

# Package ‘comat’

March 5, 2026

**Title** Creates Co-Occurrence Matrices of Spatial Data

**Version** 0.9.7

**Description** Builds co-occurrence matrices based on spatial raster data. It includes creation of weighted co-occurrence matrices (wecoma) and integrated co-occurrence matrices (incoma; Vadivel et al. (2007) <[doi:10.1016/j.patrec.2007.01.004](https://doi.org/10.1016/j.patrec.2007.01.004)>).

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 7.3.3

**Depends** R (>= 2.10)

**LinkingTo** Rcpp, RcppArmadillo

**Imports** Rcpp

**Suggests** tinytest, covr, knitr, rmarkdown

**URL** <https://jakubnowosad.com/comat/>

**BugReports** <https://github.com/Nowosad/comat/issues>

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Jakub Nowosad [aut, cre] (ORCID: <<https://orcid.org/0000-0002-1057-3721>>), Maximillian H.K. Hesselbarth [ctb] (Co-author of underlying C++ code for `get_class_index_map()`, `get_unique_values()`, and `rcpp_get_coma()` functions), Marco Sciaini [ctb] (Co-author of underlying C++ code for `get_class_index_map()`, `get_unique_values()`, and `rcpp_get_coma()` functions), Sebastian Hanss [ctb] (Co-author of underlying C++ code for `get_class_index_map()`, `get_unique_values()`, and `rcpp_get_coma()` functions)

**Maintainer** Jakub Nowosad <[nowosad.jakub@gmail.com](mailto:nowosad.jakub@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-03-05 15:20:02 UTC

## Contents

get_cocoma	2
get_cocove	3
get_coma	4
get_cove	4
get_incoma	5
get_incove	6
get_wecoma	7
get_wecove	8
it_metric	9
raster_w	10
raster_w_na	10
raster_x	11
raster_x_na	11
raster_y	11
<b>Index</b>	<b>12</b>

---

get_cocoma	<i>Create a co-located co-occurrence matrix (cocoma)</i>
------------	--

---

### Description

Create a co-located co-occurrence matrix (cocoma)

### Usage

```
get_cocoma(x, y, neighbourhood = 4, classes = NULL)
```

### Arguments

x	A matrix with categories
y	A matrix with categories
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.
classes	A list of length 2 with the values of selected classes from the x and y objects. It is used to calculate cocoma only for selected classes.

### Value

A co-located co-occurrence matrix

## Examples

```
library(comat)
data(raster_x, package = "comat")
data(raster_x_na, package = "comat")

coom = get_cocoma(raster_x, raster_x_na)
coom

get_cocoma(raster_x, raster_x_na, classes = list(c(1, 2), 3))
```

---

get_cocove	<i>Create a co-located co-occurrence vector (cocove)</i>
------------	--

---

## Description

Converts a co-located co-occurrence matrix (cocoma) to a co-located co-occurrence vector (cocove)

## Usage

```
get_cocove(x, ordered = TRUE, normalization = "none")
```

## Arguments

x	A matrix - an output of the <code>get_cocoma()</code> function
ordered	The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
normalization	Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "none".

## Value

A co-located co-occurrence vector

## Examples

```
library(comat)
data(raster_x, package = "comat")
data(raster_x_na, package = "comat")

coom = get_cocoma(raster_x, raster_x_na)
coom

coov = get_cocove(coom)
coov
```

---

get\_coma                      *Create a co-occurrence matrix (coma)*

---

### Description

Create a co-occurrence matrix (coma)

### Usage

```
get_coma(x, neighbourhood = 4, classes = NULL)
```

### Arguments

**x**                      A matrix with categories

**neighbourhood**      The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.

**classes**              A vector or a list with the values of selected classes from the x object. It is used to calculate coma only for selected classes.

### Value

A co-occurrence matrix

### Examples

```
#library(comat)
data(raster_x, package = "comat")

com = get_coma(raster_x)
com

com2 = get_coma(raster_x, classes = c(1, 3))
com2

data(raster_x_na, package = "comat")
com3 = get_coma(raster_x_na, classes = c(0:3, NA))
com3
```

---

get\_cove                      *Create a co-occurrence vector (cove)*

---

### Description

Converts a co-occurrence matrix (coma) to a co-occurrence vector (cove)

**Usage**

```
get_cove(x, ordered = TRUE, normalization = "none")
```

**Arguments**

**x** A matrix - an output of the `get_coma()` function

**ordered** The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.

**normalization** Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "none".

**Value**

A co-occurrence vector

**Examples**

```
library(comat)
data(raster_x, package = "comat")

com = get_coma(raster_x)
com

cov = get_cove(com)
cov

cov = get_cove(com, normalization = "pdf")
cov
```

---

get\_incoma

*Create an integrated co-occurrence matrix (incoma)*

---

**Description**

Create an integrated co-occurrence matrix (incoma)

**Usage**

```
get_incoma(x, neighbourhood = 4, classes = NULL)
```

**Arguments**

**x** A list object containing categorical matrices with categories

**neighbourhood** The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.

**classes** A list of the same length as x with the values of selected classes from all of the objects in x. It is used to calculate incoma only for selected classes.

**Value**

An integrated co-occurrence matrix

**Examples**

```
data(raster_x, package = "comat")
data(raster_w, package = "comat")
x = list(raster_x, raster_w, raster_x)

get_incoma(x)

get_incoma(x, classes = list(1:2, 2:4, 1))
```

---

get_incove	<i>Create an integrated co-occurrence vector (incove)</i>
------------	---

---

**Description**

Converts an integrated co-occurrence matrix (incoma) to an integrated co-occurrence vector (incove)

**Usage**

```
get_incove(x, ordered = TRUE, repeated = TRUE, normalization = "none")
```

**Arguments**

x	A matrix - an output of the <code>get_incoma()</code> function
ordered	The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE. See details for more explanation.
repeated	Should the repeated co-located co-occurrence matrices be used? Either "repeated" (TRUE) or "unrepeated" (FALSE). The default is TRUE. See details for more explanation.
normalization	Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "none".

**Details**

All values are kept when `ordered = TRUE` and `repeated = TRUE`. When `ordered = TRUE` and `repeated = FALSE` all values from `cocoma` (but only one `cocoma` for each pair) and all `coma` values are kept. `ordered = FALSE` and `repeated = TRUE` keeps all values from `cocoma`, but divides `coma` values by 2. `ordered = FALSE` and `repeated = FALSE` keeps all values from `cocoma` (but only one `cocoma` for each pair), and divides `coma` values by 2.

**Value**

An integrated co-occurrence vector

## Examples

```
library(comat)

data(raster_x, package = "comat")
data(raster_w, package = "comat")
x = list(raster_x, raster_w, raster_x)

incom = get_incoma(x)
incom

incov1 = get_incove(incom)
incov1

incov2 = get_incove(incom, ordered = FALSE)
incov2

incov3 = get_incove(incom, ordered = FALSE, normalization = "pdf")
incov3
```

---

get\_wecoma

*Create a weighted co-occurrence matrix (wecoma)*

---

## Description

Create a weighted co-occurrence matrix (wecoma)

## Usage

```
get_wecoma(
  x,
  w,
  neighbourhood = 4,
  classes = NULL,
  fun = "mean",
  na_action = "replace"
)
```

## Arguments

x	A matrix with categories
w	A matrix with weights
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.
classes	A vector or a list with the values of selected classes from the x object. It is used to calculate wecoma only for selected classes.

fun	Function to calculate values from adjacent cells to contribute to output matrix, "mean" - calculate average values from adjacent cells of weight matrix, "geometric_mean" - calculate geometric mean values from adjacent cells of weight matrix, or "focal" assign value from the focal cell.
na_action	Decides on how to behave in the presence of missing values in w. Possible options are "replace", "omit", "keep". The default, "replace", replaces missing values with 0, "omit" does not use cells with missing values, and "keep" keeps missing values.

**Value**

A weighted co-occurrence matrix

**Examples**

```
library(comat)
data(raster_x, package = "comat")
data(raster_w, package = "comat")

wom = get_wecoma(raster_x, raster_w)
wom

get_wecoma(raster_x, raster_w, classes = list(c(1, 3)))
```

---

get_wecove	<i>Create a weighted co-occurrence vector (wecove)</i>
------------	--

---

**Description**

Converts a weighted co-occurrence matrix (wecoma) to a weighted co-occurrence vector (wecove)

**Usage**

```
get_wecove(x, ordered = TRUE, normalization = "none")
```

**Arguments**

x	A matrix - an output of the <code>get_wecoma()</code> function
ordered	The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
normalization	Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "none".

**Value**

A weighted co-occurrence vector

**Examples**

```

library(comat)
data(raster_x, package = "comat")
data(raster_w, package = "comat")

wom = get_wecoma(raster_x, raster_w)
wom

wov = get_wecove(wom)
wov

```

---

it_metric	<i>Calculates an Information Theory-based metric</i>
-----------	--

---

**Description**

Calculates a selected Information Theory-based metric based on a provided co-occurrence matrix

**Usage**

```
it_metric(x, metric, base = "log2", ordered = TRUE)
```

**Arguments**

x	A matrix - an output of the <code>get_coma()</code> function
metric	One of the following: "ent" (Marginal entropy), "jointent" (Joint entropy), "condent" (Conditional entropy), "mutinf" (Mutual information), or "relmutinf" (Relative mutual information)
base	The unit in which entropy is measured. The default is "log2", which compute entropy in "bits". "log" and "log10" can be also used.
ordered	The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.

**Value**

A single numeric value

**References**

Nowosad J., TF Stepinski. 2019. Information theory as a consistent framework for quantification and classification of landscape patterns. <https://doi.org/10.1007/s10980-019-00830-x>

**Examples**

```
library(comat)
data(raster_x, package = "comat")

com = get_coma(raster_x)
com

it_metric(com, metric = "ent")
it_metric(com, metric = "joinent")
it_metric(com, metric = "condent")
it_metric(com, metric = "mutinf")
it_metric(com, metric = "relmutinf")
```

---

raster_w	<i>A matrix with weights</i>
----------	------------------------------

---

**Description**

A matrix with weights

**Usage**

```
data(raster_w)
```

**Format**

A matrix

---

raster_w_na	<i>A matrix with weights and missing values</i>
-------------	---

---

**Description**

A matrix with weights and missing values

**Usage**

```
data(raster_w_na)
```

**Format**

A matrix

---

raster_x	<i>A matrix with categories</i>
----------	---------------------------------

---

**Description**

A matrix with categories

**Usage**

```
data(raster_x)
```

**Format**

A matrix

---

raster_x_na	<i>A matrix with categories and missing values</i>
-------------	--

---

**Description**

A matrix with categories and missing values

**Usage**

```
data(raster_x_na)
```

**Format**

A matrix

---

raster_y	<i>A matrix with categories</i>
----------	---------------------------------

---

**Description**

A matrix with categories

**Usage**

```
data(raster_y)
```

**Format**

A matrix

# Index

## \* datasets

- raster\_w, [10](#)
- raster\_w\_na, [10](#)
- raster\_x, [11](#)
- raster\_x\_na, [11](#)
- raster\_y, [11](#)

- get\_cocoma, [2](#)
- get\_cocoma(), [3](#)
- get\_cocove, [3](#)
- get\_coma, [4](#)
- get\_coma(), [5](#), [9](#)
- get\_cove, [4](#)
- get\_incoma, [5](#)
- get\_incoma(), [6](#)
- get\_incove, [6](#)
- get\_wecoma, [7](#)
- get\_wecoma(), [8](#)
- get\_wecove, [8](#)

- it\_metric, [9](#)

- raster\_w, [10](#)
- raster\_w\_na, [10](#)
- raster\_x, [11](#)
- raster\_x\_na, [11](#)
- raster\_y, [11](#)