, `select.TidySet()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- data.frame(
  sets = c("A", "A", "B", "B", "C", "C"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
remove_set(TS, "B")
```

---

rename_elements	<i>Rename elements</i>
-----------------	------------------------

---

**Description**

Change the default names of sets and elements.

**Usage**

```
rename_elements(object, old, new)
```

```
## S4 method for signature 'TidySet'
rename_elements(object, old, new)
```

**Arguments**

object	A TidySet object.
old	A character vector of to be renamed.
new	A character vector of with new names.

**Value**

A TidySet object.

**Methods (by class)**

- `rename_elements(TidySet)`: Rename elements



**See Also**

[name\\_elements\(\)](#)

Other renames: [rename\\_set\(\)](#)

Other names: [name\\_elements<-\(\)](#), [name\\_elements\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [rename\\_set\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
x <- list("A" = letters[1:5], "B" = letters[3:7])
TS <- tidySet(x)
name_elements(TS)
TS2 <- rename_elements(TS, "a", "first")
name_elements(TS2)
```

---

rename\_set

*Rename sets*

---

**Description**

Change the default names of sets and elements.

**Usage**

```
rename_set(object, old, new)
```

```
## S4 method for signature 'TidySet'
rename_set(object, old, new)
```

**Arguments**

object	A TidySet object.
old	A character vector of to be renamed.
new	A character vector of with new names.

**Value**

A TidySet object.

**Methods (by class)**

- `rename_set(TidySet)`: Rename sets

**See Also**

[name\\_sets\(\)](#)

Other renames: [rename\\_elements\(\)](#)

Other names: [name\\_elements<-\(\)](#), [name\\_elements\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [rename\\_elements\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

**Examples**

```
x <- list("A" = letters[1:5], "B" = letters[3:7])
TS <- tidySet(x)
name_sets(TS)
TS2 <- rename_set(TS, "A", "C")
name_sets(TS2)
```

---

`select.TidySet`

*select from a TidySet*

---

**Description**

Use `select` to extract the columns of a `TidySet` object. You can use `activate` with `filter` or use the specific function. The S3 method filters using all the information on the `TidySet`.

**Usage**

```
## S3 method for class 'TidySet'
select(.data, ...)

select_set(.data, ...)

select_element(.data, ...)

select_relation(.data, ...)
```

**Arguments**

`.data`            The `TidySet` object

`...`            The name of the columns you want to keep, remove or rename.

**Value**

A `TidySet` object

**See Also**

`dplyr::select()` and `activate()`

Other methods: `TidySet-class`, `activate()`, `add_column()`, `add_relation()`, `arrange.TidySet()`, `cartesian()`, `complement_element()`, `complement_set()`, `complement()`, `element_size()`, `elements()`, `filter.TidySet()`, `group_by.TidySet()`, `group()`, `incidence()`, `intersection()`, `is.fuzzy()`, `is_nested()`, `move_to()`, `mutate.TidySet()`, `nElements()`, `nRelations()`, `nSets()`, `name_elements<-()`, `name_sets<-()`, `name_sets()`, `power_set()`, `pull.TidySet()`, `relations()`, `remove_column()`, `remove_element()`, `remove_relation()`, `remove_set()`, `rename_elements()`, `rename_set()`, `set_size()`, `sets()`, `subtract()`, `union()`

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b", rep("a2", 5), "b2"),
  elements = rep(letters[seq_len(6)], 2),
  fuzzy = runif(12)
)
a <- tidySet(relations)
a <- mutate_element(a,
  type = c(rep("Gene", 4), rep("lncRNA", 2))
)
a <- mutate_set(a, Group = c("UFM", "UAB", "UPF", "MIT"))
b <- select(a, -type)
elements(b)
b <- select_element(a, elements)
elements(b)
# Select sets
select_set(a, sets)
```

---

sets

*Sets of the TidySet*

---

**Description**

Given TidySet retrieve the sets or substitute them.

**Usage**

```
sets(object)
```

```
sets(object) <- value
```

```
## S4 method for signature 'TidySet'
sets(object)
```

```
## S4 replacement method for signature 'TidySet'
sets(object) <- value
```

```
replace_sets(object, value)

## S4 method for signature 'TidySet'
nSets(object)
```

### Arguments

object	A SetCollection object.
value	Modification of the sets.

### Value

A data.frame with information from the sets.

### Methods (by class)

- `sets(TidySet)`: Retrieve the sets information
- `sets(TidySet) <- value`: Modify the sets information
- `nSets(TidySet)`: Return the number of sets

### See Also

[nSets\(\)](#)

Other slots: [elements\(\)](#), [relations\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [subtract\(\)](#), [union\(\)](#)

### Examples

```
TS <- tidySet(list(A = letters[1:5], B = letters[2:10]))
sets(TS)
sets(TS) <- data.frame(sets = c("B", "A"))
TS2 <- replace_sets(TS, data.frame(sets = c("A", "B", "C")))
sets(TS2)
nSets(TS)
nSets(TS2)
```

---

set_size	<i>Calculates the size of a set</i>
----------	-------------------------------------

---

### Description

Assuming that the fuzzy values are probabilities, calculates the probability of being of different sizes for a given set.

### Usage

```
set_size(object, sets = NULL)

## S4 method for signature 'TidySet'
set_size(object, sets = NULL)
```

### Arguments

object	A TidySet object.
sets	The sets from which the length is calculated.

### Value

A list with the size of the set or the probability of having that size.

### Methods (by class)

- `set_size(TidySet)`: Calculates the size of a set using [length\\_set\(\)](#)

### See Also

[cardinality](#)

Other sizes: [element\\_size\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

### Examples

```
relations <- data.frame(
  sets = c(rep("A", 5), "B", "C"),
  elements = c(letters[seq_len(6)], letters[6]),
  fuzzy = runif(7)
)
```

```
a <- tidySet(relations)
set_size(a)
```

---

set_symbols	<i>A subset of symbols related to sets</i>
-------------	--

---

### Description

Name and symbol of operations related to sets, including intersection and union among others:

### Usage

```
set_symbols
```

### Format

An object of class character of length 16.

### References

<https://www.fileformat.info/info/unicode/category/Sm/list.htm>

### Examples

```
set_symbols
```

---

show, TidySet-method	<i>Method to show the TidySet object</i>
----------------------	--

---

### Description

Prints the resulting table of a TidySet object. Does not shown elements or sets without any relationship (empty sets). To see them use `sets()` or `elements()`.

### Usage

```
## S4 method for signature 'TidySet'
show(object)
```

### Arguments

object	A TidySet
--------	-----------

### Value

A table with the information of the relationships.

---

size	<i>Size</i>
------	-------------

---

### Description

Calculate the size of the elements or sets, using the fuzzy values as probabilities. First it must have active either sets or elements.

### Usage

```
size(object, ...)
```

### Arguments

object	A TidySet object
...	Character vector with the name of elements or sets you want to calculate the size of.

### Value

The size of the elements or sets. If there is no active slot or it is the relations slot returns the TidySet object with a warning.

### See Also

A related concept [cardinality\(\)](#). It is calculated using [length\\_set\(\)](#).

### Examples

```
rel <- data.frame(
  sets = c(rep("A", 5), "B", "C"),
  elements = c(letters[seq_len(6)], letters[6])
)
TS <- tidySet(rel)
TS <- activate(TS, "elements")
size(TS)
TS <- activate(TS, "sets")
size(TS)
# With fuzzy sets
relations <- data.frame(
  sets = c(rep("A", 5), "B", "C"),
  elements = c(letters[seq_len(6)], letters[6]),
  fuzzy = runif(7)
)
TS <- tidySet(relations)
TS <- activate(TS, "elements")
size(TS)
TS <- activate(TS, "sets")
size(TS)
```

---

 subtract
 

---

*Subtract***Description**

Elements in a set not present in the other set. Equivalent to `setdiff()`.

**Usage**

```
subtract(object, set_in, not_in, ...)
```

```
## S4 method for signature 'TidySet,characterORfactor,characterORfactor'
subtract(
  object,
  set_in,
  not_in,
  name = NULL,
  keep = TRUE,
  keep_relations = keep,
  keep_elements = keep,
  keep_sets = keep
)
```

**Arguments**

<code>object</code>	A TidySet object.
<code>set_in</code>	Name of the sets where the elements should be present.
<code>not_in</code>	Name of the sets where the elements should not be present.
<code>...</code>	Placeholder for other arguments that could be passed to the method. Currently not used.
<code>name</code>	Name of the new set. By default it adds a "C".
<code>keep</code>	Logical value to keep all the other sets.
<code>keep_relations</code>	A logical value if you want to keep old relations.
<code>keep_elements</code>	A logical value if you want to keep old elements.
<code>keep_sets</code>	A logical value if you want to keep old sets.

**Value**

A TidySet object.

**Methods (by class)**

- `subtract(object = TidySet, set_in = characterORfactor, not_in = characterORfactor)`: Elements present in sets but not in other sets



**See Also**

[setdiff\(\)](#)

Other complements: [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#)

Other methods that create new sets: [complement\\_element\(\)](#), [complement\\_set\(\)](#), [intersection\(\)](#), [union\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [union\(\)](#)

**Examples**

```
relations <- data.frame(
  sets = c("A", "A", "B", "B", "C", "C"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
TS <- tidySet(relations)
subtract(TS, "A", "B")
subtract(TS, "A", "B", keep = FALSE)
```

---

tidy

---

*Convert GSEABase classes to a TidySet*


---

**Description**

Convert GSEABase classes to a TidySet

**Usage**

```
tidy(object)

## S3 method for class 'GeneSetCollection'
tidy(object)

## S3 method for class 'GeneSet'
tidy(object)
```

**Arguments**

object            A GeneSetCollection or a GeneSet derived object

**Value**

A TidySet object.

**Methods (by class)**

- tidy(GeneSetCollection): Converts to a tidySet given a GeneSetCollection
- tidy(GeneSet): Converts to a tidySet given a GeneSet

**Examples**

```
# Needs GSEABase package from Bioconductor
if (requireNamespace("GSEABase", quietly = TRUE)) {
  library("GSEABase")
  gs <- GeneSet()
  gs
  tidy(gs)
  f1 <- system.file("extdata", "Broad.xml", package="GSEABase")
  gs2 <- getBroadSets(f1) # actually, a list of two gene sets
  TS <- tidy(gs2)
  dim(TS)
  sets(TS)
}
```

---

tidySet

*Create a TidySet object*


---

**Description**

These functions help to create a TidySet object from data.frame, list, matrix, and GO3AnnDbBimap. They can create both fuzzy and standard sets.

**Usage**

```
tidySet(relations)

## S3 method for class 'data.frame'
tidySet(relations)

## S3 method for class 'list'
tidySet(relations)

## S3 method for class 'matrix'
tidySet(relations)

## S3 method for class 'Go3AnnDbBimap'
tidySet(relations)

## S3 method for class 'TidySet'
tidySet(relations)
```

**Arguments**

relations      An object to be coerced to a TidySet.

**Details**

Elements or sets without any relation are not shown when printed.

**Value**

A TidySet object.

**Methods (by class)**

- tidySet(data.frame): Given the relations in a data.frame
- tidySet(list): Convert to a TidySet from a list.
- tidySet(matrix): Convert an incidence matrix into a TidySet
- tidySet(Go3AnnDbBimap): Convert Go3AnnDbBimap into a TidySet object.
- tidySet(TidySet): Convert TidySet into a TidySet object.

**See Also**

[TidySet](#)

**Examples**

```
relations <- data.frame(
  sets = c(rep("a", 5), "b"),
  elements = letters[seq_len(6)]
)
tidySet(relations)
relations2 <- data.frame(
  sets = c(rep("A", 5), "B"),
  elements = letters[seq_len(6)],
  fuzzy = runif(6)
)
tidySet(relations2)
# A
x <- list("A" = letters[1:5], "B" = LETTERS[3:7])
tidySet(x)
# A fuzzy set taken encoded as a list
A <- runif(5)
names(A) <- letters[1:5]
B <- runif(5)
names(B) <- letters[3:7]
relations <- list(A, B)
tidySet(relations)
# Will error
# x <- list("A" = letters[1:5], "B" = LETTERS[3:7], "c" = runif(5))
# a <- tidySet(x) # Only characters or factors are allowed as elements.
M <- matrix(c(1, 0.5, 1, 0), ncol = 2,
```

```

      dimnames = list(c("A", "B"), c("a", "b")))
tidySet(M)

```

---

TidySet-class	<i>A tidy class to represent a set</i>
---------------	--

---

## Description

A set is a group of unique elements it can be either a fuzzy set, where the relationship is between 0 or 1 or nominal.

## Details

When printed if an element or a set do not have any relationship is not shown. They can be created from lists, matrices or data.frames. Check [tidySet\(\)](#) constructor for more information.

## Slots

`relations` A data.frame with elements and the sets were they belong.

`elements` A data.frame of unique elements and related information.

`sets` A data.frame of unique sets and related information.

## See Also

### [tidySet](#)

Other methods: [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is.nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#), [union\(\)](#)

## Examples

```

x <- list("A" = letters[1:5], "B" = LETTERS[3:7])
a <- tidySet(x)
a
x <- list("A" = letters[1:5], "B" = character())
b <- tidySet(x)
b
name_sets(b)

```

---

union	<i>Join sets</i>
-------	------------------

---

### Description

Given a TidySet merges several sets into the new one using the logic defined on FUN.

### Usage

```
union(object, ...)

## S3 method for class 'TidySet'
union(
  object,
  sets,
  name = NULL,
  FUN = "max",
  keep = FALSE,
  keep_relations = keep,
  keep_elements = keep,
  keep_sets = keep,
  ...
)
```

### Arguments

object	A TidySet object.
...	Other named arguments passed to FUN.
sets	The name of the sets to be used.
name	The name of the new set. By defaults joins the sets with an $\cap$ .
FUN	A function to be applied when performing the union. The standard union is the "max" function, but you can provide any other function that given a numeric vector returns a single number.
keep	A logical value if you want to keep.
keep_relations	A logical value if you wan to keep old relations.
keep_elements	A logical value if you wan to keep old elements.
keep_sets	A logical value if you wan to keep old sets.

### Details

The default uses the max function following the [standard fuzzy definition](#), but it can be changed. See examples below.

### Value

A TidySet object.

**See Also**

[union\\_probability\(\)](#)

Other methods that create new sets: [complement\\_element\(\)](#), [complement\\_set\(\)](#), [intersection\(\)](#), [subtract\(\)](#)

Other methods: [TidySet-class](#), [activate\(\)](#), [add\\_column\(\)](#), [add\\_relation\(\)](#), [arrange.TidySet\(\)](#), [cartesian\(\)](#), [complement\\_element\(\)](#), [complement\\_set\(\)](#), [complement\(\)](#), [element\\_size\(\)](#), [elements\(\)](#), [filter.TidySet\(\)](#), [group\\_by.TidySet\(\)](#), [group\(\)](#), [incidence\(\)](#), [intersection\(\)](#), [is.fuzzy\(\)](#), [is\\_nested\(\)](#), [move\\_to\(\)](#), [mutate.TidySet\(\)](#), [nElements\(\)](#), [nRelations\(\)](#), [nSets\(\)](#), [name\\_elements<-\(\)](#), [name\\_sets<-\(\)](#), [name\\_sets\(\)](#), [power\\_set\(\)](#), [pull.TidySet\(\)](#), [relations\(\)](#), [remove\\_column\(\)](#), [remove\\_element\(\)](#), [remove\\_relation\(\)](#), [remove\\_set\(\)](#), [rename\\_elements\(\)](#), [rename\\_set\(\)](#), [select.TidySet\(\)](#), [set\\_size\(\)](#), [sets\(\)](#), [subtract\(\)](#)

**Examples**

```
# Classical set
rel <- data.frame(
  sets = c(rep("A", 5), "B", "B"),
  elements = c(letters[seq_len(6)], "a")
)
TS <- tidySet(rel)
union(TS, c("B", "A"))
# Fuzzy set
rel <- data.frame(
  sets = c(rep("A", 5), "B", "B"),
  elements = c(letters[seq_len(6)], "a"),
  fuzzy = runif(7)
)
TS2 <- tidySet(rel)
# Standard default logic
union(TS2, c("B", "A"), "C")
# Probability logic
union(TS2, c("B", "A"), "C", FUN = union_probability)
```

---

union\_closed

*Union closed sets*

---

**Description**

Tests if a given object is union-closed.

**Usage**

```
union_closed(object, ...)
```

```
## S3 method for class 'TidySet'
union_closed(object, sets = NULL, ...)
```

**Arguments**

object	A TidySet object.
...	Other named arguments passed to FUN.
sets	The name of the sets to be used.

**Value**

A logical value: TRUE if the combinations of sets produce already existing sets, FALSE otherwise.

**Examples**

```
l <- list(A = "1",
         B = c("1", "2"),
         C = c("2", "3", "4"),
         D = c("1", "2", "3", "4"))
)
TS <- tidySet(l)
union_closed(TS)
union_closed(TS, sets = c("A", "B", "C"))
union_closed(TS, sets = c("A", "B", "C", "D"))
```

---

union\_probability      *Calculates the probability of a single length*

---

**Description**

Creates all the possibilities and then add them up. union\_probability Assumes independence between the probabilities to calculate the final size.

**Usage**

```
union_probability(p)

length_probability(p, size)
```

**Arguments**

p	Numeric vector of probabilities.
size	Integer value of the size of the selected values.

**Value**

A numeric value of the probability of the given size.

**See Also**

[multiply\\_probabilities\(\)](#) and [length\\_set\(\)](#)

**Examples**

```
length_probability(c(0.5, 0.75), 2)
length_probability(c(0.5, 0.75, 0.66), 1)
length_probability(c(0.5, 0.1, 0.3, 0.5, 0.25, 0.23), 2)
union_probability(c(0.5, 0.1, 0.3))
```



# Index

- \* **IO functions**
  - getGAF, 26
  - getGMT, 27
  - getOBO, 28
- \* **add functions**
  - add\_relation, 7
- \* **add\_\***
  - add\_elements, 6
  - add\_relations, 8
  - add\_sets, 9
- \* **column**
  - add\_column, 5
  - remove\_column, 52
- \* **complements**
  - complement, 16
  - complement\_element, 17
  - complement\_set, 18
  - subtract, 64
- \* **count functions**
  - nElements, 46
  - nRelations, 47
  - nSets, 48
- \* **datasets**
  - set\_symbols, 62
- \* **methods that create new sets**
  - complement\_element, 17
  - complement\_set, 18
  - intersection, 32
  - subtract, 64
  - union, 69
- \* **methods**
  - activate, 3
  - add\_column, 5
  - add\_relation, 7
  - arrange.TidySet, 11
  - cartesian, 14
  - complement, 16
  - complement\_element, 17
  - complement\_set, 18
  - element\_size, 22
  - elements, 21
  - filter.TidySet, 25
  - group, 28
  - group\_by.TidySet, 30
  - incidence, 31
  - intersection, 32
  - is.fuzzy, 34
  - is\_nested, 35
  - move\_to, 38
  - mutate.TidySet, 40
  - name\_elements<-, 42
  - name\_sets, 43
  - name\_sets<-, 44
  - nElements, 46
  - nRelations, 47
  - nSets, 48
  - power\_set, 48
  - pull.TidySet, 49
  - relations, 50
  - remove\_column, 52
  - remove\_element, 53
  - remove\_relation, 54
  - remove\_set, 55
  - rename\_elements, 56
  - rename\_set, 57
  - select.TidySet, 58
  - set\_size, 61
  - sets, 59
  - subtract, 64
  - TidySet-class, 68
  - union, 69
- \* **names**
  - name\_elements, 41
  - name\_elements<-, 42
  - name\_sets, 43
  - name\_sets<-, 44
  - rename\_elements, 56
  - rename\_set, 57

- \* **remove functions**
  - remove\_element, 53
  - remove\_relation, 54
  - remove\_set, 55
- \* **renames**
  - rename\_elements, 56
  - rename\_set, 57
- \* **sizes**
  - element\_size, 22
  - set\_size, 61
- \* **slots**
  - elements, 21
  - relations, 50
  - sets, 59
- [, TidySet-method (extract-TidySet), 23
- [<-, TidySet-method (extract-TidySet), 23
- [[, TidySet-method (extract-TidySet), 23
- [[<-, TidySet-method (extract-TidySet), 23
- \$, TidySet-method (extract-TidySet), 23
- \$<-, TidySet-method (extract-TidySet), 23
- activate, 3, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- activate(), 11, 16, 25, 30, 40, 50, 59
- active(activate), 3
- add\_column, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- add\_column, TidySet, character-method (add\_column), 5
- add\_elements, 6, 8, 9
- add\_relation, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- add\_relation(), 8
- add\_relation, TidySet, data.frame-method (add\_relation), 7
- add\_relations, 6, 8, 9
- add\_sets, 6, 8, 9
- adjacency, 10
- adjacency(), 31
- adjacency\_element (adjacency), 10
- adjacency\_set (adjacency), 10
- arrange.TidySet, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- arrange\_element (arrange.TidySet), 11
- arrange\_relation (arrange.TidySet), 11
- arrange\_set (arrange.TidySet), 11
- as.data.frame.TidySet, 12
- as.list.TidySet, 12
- c, TidySet-method, 13
- cardinality, 14
- cardinality(), 63
- cardinality, TidySet-method (cardinality), 14
- cartesian, 4, 5, 7, 11, 14, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- complement, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- complement\_element, 4, 5, 7, 11, 15, 16, 17, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- complement\_element(), 16
- complement\_element, TidySet, characterORfactor-method (complement\_element), 17
- complement\_set, 4, 5, 7, 11, 15, 16, 18, 18, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- complement\_set(), 16
- complement\_set, TidySet, characterORfactor-method (complement\_set), 18
- deactivate (activate), 3
- dim(), 36
- dplyr::arrange(), 11
- dplyr::filter(), 25
- dplyr::group\_by(), 30
- dplyr::mutate(), 40
- dplyr::pull(), 50
- dplyr::select(), 59
- droplevels(), 24
- droplevels.TidySet, 20
- element\_size, 4, 5, 7, 11, 15, 16, 18, 19, 21, 22, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- element\_size, TidySet-method (element\_size), 22
- elements, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- elements(), 62

- elements, TidySet-method (elements), 21
- elements<- (elements), 21
- elements<- , TidySet-method (elements), 21
- extract-TidySet, 23
- filter(), 19
- filter.TidySet, 4, 5, 7, 11, 15, 16, 18, 19, 21, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- filter\_element (filter.TidySet), 25
- filter\_relation (filter.TidySet), 25
- filter\_set (filter.TidySet), 25
- getGAF, 26, 27, 28
- getGMT, 26, 27, 28
- getOBO, 26, 27, 28
- group, 4, 5, 7, 11, 15, 16, 18, 20, 21, 23, 25, 28, 30, 31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- group\_by.TidySet, 4, 5, 7, 11, 15, 16, 18, 20, 21, 23, 25, 29, 30, 31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- incidence, 4, 5, 7, 11, 15, 16, 18, 20, 21, 23, 25, 29, 30, 31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- incidence(), 10
- incidence, TidySet-method (incidence), 31
- independent, 32
- independent\_probabilities (multiply\_probabilities), 39
- intersect (intersection), 32
- intersection, 4, 5, 7, 11, 15, 16, 18–21, 23, 25, 29–31, 32, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- intersection, TidySet, character-method (intersection), 32
- is.fuzzy, 4, 5, 7, 11, 15, 16, 18, 20, 21, 23, 25, 29–31, 33, 34, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- is.fuzzy, TidySet-method (is.fuzzy), 34
- is\_nested, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- length(), 37
- length.TidySet, 36
- length\_probability (union\_probability), 71
- length\_probability(), 38, 39
- length\_set, 37
- length\_set(), 22, 61, 63, 71
- lengths(), 36
- lengths, TidySet-method, 37
- move\_to, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 38, 40, 42, 44–54, 56–61, 65, 68, 70
- move\_to, TidySet, characterORfactor, characterORfactor, characterORfactor, (move\_to), 38
- multiply\_probabilities, 39
- multiply\_probabilities(), 71
- mutate, 6, 8, 9
- mutate.TidySet, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–54, 56–61, 65, 68, 70
- mutate\_element, 6
- mutate\_element (mutate.TidySet), 40
- mutate\_relation, 8
- mutate\_relation (mutate.TidySet), 40
- mutate\_set, 9
- mutate\_set (mutate.TidySet), 40
- name\_elements, 41, 42, 44, 45, 57, 58
- name\_elements(), 57
- name\_elements, TidySet-method (name\_elements), 41
- name\_elements<- , 42
- name\_elements<- , TidySet, characterORfactor-method (name\_elements), 41
- name\_sets, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39–42, 43, 45–53, 55–61, 65, 68, 70
- name\_sets(), 58
- name\_sets, TidySet-method (name\_sets), 43
- name\_sets<- , 44
- name\_sets<- , TidySet, characterORfactor-method (name\_sets), 43
- naming, 45
- ncol(), 36
- nElements, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44, 45, 46, 47–54, 56–61, 65, 68, 70
- nElements(), 21
- nElements, TidySet-method (elements), 21
- nRelations, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–46, 47, 48–54, 56–61, 65, 68, 70

- nRelations(), 51
- nRelations, TidySet-method (relations), 50
- nrow(), 36
- nSets, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–47, 48, 49–54, 56–61, 65, 68, 70
- nSets(), 60
- nSets, TidySet-method (sets), 59
- power\_set, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–48, 48, 50–53, 55–61, 65, 68, 70
- pull.TidySet, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–49, 49, 51–53, 55–61, 65, 68, 70
- pull\_element (pull.TidySet), 49
- pull\_relation (pull.TidySet), 49
- pull\_set (pull.TidySet), 49
- relations, 4, 5, 7, 11, 15, 16, 18, 20–23, 25, 29–31, 33, 35, 39, 40, 42, 44–50, 50, 52, 53, 55–61, 65, 68, 70
- relations, TidySet-method (relations), 50
- relations<- (relations), 50
- relations<- , TidySet-method (relations), 50
- remove\_column, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–51, 52, 53, 55–61, 65, 68, 70
- remove\_column, TidySet, character, character-method (remove\_column), 52
- remove\_element, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–52, 53, 54–61, 65, 68, 70
- remove\_element, TidySet, characterORfactor-method (remove\_element), 53
- remove\_relation, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–53, 54, 56–61, 65, 68, 70
- remove\_relation, TidySet, characterORfactor, characterORfactor-method (remove\_relation), 54
- remove\_set, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–55, 55, 57–61, 65, 68, 70
- remove\_set, TidySet, characterORfactor-method (remove\_set), 55
- rename\_elements, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39–42, 44–53, 55, 56, 56, 58–61, 65, 68, 70
- rename\_elements(), 42
- rename\_elements, TidySet-method (rename\_elements), 56
- rename\_set, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39–42, 44–53, 55–57, 57, 59–61, 65, 68, 70
- rename\_set(), 5, 45, 52
- rename\_set, TidySet-method (rename\_set), 57
- replace\_elements (elements), 21
- replace\_relations (relations), 50
- replace\_sets (sets), 59
- select.TidySet, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–53, 55–58, 58, 60, 61, 65, 68, 70
- select\_element (select.TidySet), 58
- select\_relation (select.TidySet), 58
- select\_set (select.TidySet), 58
- set\_size, 4, 5, 7, 11, 15, 16, 18, 20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–53, 55–60, 61, 65, 68, 70
- set\_size(), 37
- set\_size, TidySet-method (set\_size), 61
- set\_symbols, 62
- set\_symbols(), 46
- setdiff(), 64, 65
- sets, 4, 5, 7, 11, 15, 16, 18, 20–23, 25, 29–31, 33, 35, 39, 40, 42, 44–53, 55–59, 59, 61, 65, 68, 70
- sets(), 62
- sets, TidySet-method (sets), 59
- sets<- (sets), 59
- sets<- , TidySet-method (sets), 59
- show, TidySet-method, 62
- size, 63
- size(), 14
- subtract, 4, 5, 7, 11, 15, 16, 18–20, 22, 23, 25, 29–31, 33, 35, 39, 40, 42, 44–53, 54, 56–61, 65, 68, 70
- subtract, TidySet, characterORfactor, characterORfactor-method (subtract), 64
- tidy, 65
- TidySet, 6, 8, 9, 24, 32, 67
- TidySet (TidySet-class), 68
- tidySet, 66, 68
- TidySet(), 50

`tidySet()`, [31](#), [68](#)

TidySet-class, [68](#)

`union`, [4](#), [5](#), [7](#), [11](#), [15](#), [16](#), [18–20](#), [22](#), [23](#), [25](#),  
[29–31](#), [33](#), [35](#), [39](#), [40](#), [42](#), [44–53](#),  
[55–61](#), [65](#), [68](#), [69](#)

`union_closed`, [70](#)

`union_probability`, [71](#)

`union_probability()`, [70](#)