

# Package ‘nlist’

August 4, 2021

**Title** Lists of Numeric Atomic Objects

**Version** 0.3.2

**Description** Create and manipulate numeric list ('nlist') objects. An 'nlist' is an S3 list of uniquely named numeric objects. An numeric object is an integer or double vector, matrix or array. An 'nlists' object is a S3 class list of 'nlist' objects with the same names, dimensionalities and typeofs. Numeric list objects are of interest because they are the raw data inputs for analytic engines such as 'JAGS', 'STAN' and 'TMB'. Numeric lists objects, which are useful for storing multiple realizations of of simulated data sets, can be converted to coda::mcmc and coda::mcmc.list objects.

**License** MIT + file LICENSE

**URL** <https://github.com/poissonconsulting/nlist>

**BugReports** <https://github.com/poissonconsulting/nlist/issues>

**Depends** R (>= 3.4)

**Imports** abind,

chk,  
coda,  
extras,  
generics,  
lifecycle,  
purrr,  
stats,  
term,  
tibble,  
universals

**Suggests** covr,

rlang,  
testthat

**RdMacros** lifecycle

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1

**R topics documented:**

aggregate.nlist . . . . .	3
aggregate.nlists . . . . .	4
as_mcmc . . . . .	4
as_mcmc_list . . . . .	5
as_nlist . . . . .	6
as_nlists . . . . .	8
as_term.mcmc . . . . .	9
as_term.nlist . . . . .	9
as_term.nlists . . . . .	10
as_term_frame . . . . .	10
as_term_frame.nlist . . . . .	11
as_term_frame.nlists . . . . .	11
bind_iterations.mcmc . . . . .	12
bind_iterations.mcmc.list . . . . .	13
chk_nlist . . . . .	13
collapse_chains.mcmc . . . . .	14
collapse_chains.mcmc.list . . . . .	15
collapse_chains.nlist . . . . .	15
collapse_chains.nlists . . . . .	16
complete_terms.mcmc . . . . .	17
estimates.nlist . . . . .	17
estimates.nlists . . . . .	18
fill_all.nlist . . . . .	19
fill_all.nlists . . . . .	20
fill_na.nlist . . . . .	21
fill_na.nlists . . . . .	22
is_numeric . . . . .	23
nchains.mcmc . . . . .	24
nchains.mcmc.list . . . . .	24
nchains.nlist . . . . .	25
nchains.nlists . . . . .	25
niters.mcmc . . . . .	26
niters.mcmc.list . . . . .	27
niters.nlist . . . . .	27
niters.nlists . . . . .	28
nlist . . . . .	28
nlists . . . . .	29
npdims.mcmc.list . . . . .	30
npdims.nlist . . . . .	30
npdims.nlists . . . . .	31
nsims.nlist . . . . .	32
nsims.nlists . . . . .	32
nterms.mcmc . . . . .	33
nterms.mcmc.list . . . . .	34
nterms.nlist . . . . .	34
nterms.nlists . . . . .	35
pars.mcmc . . . . .	35
pars.mcmc.list . . . . .	36
pars.nlist . . . . .	37
pars.nlists . . . . .	37

pdims.mcmc . . . . .	38
pdims.mcmc.list . . . . .	38
pdims.nlist . . . . .	39
pdims.nlists . . . . .	40
relist_nlist . . . . .	40
set_pars.mcmc . . . . .	41
set_pars.mcmc.list . . . . .	42
set_pars.nlist . . . . .	42
set_pars.nlists . . . . .	43
split_chains.nlists . . . . .	44
subset.mcmc . . . . .	44
subset.mcmc.list . . . . .	45
subset.nlist . . . . .	46
subset.nlists . . . . .	47
thin.default . . . . .	48
tidy.mcmc . . . . .	48
tidy.mcmc.list . . . . .	49
tidy.nlists . . . . .	49
unlist.nlist . . . . .	50
unlist_nlist . . . . .	51
vld_nlist . . . . .	51

<b>Index</b>	<b>53</b>
--------------	-----------

---

aggregate.nlist	<i>Aggregate nlist</i>
-----------------	------------------------

---

## Description

Aggregates an [nlist\\_object\(\)](#) into a named list of numeric scalars.

## Usage

```
## S3 method for class 'nlist'
aggregate(x, fun = mean, ...)
```

## Arguments

x	An nlist object.
fun	A function that given a numeric vector returns a numeric scalar.
...	Additional arguments passed to fun.

## Value

An named list of numeric scalars

## See Also

Other aggregate: [aggregate.nlists\(\)](#)

## Examples

```
aggregate(nlist(x = 1:9))
aggregate(nlist(y = 3:5, zz = matrix(1:9, 3)), fun = function(x) x[1])
```

---

aggregate.nlists	<i>Aggregate nlists</i>
------------------	-------------------------

---

## Description

Aggregates an [nlists\\_object\(\)](#) into a [nlist\\_object\(\)](#) or `by_chain = TRUE` an [nlists\\_object\(\)](#) with `nchains` [nlist\\_object\(\)](#)s.

## Usage

```
## S3 method for class 'nlists'
aggregate(x, fun = mean, ..., by_chain = FALSE)
```

## Arguments

<code>x</code>	An object.
<code>fun</code>	A function that given a numeric vector returns a numeric scalar.
<code>...</code>	Unused.
<code>by_chain</code>	A flag specifying whether to aggregate by chains.

## Value

An nlist object if `by_chain = FALSE` otherwise an nlists object.

## See Also

Other aggregate: [aggregate.nlist\(\)](#)

## Examples

```
aggregate(nlists(nlist(x = 1:3), nlist(x = 2:4)))
```

---

as_mcmc	<i>Coerce to mcmc Object</i>
---------	------------------------------

---

## Description

Coerce an R object to an mcmc object.

**Usage**

```
as_mcmc(x, ...)
```

```
## S3 method for class 'mcmc.list'
```

```
as_mcmc(x, ...)
```

```
## S3 method for class 'nlist'
```

```
as_mcmc(x, ...)
```

```
## S3 method for class 'nlists'
```

```
as_mcmc(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**Value**

An mcmc object.

**Methods (by class)**

- `mcmc.list`: Coerce an `mcmc.list` object to an mcmc object.
- `nlist`: Coerce an `nlist` object to an mcmc object.
- `nlists`: Coerce an `nlists` object to an mcmc object.

**See Also**

[coda::as.mcmc\(\)](#)

Other mcmc: [as\\_mcmc\\_list\(\)](#)

**Examples**

```
as_mcmc(as_mcmc_list(nlists(nlist(x = 2), nlist(x = 3))))
```

```
as_mcmc(nlist(x = matrix(1:6, 2)))
```

```
as_mcmc(nlists(
```

```
  nlist(x = matrix(1:6, 2)),
```

```
  nlist(x = matrix(3:8, 2))
```

```
))
```

---

as\_mcmc\_list

*Coerce to an mcmc.list Object*


---

**Description**

Coerce an R object to an `mcmc.list` object.

**Usage**

```
as_mcmc_list(x, ...)

## S3 method for class 'mcmc'
as_mcmc_list(x, ...)

## S3 method for class 'nlist'
as_mcmc_list(x, ...)

## S3 method for class 'nlists'
as_mcmc_list(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**Value**

An mcmc.list object.

**Methods (by class)**

- mcmc: Coerce an mcmc object to an mcmc.list object.
- nlist: Coerce an nlist object to an mcmc.list object.
- nlists: Coerce an nlists object to an mcmc.list object.

**See Also**

Other mcmc: [as\\_mcmc\(\)](#)

**Examples**

```
as_mcmc_list(nlist(x = matrix(1:6, 2)))

as_mcmc_list(nlists(
  nlist(x = matrix(1:6, 2)),
  nlist(x = matrix(3:8, 2))
))
```

---

as\_nlist

*Coerce to nlist*


---

**Description**

Coerce an R object to an [nlist\\_object\(\)](#).

**Usage**

```
as_nlist(x, ...)  
  
as.nlist(x, ...)  
  
## S3 method for class 'numeric'  
as_nlist(x, ...)  
  
## S3 method for class 'list'  
as_nlist(x, ...)  
  
## S3 method for class 'data.frame'  
as_nlist(x, ...)  
  
## S3 method for class 'mcmc'  
as_nlist(x, ...)  
  
## S3 method for class 'mcmc.list'  
as_nlist(x, ...)  
  
as.nlists(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**Value**

An nlist object.

**Methods (by class)**

- `numeric`: Coerce named numeric vector to nlist
- `list`: Coerce list to nlist
- `data.frame`: Coerce data.frame to nlist
- `mcmc`: Coerce mcmc (with one iteration) to nlist
- `mcmc.list`: Coerce mcmc.list (with one iteration) to nlist

**See Also**

Other coerce: [as\\_nlists\(\)](#)

**Examples**

```
as_nlist(list(x = 1:4))  
as_nlist(c(`a[2]` = 3, `a[1]` = 2))
```

---

**as\_nlists***Coerce to nlists*

---

**Description**

Coerce an R object to an `nlists_object()`.

**Usage**

```
as_nlists(x, ...)  
  
## S3 method for class 'list'  
as_nlists(x, ...)  
  
## S3 method for class 'mcmc'  
as_nlists(x, ...)  
  
## S3 method for class 'mcmc.list'  
as_nlists(x, ...)  
  
## S3 method for class 'nlist'  
as_nlists(x, ...)
```

**Arguments**

<code>x</code>	An object.
<code>...</code>	Unused.

**Value**

An nlists object.

**Methods (by class)**

- `list`: Coerce list to nlists
- `mcmc`: Coerce mcmc to nlists
- `mcmc.list`: Coerce mcmc.list to nlists
- `nlist`: Coerce nlist to nlists

**See Also**

Other coerce: `as_nlist()`

**Examples**

```
as_nlists(list(nlist(x = c(1, 5)), nlist(x = c(2, 3)), nlist(x = c(3, 2))))
```



---

as_term.mcmc	<i>Coerce to a Term Vector</i>
--------------	--------------------------------

---

**Description**

Coerce to a Term Vector

**Usage**

```
## S3 method for class 'mcmc'
as_term(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**See Also**

Other coerce term: [as\\_term.nlists\(\)](#), [as\\_term.nlist\(\)](#), [as\\_term\\_frame.nlists\(\)](#), [as\\_term\\_frame.nlist\(\)](#), [as\\_term\\_frame\(\)](#)

**Examples**

```
as_term(as_mcmc(nlist(x = matrix(1:4, ncol = 2))))
```

---

as_term.nlist	<i>Coerce to a Term Vector</i>
---------------	--------------------------------

---

**Description**

Coerce to a Term Vector

**Usage**

```
## S3 method for class 'nlist'
as_term(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**See Also**

Other coerce term: [as\\_term.mcmc\(\)](#), [as\\_term.nlists\(\)](#), [as\\_term\\_frame.nlists\(\)](#), [as\\_term\\_frame.nlist\(\)](#), [as\\_term\\_frame\(\)](#)

**Examples**

```
as_term(nlist(x = matrix(1:4, ncol = 2)))
```

---

as_term.nlists	<i>Coerce to a Term Vector</i>
----------------	--------------------------------

---

**Description**

Coerce to a Term Vector

**Usage**

```
## S3 method for class 'nlists'
as_term(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**See Also**

Other coerce term: [as\\_term.mcmc\(\)](#), [as\\_term.nlist\(\)](#), [as\\_term\\_frame.nlists\(\)](#), [as\\_term\\_frame.nlist\(\)](#), [as\\_term\\_frame\(\)](#)

**Examples**

```
as_term(nlists(nlist(x = matrix(1:4, ncol = 2))))
```

---

as_term_frame	<i>Coerce to a Term Frame</i>
---------------	-------------------------------

---

**Description**

A term frame is a tibble with the first column a term vector called `term` and a numeric column called `value` and in the case of an `nlists` object an integer vector called `samples`. It includes the original `nlist` or `nlists` object.

**Usage**

```
as_term_frame(x, ...)
```

**Arguments**

x	An object.
...	Unused.

**Value**

An `term_frame` object.

**See Also**

Other coerce term: [as\\_term.mcmc\(\)](#), [as\\_term.nlists\(\)](#), [as\\_term.nlist\(\)](#), [as\\_term\\_frame.nlists\(\)](#), [as\\_term\\_frame.nlist\(\)](#)

---

as\_term\_frame.nlist     *Coerce nlist Object to Data Frame*

---

### Description

Coerces an nlist object to a data.frame with an term column and a value column.

### Usage

```
## S3 method for class 'nlist'  
as_term_frame(x, ...)
```

### Arguments

x	An nlist object.
...	Unused.

### Value

A data.frame.

### See Also

Other coerce term: [as\\_term.mcmc\(\)](#), [as\\_term.nlists\(\)](#), [as\\_term.nlist\(\)](#), [as\\_term\\_frame.nlists\(\)](#), [as\\_term\\_frame\(\)](#)

### Examples

```
as_term_frame(nlist(x = 1, y = 4:6))
```

---

as\_term\_frame.nlists     *Coerce nlists Object to Data Frame*

---

### Description

Coerces an nlists object to a data.frame with a term, sample and value column.

### Usage

```
## S3 method for class 'nlists'  
as_term_frame(x, ...)
```

### Arguments

x	An nlists object.
...	Unused.

### Value

A data.frame.

**See Also**

Other coerce term: [as\\_term.mcmc\(\)](#), [as\\_term.nlists\(\)](#), [as\\_term.nlist\(\)](#), [as\\_term\\_frame.nlist\(\)](#), [as\\_term\\_frame\(\)](#)

**Examples**

```
as_term_frame(nlists(
  nlist(x = 1, y = 4:6),
  nlist(x = 3, y = 1:3)
))
```

---

bind\_iterations.mcmc    *Bind Iterations*

---

**Description**

Combines two MCMC objects (with the same parameters and chains) by iterations.

**Usage**

```
## S3 method for class 'mcmc'
bind_iterations(x, x2, ...)
```

**Arguments**

x	An object.
x2	A second object.
...	Other arguments passed to methods.

**Value**

The combined object.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [collapse\\_chains\(\)](#), [estimates\(\)](#), [split\\_chains\(\)](#)

**Examples**

```
bind_iterations(as_mcmc(nlist(x = 1)), as_mcmc(nlist(x = 3)))
```

---

```
bind_iterations.mcmc.list
```

*Bind Iterations*

---

**Description**

Combines two MCMC objects (with the same parameters and chains) by iterations.

**Usage**

```
## S3 method for class 'mcmc.list'
bind_iterations(x, x2, ...)
```

**Arguments**

x	An object.
x2	A second object.
...	Other arguments passed to methods.

**Value**

The combined object.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [collapse\\_chains\(\)](#), [estimates\(\)](#), [split\\_chains\(\)](#)

**Examples**

```
bind_iterations(as_mcmc_list(nlist(x = 1)), as_mcmc_list(nlist(x = 3)))
```

---

```
chk_nlist
```

*Check nlist Object or nlists Object*

---

**Description**

chk\_nlist checks if an [nlist-object\(\)](#).

**Usage**

```
chk_nlist(x, x_name = NULL)

chk_nlists(x, x_name = NULL)
```

**Arguments**

x	The object to check.
x_name	A string of the name of object x or NULL.

**Value**

NULL, invisibly. Called for the side effect of throwing an error if the condition is not met.

**Functions**

- `chk_nlists`: Check nlists Object  
`chk_nlists` checks if an [nlists-object\(\)](#).

**Examples**

```
# chk_nlist
chk_nlist(nlist(x = 1))
try(chk_nlist(list(x = 1)))

# chk_nlists
chk_nlists(nlists(nlist(x = 1)))
```

---

`collapse_chains.mcmc`    *Collapse Chains*

---

**Description**

Collapses an MCMC object's chains into a single chain.

**Usage**

```
## S3 method for class 'mcmc'
collapse_chains(x, ...)
```

**Arguments**

<code>x</code>	An object.
<code>...</code>	Other arguments passed to methods.

**Details**

As mcmc objects can only have 1 chain the object is unchanged.

**Value**

The modified object with one chain.

**See Also**

Other collapse: [collapse\\_chains.nlists\(\)](#), [collapse\\_chains.nlist\(\)](#)

**Examples**

```
collapse_chains(as_mcmc(nlist(x = 2)))
```

---

```
collapse_chains.mcmc.list
```

*Collapse Chains*

---

**Description**

Collapses an MCMC object's chains into a single chain.

**Usage**

```
## S3 method for class 'mcmc.list'
collapse_chains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

The modified object with one chain.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [bind\\_iterations\(\)](#), [estimates\(\)](#), [split\\_chains\(\)](#)

---

```
collapse_chains.nlist
```

*Collapse Chains*

---

**Description**

Collapses an MCMC object's chains into a single chain.

**Usage**

```
## S3 method for class 'nlist'
collapse_chains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Details**

As nlist objects can only have 1 chain the object is unchanged.

**Value**

The modified object with one chain.

**See Also**

Other collapse: [collapse\\_chains.mcmc\(\)](#), [collapse\\_chains.nlists\(\)](#)

**Examples**

```
collapse_chains(nlist(x = 2))
```

---

`collapse_chains.nlists`*Collapse Chains*

---

**Description**

Collapses an MCMC object's chains into a single chain.

**Usage**

```
## S3 method for class 'nlists'  
collapse_chains(x, ...)
```

**Arguments**

<code>x</code>	An object.
<code>...</code>	Other arguments passed to methods.

**Value**

The modified object with one chain.

**See Also**

Other collapse: [collapse\\_chains.mcmc\(\)](#), [collapse\\_chains.nlist\(\)](#)

**Examples**

```
collapse_chains(nlist(x = 2))
```



---

complete_terms.mcmc	<i>Complete Terms</i>
---------------------	-----------------------

---

**Description**

Adds any absent elements to an mcmc object.

**Usage**

```
## S3 method for class 'mcmc'
complete_terms(x, silent = FALSE, ...)
```

**Arguments**

x	An mcmc object.
silent	A flag specifying whether to suppress warning messages.
...	Unused.

**Details**

The terms are repaired before being completed. Missing or invalid or inconsistent terms are dropped with a warning.

**Value**

The repaired and complete mcmc object.

**Examples**

```
mcmc <- as_mcmc(nlist(beta = matrix(1:4, nrow = 2)))
mcmc <- mcmc[, -4, drop = FALSE]
complete_terms(mcmc)
```

---

estimates.nlist	<i>Estimates</i>
-----------------	------------------

---

**Description**

Calculates the estimates for an MCMC object.

**Usage**

```
## S3 method for class 'nlist'
estimates(x, fun = median, ...)
```

**Arguments**

x	An object.
fun	A function that given a numeric vector returns a numeric scalar.
...	Additional arguments passed to fun.

**Value**

A list of uniquely named numeric objects.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [bind\\_iterations\(\)](#), [collapse\\_chains\(\)](#), [split\\_chains\(\)](#)

**Examples**

```
estimates(nlist(x = 1:9))
estimates(nlist(y = 3:5, zz = matrix(1:9, 3)))
```

---

estimates.nlists	<i>Estimates</i>
------------------	------------------

---

**Description**

Calculates the estimates for an MCMC object.

**Usage**

```
## S3 method for class 'nlists'
estimates(x, fun = median, ...)
```

**Arguments**

x	An object.
fun	A function that given a numeric vector returns a numeric scalar.
...	Additional arguments passed to fun.

**Value**

A list of uniquely named numeric objects.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [bind\\_iterations\(\)](#), [collapse\\_chains\(\)](#), [split\\_chains\(\)](#)

**Examples**

```
estimates(nlists(nlist(x = 1:3), nlist(x = 2:4)), fun = mean)
```

---

fill\_all.nlist*Fill All Values*

---

## Description

Fills all of an object's (missing and non-missing) values while preserving the object's dimensionality and class.

## Usage

```
## S3 method for class 'nlist'  
fill_all(x, value = 0L, nas = TRUE, ...)
```

## Arguments

x	An object.
value	A scalar of the value to replace values with.
nas	A flag specifying whether to also fill missing values.
...	Other arguments passed to methods.

## Details

It should only be defined for objects with values of consistent class ie not standard data.frames.

## Value

The modified object.

## Methods (by class)

- logical: Fill All for logical Objects
- integer: Fill All for integer Objects
- numeric: Fill All for numeric Objects
- character: Fill All for character Objects

## See Also

Other fill: [fill\\_na\(\)](#)

## Examples

```
fill_all(nlist(x = c(2, NA), y = matrix(c(1:3, NA), nrow = 2)))  
fill_all(nlist(x = c(2, NA), y = matrix(c(1:3, NA), nrow = 2)), nas = FALSE)
```

---

fill_all.nlists	<i>Fill All Values</i>
-----------------	------------------------

---

## Description

Fills all of an object's (missing and non-missing) values while preserving the object's dimensionality and class.

## Usage

```
## S3 method for class 'nlists'
fill_all(x, value = 0L, nas = TRUE, ...)
```

## Arguments

x	An object.
value	A scalar of the value to replace values with.
nas	A flag specifying whether to also fill missing values.
...	Other arguments passed to methods.

## Details

It should only be defined for objects with values of consistent class ie not standard data.frames.

## Value

The modified object.

## Methods (by class)

- logical: Fill All for logical Objects
- integer: Fill All for integer Objects
- numeric: Fill All for numeric Objects
- character: Fill All for character Objects

## See Also

Other fill: [fill\\_na\(\)](#)

## Examples

```
fill_all(nlists(nlist(x = c(2, NA)), nlist(x = c(NA_real_, NA))))
fill_all(nlists(nlist(x = c(2, NA)), nlist(x = c(NA_real_, NA))), nas = FALSE)
```

---

fill_na.nlist	<i>Fill Missing Values</i>
---------------	----------------------------

---

## Description

Fills all of an object's missing values while preserving the object's dimensionality and class.

## Usage

```
## S3 method for class 'nlist'
fill_na(x, value = 0L, ...)
```

## Arguments

x	An object.
value	A scalar of the value to replace values with.
...	Other arguments passed to methods.

## Details

It should only be defined for objects with values of consistent class ie not standard data.frames.

## Value

The modified object.

## Methods (by class)

- logical: Fill Missing Values for logical Objects
- integer: Fill Missing Values for integer Objects
- numeric: Fill Missing Values for numeric Objects
- character: Fill Missing Values for character Objects

## See Also

Other fill: [fill\\_all\(\)](#)

## Examples

```
fill_na(nlist(x = c(2, NA), y = matrix(c(1:3, NA), nrow = 2)))
fill_na(nlists(nlist(x = c(2, NA)), nlist(x = c(NA_real_, NA))))
```

---

fill_na.nlists	<i>Fill Missing Values</i>
----------------	----------------------------

---

### Description

Fills all of an object's missing values while preserving the object's dimensionality and class.

### Usage

```
## S3 method for class 'nlists'  
fill_na(x, value = 0L, ...)
```

### Arguments

x	An object.
value	A scalar of the value to replace values with.
...	Other arguments passed to methods.

### Details

It should only be defined for objects with values of consistent class ie not standard data.frames.

### Value

The modified object.

### Methods (by class)

- logical: Fill Missing Values for logical Objects
- integer: Fill Missing Values for integer Objects
- numeric: Fill Missing Values for numeric Objects
- character: Fill Missing Values for character Objects

### See Also

Other fill: [fill\\_all\(\)](#)

### Examples

```
fill_na(nlist(x = c(2, NA), y = matrix(c(1:3, NA), nrow = 2)))
```

---

is_numeric	<i>Is numeric, nlist or nlists</i>
------------	------------------------------------

---

## Description

Ask whether x is a numeric object, `nlist_object()` or `nlists_object()`.

## Usage

```
is_numeric(x)

is_nlist(x)

is_nlists(x)
```

## Arguments

x                      An object.

## Value

A flag indicating whether x is a numeric object or inherits from S3 class nlist or nlists.

## Functions

- `is_nlist`: Is nlist
- `is_nlists`: Is nlists

## Examples

```
# is_numeric
is_numeric(list(x = 1))
is_numeric(1)

# is_nlist
is_nlist(1)
is_nlist(list(x = 1))
is_nlist(nlist(x = 1))

# is_nlists
is_nlists(nlist(x = 1))
is_nlists(nlists(nlist(x = 2), nlist(x = 3.5)))
```

---

nchains.mcmc	<i>Number of Chains</i>
--------------	-------------------------

---

**Description**

Gets the number of chains of an MCMC object.

**Usage**

```
## S3 method for class 'mcmc'
nchains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

An integer scalar of the number of chains.

**See Also**

Other MCMC dimensions: [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)

---

nchains.mcmc.list	<i>Number of Chains</i>
-------------------	-------------------------

---

**Description**

Gets the number of chains of an MCMC object.

**Usage**

```
## S3 method for class 'mcmc.list'
nchains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

An integer scalar of the number of chains.

**See Also**

Other MCMC dimensions: [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)



---

nchains.nlist	<i>Number of Terms</i>
---------------	------------------------

---

**Description**

Gets the number of terms of an MCMC object.

**Usage**

```
## S3 method for class 'nlist'
nchains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Details**

Always 1L.

**Value**

A integer scalar of the number of terms.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

**Examples**

```
nchains(nlist(x = 1:2))
```

---

nchains.nlists	<i>Number of Terms</i>
----------------	------------------------

---

**Description**

Gets the number of terms of an MCMC object.

**Usage**

```
## S3 method for class 'nlists'
nchains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A integer scalar of the number of terms.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

**Examples**

```
nchains(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
nchains(split_chains(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))))
```

---

niters.mcmc	<i>Number of Iterations</i>
-------------	-----------------------------

---

**Description**

Gets the number of iterations (in a chain) of an MCMC object.

**Usage**

```
## S3 method for class 'mcmc'
niters(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

An integer scalar of the number of iterations.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)

---

niters.mcmc.list	<i>Number of Iterations</i>
------------------	-----------------------------

---

**Description**

Gets the number of iterations (in a chain) of an MCMC object.

**Usage**

```
## S3 method for class 'mcmc.list'  
niters(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

An integer scalar of the number of iterations.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)

---

niters.nlist	<i>Number of Iterations</i>
--------------	-----------------------------

---

**Description**

Gets the number of iterations (in a chain) of an MCMC object.

**Usage**

```
## S3 method for class 'nlist'  
niters(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Details**

Always 1.

**Value**

An integer scalar of the number of iterations.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)

**Examples**

```
niters(nlist(x = 1:2))
```

---

niters.nlists	<i>Number of Iterations</i>
---------------	-----------------------------

---

**Description**

Gets the number of iterations (in a chain) of an MCMC object.

**Usage**

```
## S3 method for class 'nlists'
niters(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

An integer scalar of the number of iterations.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#), [nterms\(\)](#)

**Examples**

```
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

---

nlist	<i>Create nlist Object</i>
-------	----------------------------

---

**Description**

Creates a [nlist\\_object\(\)](#) from one of more uniquely named numeric arguments.

**Usage**

```
nlist(...)
```

**Arguments**

...	Uniquely named numeric objects.
-----	---------------------------------

**Details**

An nlist object is an S3 class list of uniquely named numeric elements.

nlist objects are the raw data inputs for analytic engines such as JAGS, STAN and TMB.

**Value**

An nlist object.

**See Also**

[nlists\(\)](#)

**Examples**

```
nlist()
nlist(x = 1)
nlist(y = 1:4, zz = matrix(1:9, 3))
```

---

nlists

---

*Create nlists Object*


---

**Description**

Creates an [nlists\\_object\(\)](#) from one of more [nlist\\_object\(\)](#)s.

**Usage**

```
nlists(...)
```

**Arguments**

...                    nlist objects.

**Details**

An nlists object is a S3 class list of [nlist\\_object\(\)](#) elements with the same names, dimensionalities and typeofs.

nlists objects are useful for storing individual realizations of a simulated data set.

**Value**

An nlists object.

**See Also**

[nlist\(\)](#)

**Examples**

```
nlists()
nlists(nlist())
nlists(nlist(x = 1))
nlists(nlist(x = 1), nlist(x = -3))
```

---

npdims.mcmc.list	<i>Number of Parameter Dimensions</i>
------------------	---------------------------------------

---

### Description

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of `pdims()` as an integer vector.

### Usage

```
## S3 method for class 'mcmc.list'
npdims(x, ...)
```

### Arguments

x	An object.
...	Other arguments passed to methods.

### Value

A named integer vector of the number of dimensions of each parameter.

### See Also

Other dimensions: `dims()`, `ndims()`, `pdims()`

---

npdims.nlist	<i>Number of Parameter Dimensions</i>
--------------	---------------------------------------

---

### Description

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of `pdims()` as an integer vector.

### Usage

```
## S3 method for class 'nlist'
npdims(x, ...)
```

### Arguments

x	An object.
...	Other arguments passed to methods.

### Value

A named integer vector of the number of dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [pdims\(\)](#)

**Examples**

```
npdims(nlist(x = 1:3))
npdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

---

npdims.nlists	<i>Number of Parameter Dimensions</i>
---------------	---------------------------------------

---

**Description**

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of [pdims\(\)](#) as an integer vector.

**Usage**

```
## S3 method for class 'nlists'
npdims(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A named integer vector of the number of dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [pdims\(\)](#)

**Examples**

```
npdims(nlists(nlist(x = 1:3)))
npdims(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

---

nsims.nlist	<i>Number of Simulations</i>
-------------	------------------------------

---

**Description**

Gets the number of simulations (iterations \* chains) of an MCMC object.

The default methods returns the product of `nchains()` and `niters()`.

**Usage**

```
## S3 method for class 'nlist'
nsims(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Details**

Always 1L.

**Value**

An integer scalar of the number of simulations.

**See Also**

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nterms()`

**Examples**

```
nsims(nlist(x = 1:2))
```

---

nsims.nlists	<i>Number of Simulations</i>
--------------	------------------------------

---

**Description**

Gets the number of simulations (iterations \* chains) of an MCMC object.

The default methods returns the product of `nchains()` and `niters()`.

**Usage**

```
## S3 method for class 'nlists'
nsims(x, ...)
```



## Arguments

`x`                      An object.  
`...`                    Other arguments passed to methods.

## Value

An integer scalar of the number of simulations.

## See Also

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nterms\(\)](#)

## Examples

```
nsims(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
nsims(split_chains(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))))
```

---

<code>nterms.mcmc</code>	<i>Number of Terms</i>
--------------------------	------------------------

---

## Description

Gets the number of terms of an MCMC object.

## Usage

```
## S3 method for class 'mcmc'
nterms(x, ...)
```

## Arguments

`x`                      An object.  
`...`                    Other arguments passed to methods.

## Value

A integer scalar of the number of terms.

## See Also

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

---

nterms.mcmc.list	<i>Number of Terms</i>
------------------	------------------------

---

**Description**

Gets the number of terms of an MCMC object.

**Usage**

```
## S3 method for class 'mcmc.list'
nterms(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A integer scalar of the number of terms.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

---

nterms.nlist	<i>Number of Terms</i>
--------------	------------------------

---

**Description**

Gets the number of terms of an MCMC object.

**Usage**

```
## S3 method for class 'nlist'
nterms(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A integer scalar of the number of terms.

**See Also**

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

## Examples

```
nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
```

---

nterms.nlists	<i>Number of Terms</i>
---------------	------------------------

---

## Description

Gets the number of terms of an MCMC object.

## Usage

```
## S3 method for class 'nlists'
nterms(x, ...)
```

## Arguments

x                      An object.  
...                      Other arguments passed to methods.

## Value

A integer scalar of the number of terms.

## See Also

Other MCMC dimensions: [nchains\(\)](#), [niters\(\)](#), [npars\(\)](#), [nsams\(\)](#), [nsims\(\)](#)

## Examples

```
nterms(nlists(nlist(x = 1:3)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

---

pars.mcmc	<i>Parameter Names</i>
-----------	------------------------

---

## Description

Gets the parameter names.

## Usage

```
## S3 method for class 'mcmc'
pars(x, scalar = NULL, terms = FALSE, ...)
```

**Arguments**

x	An object.
scalar	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
terms	A flag specifying whether to return the parameter name for each term element.
...	Other arguments passed to methods.

**Value**

A character vector of the names of the parameters.

**See Also**

Other parameters: [npars\(\)](#), [set\\_pars\(\)](#)

---

pars.mcmc.list	<i>Parameter Names</i>
----------------	------------------------

---

**Description**

Gets the parameter names.

**Usage**

```
## S3 method for class 'mcmc.list'
pars(x, scalar = NULL, terms = FALSE, ...)
```

**Arguments**

x	An object.
scalar	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
terms	A flag specifying whether to return the parameter name for each term element.
...	Other arguments passed to methods.

**Value**

A character vector of the names of the parameters.

**See Also**

Other parameters: [npars\(\)](#), [set\\_pars\(\)](#)

---

pars.nlist	<i>Parameter Names</i>
------------	------------------------

---

**Description**

Gets the parameter names.

**Usage**

```
## S3 method for class 'nlist'
pars(x, scalar = NULL, terms = FALSE, ...)
```

**Arguments**

x	An object.
scalar	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
terms	A flag specifying whether to return the parameter name for each term element.
...	Other arguments passed to methods.

**Value**

A character vector of the names of the parameters.

**See Also**

Other parameters: [npars\(\)](#), [set\\_pars\(\)](#)

**Examples**

```
pars(nlist(zz = 1, y = 3:6))
```

---

pars.nlists	<i>Parameter Names</i>
-------------	------------------------

---

**Description**

Gets the parameter names.

**Usage**

```
## S3 method for class 'nlists'
pars(x, scalar = NULL, terms = FALSE, ...)
```

**Arguments**

x	An object.
scalar	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
terms	A flag specifying whether to return the parameter name for each term element.
...	Other arguments passed to methods.

**Value**

A character vector of the names of the parameters.

**See Also**

Other parameters: [npars\(\)](#), [set\\_pars\(\)](#)

**Examples**

```
pars(nlists(nlist(zz = 1, y = 3:6), nlist(zz = 4, y = 13:16)))
```

---

pdims.mcmc	<i>Parameter Dimensions</i>
------------	-----------------------------

---

**Description**

Gets the dimensions of each parameter of an object.

**Usage**

```
## S3 method for class 'mcmc'
pdims(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A named list of integer vectors of the dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [npdims\(\)](#)

---

pdims.mcmc.list	<i>Parameter Dimensions</i>
-----------------	-----------------------------

---

**Description**

Gets the dimensions of each parameter of an object.

**Usage**

```
## S3 method for class 'mcmc.list'
pdims(x, ...)
```

**Arguments**

x                    An object.  
 ...                  Other arguments passed to methods.

**Value**

A named list of integer vectors of the dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [npdims\(\)](#)

---

pdims.nlist	<i>Parameter Dimensions</i>
-------------	-----------------------------

---

**Description**

Gets the dimensions of each parameter of an object.

**Usage**

```
## S3 method for class 'nlist'
pdims(x, ...)
```

**Arguments**

x                    An object.  
 ...                  Other arguments passed to methods.

**Value**

A named list of integer vectors of the dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [npdims\(\)](#)

**Examples**

```
pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

---

pdims.nlists	<i>Parameter Dimensions</i>
--------------	-----------------------------

---

**Description**

Gets the dimensions of each parameter of an object.

**Usage**

```
## S3 method for class 'nlists'
pdims(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

A named list of integer vectors of the dimensions of each parameter.

**See Also**

Other dimensions: [dims\(\)](#), [ndims\(\)](#), [npdims\(\)](#)

**Examples**

```
pdims(nlists(nlist(x = 1:3)))
pdims(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

---

relist_nlist	<i>Relists an unlist nlist Object</i>
--------------	---------------------------------------

---

**Description**

Relists an nlist object that has been unlisted to a named numeric vector. Ensures absent terms are included and preserves integer class.

**Usage**

```
relist_nlist(flesh, skeleton)
```

**Arguments**

flesh	An atomic vector
skeleton	An nlist object.



**Value**

A numeric vector of the values in `x`.

**See Also**

`as_nlist.numeric()` and `unlist_nlist()`

**Examples**

```
relist_nlist(c(`a[2]` = 5), nlist(a = 1:3))
```

---

 set\_pars.mcmc

*Set Parameters*


---

**Description**

Sets an object's parameter names.

The assignment version `pars<-( )` forwards to `set_pars( )`.

**Usage**

```
## S3 method for class 'mcmc'
set_pars(x, value, ...)
```

**Arguments**

<code>x</code>	An object.
<code>value</code>	A character vector of the new parameter names.
<code>...</code>	Other arguments passed to methods.

**Details**

`value` must be a unique character vector of the same length as the object's parameters.

**Value**

The modified object.

**See Also**

Other parameters: `npars( )`, `pars( )`

---

set_pars.mcmc.list	<i>Set Parameters</i>
--------------------	-----------------------

---

**Description**

Sets an object's parameter names.

The assignment version `pars<-()` forwards to `set_pars()`.

**Usage**

```
## S3 method for class 'mcmc.list'
set_pars(x, value, ...)
```

**Arguments**

<code>x</code>	An object.
<code>value</code>	A character vector of the new parameter names.
<code>...</code>	Other arguments passed to methods.

**Details**

`value` must be a unique character vector of the same length as the object's parameters.

**Value**

The modified object.

**See Also**

Other parameters: [npars\(\)](#), [pars\(\)](#)

---

set_pars.nlist	<i>Set Parameter Names</i>
----------------	----------------------------

---

**Description**

Sets an object's parameter names.

The assignment version `pars<-()` forwards to `set_pars()`.

**Usage**

```
## S3 method for class 'nlist'
set_pars(x, value, ...)
```

**Arguments**

<code>x</code>	An object.
<code>value</code>	A character vector of the new parameter names.
<code>...</code>	Other arguments passed to methods.

**Details**

value must be a unique character vector of the same length as the object's parameters.

**Value**

The modified object.

**See Also**

Other parameters: [npars\(\)](#), [pars\(\)](#)

**Examples**

```
nlist <- nlist(x = 1, y = 3:4)
pars(nlist) <- c("a", "b")
nlist
set_pars(nlist, c("z", "c1"))
```

---

set_pars.nlists	<i>Set Parameter Names</i>
-----------------	----------------------------

---

**Description**

Sets an object's parameter names.

The assignment version `pars<-()` forwards to `set_pars()`.

**Usage**

```
## S3 method for class 'nlists'
set_pars(x, value, ...)
```

**Arguments**

x	An object.
value	A character vector of the new parameter names.
...	Other arguments passed to methods.

**Details**

value must be a unique character vector of the same length as the object's parameters.

**Value**

The modified object.

**See Also**

Other parameters: [npars\(\)](#), [pars\(\)](#)

**Examples**

```
nlists <- nlists(nlist(x = 2), nlist(x = 3))
pars(nlists) <- "a"
nlists
set_pars(nlists, "zz")
```

---

split_chains.nlists	<i>Split Chains</i>
---------------------	---------------------

---

**Description**

Splits each of an MCMC object's chains in half to double the number of chains and halve the number of iterations.

**Usage**

```
## S3 method for class 'nlists'
split_chains(x, ...)
```

**Arguments**

x	An object.
...	Other arguments passed to methods.

**Value**

The modified object.

**See Also**

Other MCMC manipulations: [bind\\_chains\(\)](#), [bind\\_iterations\(\)](#), [collapse\\_chains\(\)](#), [estimates\(\)](#)

**Examples**

```
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

---

subset.mcmc	<i>Subset mcmc Object</i>
-------------	---------------------------

---

**Description**

Subsets an mcmc object by its parameters and/or iterations.

**Usage**

```
## S3 method for class 'mcmc'
subset(x, iters = NULL, pars = NULL, iterations = NULL, parameters = NULL, ...)
```

**Arguments**

x	An mcmc object.
iters	An integer vector of iterations.
pars	A character vector of parameter names.
iterations	An integer vector (or NULL) of the iterations to subset by.
parameters	A character vector (or NULL) of the parameters to subset by.
...	Unused.

**Details**

Future versions should allow it to be reordered by its parameters.

**Value**

An mcmc object.

**Examples**

```
mcmc <- as_mcmc(nlist(beta = 1:2, theta = 1))
subset(mcmc, pars = "beta")
subset(mcmc, iters = c(1L,1L))
```

---

subset.mcmc.list	<i>Subset mcmc.list Object</i>
------------------	--------------------------------

---

**Description**

Subsets an mcmc.list object by its chains, parameters and/or iterations.

**Usage**

```
## S3 method for class 'mcmc.list'
subset(
  x,
  chains = NULL,
  iters = NULL,
  pars = NULL,
  iterations = NULL,
  parameters = NULL,
  ...
)
```

**Arguments**

x	An mcmc.list object.
chains	An integer vector of chains.
iters	An integer vector of iterations.
pars	A character vector of parameter names.
iterations	An integer vector (or NULL) of the iterations to subset by.
parameters	A character vector (or NULL) of the parameters to subset by.
...	Unused.

**Details**

Future versions should allow it to be reordered by its parameters.

**Value**

An mcmc.list object.

**Examples**

```
mcmc.list <- as_mcmc_list(nlists(nlist(beta = 1:2, theta = 1),
                                nlist(beta = 3:4, theta = -1)))
subset(mcmc.list, pars = "beta")
subset(mcmc.list, iters = c(1L,1L))
```

---

subset.nlist

*Subset nlist Object*


---

**Description**

Subsets an nlist object by its parameters.

**Usage**

```
## S3 method for class 'nlist'
subset(x, pars = NULL, ...)
```

**Arguments**

x	An nlist object.
pars	A character vector of parameter names.
...	Unused.

**Details**

It can also be used to reorder the parameters.

**Value**

An nlist object.

**Examples**

```
nlist <- nlist(a = 1, y = 3, x = 1:4)
subset(nlist)
subset(nlist, "a")
subset(nlist, c("x", "a"))
```

---

subset.nlists	<i>Subset nlists Object</i>
---------------	-----------------------------

---

## Description

Subsets an nlists object by its parameters, chains and iterations.

## Usage

```
## S3 method for class 'nlists'  
subset(x, chains = NULL, iters = NULL, pars = NULL, ...)
```

## Arguments

x	An nlists object.
chains	An integer vector of chains.
iters	An integer vector of iterations.
pars	A character vector of parameter names.
...	Unused.

## Details

It can also be used to reorder the parameters as well as duplicate chains and iterations.

## Value

An nlists object.

## Examples

```
nlists <- nlists(  
  nlist(a = 1, y = 3, x = 1:4),  
  nlist(a = 2, y = 4, x = 4:1),  
  nlist(a = 3, y = 6, x = 5:2)  
)  
subset(nlists)  
subset(nlists, pars = "a")  
subset(nlists, pars = c("x", "a"))  
subset(nlists, iters = 1L)  
subset(nlists, iters = c(2L, 2L))
```

---

thin.default	<i>Thin MCMC Object</i>
--------------	-------------------------

---

**Description**

Thins an MCMC object's iterations.

**Usage**

```
## Default S3 method:
thin(x, nthin = 1L, ...)
```

**Arguments**

x	An object.
nthin	A positive integer of the thinning rate.
...	Unused.

**Value**

The thinned MCMC object.

**Examples**

```
thin(nlists(nlist(x = 1), nlist(x = 2), nlist(x = 3), nlist(x = 4)), nthin = 2)
```

---

tidy.mcmc	<i>Turn an object into a tidy tibble</i>
-----------	--

---

**Description**

Turn an object into a tidy tibble

**Usage**

```
## S3 method for class 'mcmc'
tidy(x, simplify = FALSE, ...)
```

**Arguments**

x	An object.
simplify	A flag specifying whether to drop sd and zscore columns.
...	Unused.

**Value**

A `tibble::tibble()` with information about model components.

**Methods**

No methods found in currently loaded packages.



---

tidy.mcmc.list	Turn an object into a tidy tibble
----------------	-----------------------------------

---

**Description**

Turn an object into a tidy tibble

**Usage**

```
## S3 method for class 'mcmc.list'  
tidy(x, simplify = FALSE, ...)
```

**Arguments**

x	An object.
simplify	A flag specifying whether to drop sd and zscore columns.
...	Unused.

**Value**

A `tibble::tibble()` with information about model components.

**Methods**

No methods found in currently loaded packages.

---

tidy.nlists	Turn an object into a tidy tibble
-------------	-----------------------------------

---

**Description**

Turn an object into a tidy tibble

**Usage**

```
## S3 method for class 'nlists'  
tidy(x, simplify = FALSE, ...)
```

**Arguments**

x	An object.
simplify	A flag specifying whether to drop sd and zscore columns.
...	Unused.

**Value**

A `tibble::tibble()` with information about model components.

## Methods

No methods found in currently loaded packages.

## Examples

```
tidy(nlists(  
  nlist(x = 1, y = 4:6),  
  nlist(x = 3, y = 7:9)  
), simplify = TRUE)
```

---

unlist.nlist	<i>Flatten nlist Object</i>
--------------	-----------------------------

---

## Description

Flatten nlist Object

## Usage

```
## S3 method for class 'nlist'  
unlist(x, recursive = TRUE, use.names = TRUE)
```

## Arguments

x	An nlist object.
recursive	Ignored.
use.names	A flag specifying whether to preserve names.

## Value

A named numeric vector of the values in x.

## See Also

[unlist\\_nlist\(\)](#)

## Examples

```
unlist(nlist(y = 2, x = matrix(4:7, ncol = 2)))
```

---

unlist_nlist	<i>Flatten nlist Object</i>
--------------	-----------------------------

---

**Description**

Simplifies an nlist object to an named numeric vector where the names are the terms.

**Usage**

```
unlist_nlist(x)
```

**Arguments**

x                      An nlist object.

**Value**

A named numeric vector of the values in x.

**See Also**

[as\\_nlist.numeric\(\)](#) and [relist\\_nlist\(\)](#)

**Examples**

```
unlist_nlist(nlist(y = 2, x = matrix(4:7, ncol = 2)))
```

---

vld_nlist	<i>Validate nlist Object or nlists Object</i>
-----------	---

---

**Description**

Validate nlist Object or nlists Object

**Usage**

```
vld_nlist(x)
```

```
vld_nlists(x)
```

**Arguments**

x                      The object to check.

**Value**

A flag indicating whether the object was validated.

**Functions**

- `vld_nlists`: Validate nlists Object

**Examples**

```
# vld_nlist
vld_nlist(nlist(x = 1))
try(vld_nlist(list(x = 1)))

# vld_nlists
vld_nlists(nlists(nlist(x = 1)))
vld_nlists(1)
```

# Index

- \* **aggregate**
  - aggregate.nlist, [3](#)
  - aggregate.nlists, [4](#)
- \* **coerce term**
  - as\_term.mcmc, [9](#)
  - as\_term.nlist, [9](#)
  - as\_term.nlists, [10](#)
  - as\_term\_frame, [10](#)
  - as\_term\_frame.nlist, [11](#)
  - as\_term\_frame.nlists, [11](#)
- \* **coerce**
  - as\_nlist, [6](#)
  - as\_nlists, [8](#)
- \* **collapse**
  - collapse\_chains.mcmc, [14](#)
  - collapse\_chains.nlist, [15](#)
  - collapse\_chains.nlists, [16](#)
- \* **mcmc**
  - as\_mcmc, [4](#)
  - as\_mcmc\_list, [5](#)

aggregate.nlist, [3](#), [4](#)  
aggregate.nlists, [3](#), [4](#)  
as.nlist(as\_nlist), [6](#)  
as.nlists(as\_nlist), [6](#)  
as\_mcmc, [4](#), [6](#)  
as\_mcmc\_list, [5](#), [5](#)  
as\_nlist, [6](#), [8](#)  
as\_nlist.numeric(), [41](#), [51](#)  
as\_nlists, [7](#), [8](#)  
as\_term.mcmc, [9](#), [9](#), [10–12](#)  
as\_term.nlist, [9](#), [9](#), [10–12](#)  
as\_term.nlists, [9](#), [10](#), [10](#), [11](#), [12](#)  
as\_term\_frame, [9](#), [10](#), [10](#), [11](#), [12](#)  
as\_term\_frame.nlist, [9](#), [10](#), [11](#), [12](#)  
as\_term\_frame.nlists, [9–11](#), [11](#)

bind\_chains, [12](#), [13](#), [15](#), [18](#), [44](#)  
bind\_iterations, [15](#), [18](#), [44](#)  
bind\_iterations.mcmc, [12](#)  
bind\_iterations.mcmc.list, [13](#)

chk\_nlist, [13](#)  
chk\_nlists(chk\_nlist), [13](#)

coda::as.mcmc(), [5](#)  
collapse\_chains, [12](#), [13](#), [18](#), [44](#)  
collapse\_chains.mcmc, [14](#), [16](#)  
collapse\_chains.mcmc.list, [15](#)  
collapse\_chains.nlist, [14](#), [15](#), [16](#)  
collapse\_chains.nlists, [14](#), [16](#), [16](#)  
complete\_terms.mcmc, [17](#)

dims, [30](#), [31](#), [38–40](#)

estimates, [12](#), [13](#), [15](#), [44](#)  
estimates.nlist, [17](#)  
estimates.nlists, [18](#)

fill\_all, [21](#), [22](#)  
fill\_all.nlist, [19](#)  
fill\_all.nlists, [20](#)  
fill\_na, [19](#), [20](#)  
fill\_na.nlist, [21](#)  
fill\_na.nlists, [22](#)

is\_nlist(is\_numeric), [23](#)  
is\_nlists(is\_numeric), [23](#)  
is\_numeric, [23](#)

nchains, [25–28](#), [32–35](#)  
nchains(), [32](#)  
nchains.mcmc, [24](#)  
nchains.mcmc.list, [24](#)  
nchains.nlist, [25](#)  
nchains.nlists, [25](#)  
ndims, [30](#), [31](#), [38–40](#)  
niters, [24–26](#), [32–35](#)  
niters(), [32](#)  
niters.mcmc, [26](#)  
niters.mcmc.list, [27](#)  
niters.nlist, [27](#)  
niters.nlists, [28](#)  
nlist, [28](#)  
nlist(), [29](#)  
nlist-object(nlist), [28](#)  
nlist\_object(nlist), [28](#)  
nlist\_object(), [3](#), [4](#), [6](#), [23](#), [28](#), [29](#)  
nlists, [29](#)  
nlists(), [29](#)

`nlists-object(nlists)`, 29  
`nlists_object(nlists)`, 29  
`nlists_object()`, 4, 8, 23, 29  
`npars`, 24–28, 32–38, 41–43  
`npdims`, 38–40  
`npdims.mcmc.list`, 30  
`npdims.nlist`, 30  
`npdims.nlists`, 31  
`nsams`, 24–28, 32–35  
`nsims`, 24–28, 33–35  
`nsims.nlist`, 32  
`nsims.nlists`, 32  
`nterms`, 24, 26–28, 32, 33  
`nterms.mcmc`, 33  
`nterms.mcmc.list`, 34  
`nterms.nlist`, 34  
`nterms.nlists`, 35

`pars`, 41–43  
`pars.mcmc`, 35  
`pars.mcmc.list`, 36  
`pars.nlist`, 37  
`pars.nlists`, 37  
`pdims`, 30, 31  
`pdims()`, 30, 31  
`pdims.mcmc`, 38  
`pdims.mcmc.list`, 38  
`pdims.nlist`, 39  
`pdims.nlists`, 40

`relist_nlist`, 40  
`relist_nlist()`, 51

`set_pars`, 36–38  
`set_pars.mcmc`, 41  
`set_pars.mcmc.list`, 42  
`set_pars.nlist`, 42  
`set_pars.nlists`, 43  
`split_chains`, 12, 13, 15, 18  
`split_chains.nlists`, 44  
`subset.mcmc`, 44  
`subset.mcmc.list`, 45  
`subset.nlist`, 46  
`subset.nlists`, 47

`thin.default`, 48  
`tibble::tibble()`, 48, 49  
`tidy.mcmc`, 48  
`tidy.mcmc.list`, 49  
`tidy.nlists`, 49

`unlist.nlist`, 50  
`unlist_nlist`, 51  
`unlist_nlist()`, 41, 50  
`vld_nlist`, 51  
`vld_nlists(vld_nlist)`, 51