Package 'mrf2d'

October 13, 2022

Type Package

Title Markov Random Field Models for Image Analysis

Version 1.0

Maintainer Victor Freguglia <victorfreguglia@gmail.com>

Description Model fitting, sampling and visualization
for the (Hidden) Markov Random Field model with pairwise interactions and
general interaction structure from
Freguglia, Garcia & Bicas (2020) <doi:10.1002/env.2613>,
which has many popular models used in 2-dimensional lattices
as particular cases, like the Ising Model and Potts Model.
A complete manuscript describing the package is available in
Freguglia & Garcia (2022) <doi:10.18637/jss.v101.i08>.

License GPL-3

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.1), dplyr(>= 0.8.1), tidyr, methods, ggplot2,
Rdpack

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

RoxygenNote 7.1.2

Suggests testthat (>= 2.1.0), covr, knitr, rmarkdown, glue

VignetteBuilder knitr

URL https://github.com/Freguglia/mrf2d

BugReports https://github.com/Freguglia/mrf2d/issues

NeedsCompilation yes

Author Victor Freguglia [aut, cre] (<https://orcid.org/0000-0002-6189-4453>)

Repository CRAN

Date/Publication 2022-01-25 23:52:42 UTC

R topics documented:

`basis_functions` *Creation of basis functions*

Description

`fourier_2d()` and `polynomial_2d()` creates a `list` of basis functions to be used as the fixed effect in `fit_ghm`.

Usage

```
fourier_2d(max_freqs, lattice_dim)

polynomial_2d(poly_deg, lattice_dim)
```

Arguments

`max_freqs`	A length 2 numeric vector with maximum frequencies considered (x-axis and y-axis direction, respectively).
`lattice_dim`	A length 2 numeric vector with lattice dimensions (N,M) to be used.
`poly_deg`	A length 2 numeric vector with degrees of the bivariate polynomial to be considered.

Details

`fourier_2d()` is for 2-dimensional Fourier transform.

Value

A `list` of functions.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
fourier_2d(c(10,10), dim(Z_potts))
polynomial_2d(c(3,3), dim(Z_potts))
```

---

bold5000 *BOLD5000 neuroimaging data*

Description

An image extracted from the "BOLD5000" open dataset. It was read from the file in path `BOLD5000/DS001499/SUB-CSI2/SE` available at the OpenNeuro platform (https://openneuro.org/datasets/ds001499/versions/1.3.0).

Usage

```
bold5000
```

Format

An object of class `matrix` (inherits from `array`) with 176 rows and 256 columns.

Details

The file was read using the `oro.nifti` package and the image was extracted from the matrix in slice 160.

References

Chang, N., Pyles, J. A., Marcus, A., Gupta, A., Tarr, M. J., & Aminoff, E. M. (2019). BOLD5000, a public fMRI dataset while viewing 5000 visual images. Scientific data, 6(1), 1-18.

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

`cpmrf2d` *Conditional probabilities in a pixel position*

Description

Computes the vector of conditional probabilities for a pixel position given a field, an interaction structure and a parameter array.

Usage

```
cp_mrf2d(Z, mrfi, theta, pos)
```

Arguments

`Z`	A `matrix` with integers in `{0,...,C}`.
`mrfi`	A `mrfi` object representing the interaction structure.
`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`pos`	Length-2 vector with the position to compute conditional probabilities in.

Value

A `numeric` vector with the conditional probabilities.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
cp_mrf2d(Z_potts, mrfi(1), theta_potts, c(57,31))
cp_mrf2d(Z_potts, mrfi(1), theta_potts*0.1, c(57,31))
```

data_examples *Example Data*

Description

`mrf2d` contains a set of simulated fields to illustrate its usage.

field1 A binary field sampled from a sparse interaction structure: `mrfi(1) + c(4,4)`

hfield1 A continuous valued field, obtained by Gaussian mixture driven by `field1`.

Usage

```
field1

hfield1
```

Format

An object of class `matrix` (inherits from `array`) with 150 rows and 150 columns.

An object of class `matrix` (inherits from `array`) with 150 rows and 150 columns.

Author(s)

Victor Freguglia

dplot *Plotting functions for lattice data*

Description

`dplot()` and `cplot()` are functions for plotting lattice data. They are an alternative to base R's `image()` function using `ggplot2` instead. `dplot` is used for discrete data and `cplot` for continuous data, they only differ in the fact that pixel values are treated as a factor in `dplot`, therefore, a discrete scale is used.

Usage

```
dplot(Z, legend = FALSE)

cplot(Y, legend = TRUE)
```

Arguments

`Z`	A `matrix` object with integers only.
`legend`	`logical` indicating whether a legend should be included or not.
`Y`	A `matrix` object with continuous values.

Details

Since returns a ggplot object, other layers can be added to it using the usual ggplot2 syntax in order to modify any aspect of the plot.

The data frame used to create the object has columns named x, y and value, which are mapped to x, y and fill, respectively, used with geom_tile().

Value

a ggplot object.

Author(s)

Victor Freguglia

Examples

```
# Plotting discrete data
dplot(Z_potts)

#Making it continuous
cplot(Z_potts + rnorm(length(Z_potts)))

#Adding extra ggplot layers
library(ggplot2)
dplot(Z_potts) + ggtitle("This is a title")
dplot(Z_potts, legend = TRUE) + scale_fill_brewer(palette = "Set1")
```

---

fit_ghm *EM estimation for Gaussian Hidden Markov field*

Description

fit_ghm fits a Gaussian Mixture model with hidden components driven by a Markov random field with known parameters. The inclusion of a linear combination of basis functions as a fixed effect is also possible.

The algorithm is a modification of of (Zhang et al. 2001), which is described in (Freguglia et al. 2020).

Usage

```
fit_ghm(
  Y,
  mrfi,
  theta,
  fixed_fn = list(),
```

```
  equal_vars = FALSE,
  init_mus = NULL,
  init_sigmas = NULL,
  maxiter = 100,
  max_dist = 10^-3,
  icm_cycles = 6,
  verbose = interactive(),
  qr = NULL
)
```

Arguments

`Y`	A matrix of observed (continuous) pixel values.
`mrfi`	A `mrfi` object representing the interaction structure.
`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`fixed_fn`	A list of functions `fn(x,y)` to be considered as a fixed effect. See `basis_functions`.
`equal_vars`	`logical` indicating if the mixture model has the same variances in all mixture components.
`init_mus`	Optional. A `numeric` with length (C+1) with the initial mean estimate for each component.
`init_sigmas`	Otional. A `numeric` with length (C+1) with initial sample deviation estimate for each component.
`maxiter`	The maximum number of iterations allowed. Defaults to 100.
`max_dist`	Defines a stopping condition. The algorithm stops if the maximum absolute difference between parameters of two consecutive iterations is less than `max_dist`.
`icm_cycles`	Number of steps used in the Iterated Conditional Modes algorithm executed in each interaction. Defaults to 6.
`verbose`	`logical` indicating wheter to print the progress or not.
`qr`	The QR decomposition of the design matrix. Used internally.

Details

If either `init_mus` or `init_sigmas` is `NULL` an EM algorithm considering an independent uniform distriburion for the hidden component is fitted first and its estimated means and sample deviations are used as initial values. This is necessary because the algorithm may not converge if the initial parameter configuration is too far from the maximum likelihood estimators.

`max_dist` defines a stopping condition. The algorithm will stop if the maximum absolute difference between (μ and σ) parameters in consecutive iterations is less than `max_dist`.

Value

A `hmrfout` containing:

- `par`: A `data.frame` with μ and σ estimates for each component.
- `fixed`: A `matrix` with the estimated fixed effect in each pixel.
- `Z_pred`: A `matrix` with the predicted component (highest probability) in each pixel.
- `predicted`: A `matrix` with the fixed effect + the μ value for the predicted component in each pixel.
- `iterations`: Number of EM iterations done.

Author(s)

Victor Freguglia

References

Freguglia V, Garcia NL, Bicas JL (2020). "Hidden Markov random field models applied to color homogeneity evaluation in dyed textile images." *Environmetrics*, **31**(4), e2613.

Zhang Y, Brady M, Smith S (2001). "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm." *IEEE transactions on medical imaging*, **20**(1), 45–57.

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
# Sample a Gaussian mixture with components given by Z_potts
# mean values are 0, 1 and 2 and a linear effect on the x-axis.

set.seed(2)
Y <- Z_potts + rnorm(length(Z_potts), sd = 0.4) +
      (row(Z_potts) - mean(row(Z_potts)))*0.01
# Check what the data looks like
cplot(Y)

fixed <- polynomial_2d(c(1,0), dim(Y))
fit <- fit_ghm(Y, mrfi = mrfi(1), theta = theta_potts, fixed_fn = fixed)
fit$par
cplot(fit$fixed)
```

fit_pl *Maximum Pseudo-likelihood fitting of MRFs on 2d lattices.*

Description

Parameter estimation for Markov random fields via Pseudo-Likelihood function optimization. See pl_mrf2d for information on the Pseudo-Likelihood function.

Usage

```
fit_pl(
  Z,
  mrfi,
  family = "onepar",
  init = 0,
  optim_args = list(method = "BFGS"),
  return_optim = FALSE
)
```

Arguments

Z	A matrix object containing the observed MRF. NA values can be used to create a subregion of the lattice for non-rectangular data.
mrfi	A mrfi object representing the interaction structure.
family	The family of parameter restrictions to potentials. Families are: 'onepar', 'oneeach', 'absdif', 'dif' or 'free'. See mrf2d-familiy.
init	The initial value to be used in the optimization. It can be: • A valid array of parameter values according to family. • 0. If set to 0 an array with '0" in all entries is created.
optim_args	Additional parameters passed to optim().
return_optim	logical indicating whether information from the optim() call are returned.

Value

An object of class mrfout with elements:

- theta: The estimated array of potential values.
- mrfi: The interaction structure considered.
- family: The parameter restriction family considered.
- method: The estimation method ("Pseudolikelihood").
- value: The optimal pseudo-likelihood value.
- opt.xxx(if return_optim is TRUE): Information returned by the optim() function used for the optimization.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
fit_pl(Z_potts, mrfi(1), family = "onepar")
fit_pl(Z_potts, mrfi(1), family = "oneeach")
fit_pl(Z_potts, mrfi(2), family = "onepar")
```

`fit_sa` *Stochastic Approximation fitting of MRFs on 2d lattices*

Description

Estimates the parameters of a MRF by successively sampling from a parameter configuration and updating it by comparing the sufficient statistics of the sampled field and the observed field.

This method aims to find the parameter value where the gradient of the likelihood function is equal to zero.

Usage

```
fit_sa(
  Z,
  mrfi,
  family = "onepar",
  gamma_seq,
  init = 0,
  cycles = 5,
  refresh_each = length(gamma_seq) + 1,
  refresh_cycles = 60,
  verbose = interactive()
)
```

Arguments

`Z`	A `matrix` object containing the observed MRF. `NA` values can be used to create a subregion of the lattice for non-rectangular data.
`mrfi`	A `mrfi` object representing the interaction structure.
`family`	The family of parameter restrictions to potentials. Families are: `'onepar'`, `'oneeach'`, `'absdif'`, `'dif'` or `'free'`. See `mrf2d-familiy`.
`gamma_seq`	A `numeric` vector with the sequence of constants used in each step γ_t.

`init`	The initial value to be used in the optimization. It can be: • A valid `array` of parameter values according to `family`. • `0`. If set to `0` an array with '0" in all entries is created.
`cycles`	The number of updates to be done (for each each pixel).
`refresh_each`	An integer with the number of iterations taken before a complete refresh (restart from a random state). This prevents the sample from being stuck in a mode for too long. Defaults to `length(gamma_seq) + 1` (no refresh happens).
`refresh_cycles`	An integer indicating how many Gibbs Sampler cycles are performed when a refresh happens. Larger is usually better, but slower.
`verbose`	`logical` indicating whether the iteration number is printed during execution.

Details

The stochastic approximation method consists of, given an observed field `Z`, and a starting parameters configuration θ_0, successively sample a field Z_t from the current parameter configuration and estimate the direction of the gradient of the likelihood function by comparing the sufficient statistics in the current sample and the observed field.

The solution is updated by moving in the estimated direction with a predefined step size γ_t, a new field Z_{t+1} is sampled using the new parameter configuration and Z_t as an initial value, and the process is repeated.

$$\theta_{t+1} = \theta_t - \gamma_t(T(Z_t) - T(Z)),$$

where $T(Z)$ is the sufficient statistics for the reference field, $T(Z_t)$ is the sufficient statistics for a field sampled from θ_t.

`gamma_seq` is normalized internally by diving values by `length(Z)`, so the choice of the sequence is invariant to the lattice dimensions. Typically, a sequence like `seq(from = 1, to = 0, length.out = 1000)` should be used for defining a sequence with `1000` steps. Some tuning of this sequence is required.

Value

A `mrfout` object with the following elements:

- `theta`: The estimated `array` of potentials.
- `mrfi`: The interaction structure considered.
- `family`: The parameter restriction family considered.
- `method`: The estimation method (`"Stochastic Approximation"`).
- `metrics`: A `data.frame` containing the the euclidean distance between the sufficient statics computed for `Z` and the current sample.

Note

Stochastic Approximation is called "Controllable Simulated Annealing" in some references.

Examples where Stochastic Approximation is used with MRFs are (Gimel'farb 1996), (Atchadé et al. 2013).

Author(s)

Victor Freguglia

References

Wikipedia (2019). "Stochastic approximation." https://en.wikipedia.org/wiki/Stochastic_approximation.

Atchadé YF, Lartillot N, Robert C, others (2013). "Bayesian computation for statistical models with intractable normalizing constants." *Brazilian Journal of Probability and Statistics*, **27**(4), 416–436.

Gimel'farb GL (1996). "Texture modeling by multiple pairwise pixel interactions." *IEEE Transactions on pattern analysis and machine intelligence*, **18**(11), 1110–1114.

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
set.seed(2)
fit1 <- fit_sa(Z_potts, mrfi(1), family = "oneeach", gamma_seq = seq(1, 0, length.out = 50))
# Estimated parameters
fit1$theta
# A visualization of estimated gradient norm over iterations.
plot(fit1$metrics, type = "l")

fit_sa(Z_potts, mrfi(1), family = "oneeach", gamma_seq = seq(1, 0, length.out = 50))
```

---

hmrfout *MRF fitting functions output*

Description

MRF fitting functions output

Usage

```
## S3 method for class 'hmrfout'
print(x, ...)

## S3 method for class 'hmrfout'
summary(object, ...)

## S3 method for class 'hmrfout'
plot(x, ...)
```

Arguments

x	a hmrfout object.
...	other arguments not used by this method.
object	a hmrfout object.

---

mrf2d-family *Parameter restriction families*

Description

Different parameter restrictions can be included in estimation processes to make sure mrf2d can successfully include a wide range of model types in its inference functions.

For model identifiability, at least one linear restriction is necessary. mrf2d always assume $\theta_{0,0,r} = 0$ for all relative positions r.

Additionally, each family of restrictions may introduce other restrictions:

'onepar'

This family assumes the model is defined by a single parameter by adding the restriction

$$\theta_{a,b,r} = \phi * 1(a! = b).$$

Here $1()$ denotes de indicator function. In words, the parameter must be the same value for any pair with different values and 0 for any equal-valued pair.

'oneeach'

Similar to 'onepar', parameters are 0 for equal-valued pairs and a constant for pairs with different values, but the constant may differ between different relative positions r:

$$\theta a, b, r = \phi_r * 1(a! = b).$$

'absdif'

All parameters $\theta_{a,b,r}$ with the same absolute difference between a and b must be equal within each relative position r. (Note that 'absdif' is equal to 'oneeach' for binary images).

$$\theta_{a,b,r} = \sum_d \phi_{d,r} * 1(|a - b| == d)$$

'dif'

The same as 'absdif', but parameters may differ between positive and negative differences.

$$\theta_{a,b,r} = \sum_d \phi_{d,r} * 1(a - b == d)$$

'free'

No additional restriction, all parameters other than $\theta_{0,0,r}$ vary freely.

Author(s)

Victor Freguglia

See Also

`vignette("mrf2d-family", package = "mrf2d")`

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

---

mrfi-class *mrfi: MRF interaction structure*

Description

The `mrfi` S4 class is a representation of the interaction structure for a spatially-stationary Markov Random Field.

The function `mrfi()` provides an interface for creation `mrfi` objects. A `plot` method is also available for visualization, as well as conversion methods like `as.list` and operators like `+`.

`mrfi()` creates an object of class `mrfi` based on a distance rule and optionally a list of relative positions. The argument `max_norm` and `norm_type` can be used to automatically include all positions within a "range" defined by the norm type chosen and distance using that norm.

A list of relative positions may also be included to specify sparse interaction structures, for example. Alternatively, `rpositions()` can be used to create a based exclusively on a list of relative positions.

Simple operations are provided to include (set union) new interacting positions to a `mrfi` object with the `'+'` operator and remove positions (set difference) with `-`. Individual positions can be included/excluded using length-2 vectors in the right hand side. Union and set difference of complete structures can also be computed by adding or subtracting two `mrfi` objects.

These operations deal with opposite directions filtering to avoid redundancy in the interaction structure.

Usage

```
mrfi(max_norm = 1, norm_type = "1", positions = NULL)

rpositions(positions)

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

## S4 method for signature 'mrfi'
length(x)
```

```
## S4 method for signature 'mrfi,numeric,missing'
x[[i]]

## S4 method for signature 'mrfi,numeric,missing'
x[i]

## S4 method for signature 'mrfi,numeric'
e1 + e2

## S4 method for signature 'mrfi,numeric'
e1 - e2

## S4 method for signature 'mrfi,mrfi'
e1 + e2

## S4 method for signature 'mrfi,mrfi'
e1 - e2

mrfi_to_string(mrfi)
```

Arguments

`max_norm`	a `numeric` value. All points with norm $\leq$ `max_dist` are included.
`norm_type`	a `character` indicating the norm type used. Possible values are "m", "1", "2", etc. See `norm` for details.
`positions`	a `list` of `numeric` vectors of length 2. Each vector corresponds to a relative position included.
`x`	`mrfi` object.
`...`	other arguments not used by this method.
`i`	vector of indexes to extract interacting positions.
`e1, mrfi`	A `mrfi` object.
`e2`	Either a second `mrfi` object or a length 2 `numeric` with the new relative position to include (`+`) or remove (`-`).

Details

The interaction structure is defined by the list of relative positions in it. For a specific pixel, conditional to the values of pixels in these relative positions from it, its value is independent of any other pixel in the image.

The relative positions are indentified by two integers `rx` and `ry` representing the "shift" in the `x`-axis and `y`-axis respectively. As an example: The relative position `(1,0)` representes the pixel in the immediate right position, while `(-1,0)` the left one.

Note that the inclusion of a relative position to the dependence also implies its opposite direction is not conditionally independent (commutativeness of dependence), but only one is included in the `mrfi` object.

To illustrate that, a nearest neighbor dependence structure can be specified by:

`mrfi(1)`

Note that it only includes the positions `(1,0)` and `(0,1)`, but when visualizing it, for example, `mrf2d` understands the opposite directions are also conditionally dependent, as in

`plot(mrfi(1))`.

Value

A `mrfi` object.

`as.list()`: converts the `mrfi` object to a list of interacting positions (list of length-2 vectors).

`[[`: converts to list and subsets it.

`[`: subsets the `mrfi` object and returns another `mrfi` object.

`+`: computes the union of the interaction structure in a `mrfi` object with a `numeric` representing an additional position to include or another `mrfi` object.

Slots

`Rmat` A 2-column `matrix` where each row represents a relative position of interaction.

Note

If a position in `positions` is already included due to the `max_norm` and `norm_type` specification, the second ocurrence is ignored. The same is valid for repeated or opposite positions in `positions`.

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
plot(mrfi(max_norm = 2, norm_type = "1"))
plot(mrfi(max_norm = 2, norm_type = "m"))
plot(mrfi(max_norm = 2, norm_type = "1", positions = list(c(4,4))))

as.list(mrfi(1))
mrfi(1)[[1]]
mrfi(2)[[1:3]]

mrfi(1)
rpositions(list(c(1,0), c(0,1)))
mrfi(2)
mrfi(2, norm_type = "m")
mrfi(1, positions = list(c(4,4), c(-4,4)))


#Repeated positions are handled automatically
mrfi(1, positions = list(c(1,0), c(2,0)))

mrfi(1) + c(2,0)
mrfi(1) - c(1,0)
```

```
mrfi(1) + mrfi(0, positions = list(c(2,0)))
mrfi(2) - mrfi(1)
```

`mrfout` *MRF fitting functions output*

Description

MRF fitting functions output

Usage

```
## S3 method for class 'mrfout'
print(x, ...)

## S3 method for class 'mrfout'
summary(object, ...)

## S3 method for class 'mrfout'
plot(x, ...)
```

Arguments

`x`	a `mrfout` object.
`...`	other arguments not used by this method.
`object`	a `mrfout` object.

`plot.mrfi` *Plotting of* `mrfi` *objects.*

Description

Plots a visual representation of the interaction structure described in a `mrfi` object. The black tile represents a reference pixel and gray tiles are shown in relative positions with dependent pixels.

A `ggplot` object is used, therefore, the user can load the `ggplot2` package and add more `ggplot` layers to freely customize the `plot`.

Usage

```
## S3 method for class 'mrfi'
plot(x, include_axis = FALSE, include_opposite = TRUE, ...)
```

Arguments

x	A mrfi object.
include_axis	logical indicating whether the axis and grid lines are included. If FALSE theme_void() is added to the ggplot object.
include_opposite	´logical‘ whether opposite directions should be included in the visualization of the dependence structure.
...	other arguments not used by this method.

Details

The data.frame used for the ggplot call has columns names rx and ry repŕesenting the relative positions.

Value

A ggplot object using geom_tile() to represent interacting relative positions.

Author(s)

Victor Freguglia

Examples

```
plot(mrfi(1))

library(ggplot2)
plot(mrfi(1)) + geom_tile(fill = "red")
plot(mrfi(1)) + geom_tile(fill = "blue") + theme_void()

plot(mrfi(1)) + geom_text(aes(label = paste0("(",rx,",",ry,")")))
```

---

pl_mrf2d *Pseudo-likelihood function for MRFs on 2d lattices*

Description

Computes the pseudo-likelihood function of a Markov Random Field on a 2-dimensional lattice.

Usage

```
pl_mrf2d(Z, mrfi, theta, log_scale = TRUE)
```

Arguments

`Z`	A `matrix` with integers in `{0,...,C}`.
`mrfi`	A `mrfi` object representing the interaction structure.
`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`log_scale`	A `logical` value indicating whether the returned value should be in logarithmic scale.

Details

The pseudo-likelihood function is defined as the product of conditional probabilities of each pixel given its neighbors:

$$\prod_i P(Z_i|Z_{N_i},\theta).$$

Value

A `numeric` with the pseudo-likelihood value.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

Examples

```
pl_mrf2d(Z_potts, mrfi(1), theta_potts)
```

`rmrf2d` *Sampling of Markov Random Fields on 2d lattices*

Description

Performs pixelwise updates based on conditional distributions to sample from a Markov random field.

Usage

```
rmrf2d(
  init_Z,
  mrfi,
  theta,
  cycles = 60,
  sub_region = NULL,
  fixed_region = NULL
)
```

Arguments

`init_Z`	One of two options: • A `matrix` object with the initial field configuration. Its valuesmust be integers in `{0,...,C}`. • A length 2 `numeric` vector with the lattice dimensions.
`mrfi`	A `mrfi` object representing the interaction structure.
`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`cycles`	The number of updates to be done (for each each pixel).
`sub_region`	`NULL` if the whole lattice is considered or a `logical matrix` with `TRUE` for pixels in the considered region.
`fixed_region`	`NULL` if the whole lattice is to be sampled or a `logical matrix` with `TRUE` for pixels to be considered fixed. Fixed pixels are not updated in the Gibbs Sampler.

Details

This function implements a Gibbs Sampling scheme to sample from a Markov random field by iteratively sampling pixel values from the conditional distribution

$$P(Z_i|Z_{N_i},\theta).$$

A cycle means exactly one update to each pixel. The order pixels are sampled is randomized within each cycle.

If `init_Z` is passed as a length 2 vector with lattice dimensions, the initial field is sampled from independent discrete uniform distributions in `{0,...,C}`. The value of C is obtained from the number of rows/columns of `theta`.

A MRF can be sampled in a non-rectangular region of the lattice with the use of the `sub_region` argument or by setting pixels to `NA` in the initial configuration `init_Z`. Pixels with `NA` values in `init_Z` are completely disconsidered from the conditional probabilities and have the same effect as setting `sub_region = is.na(init_Z)`. If `init_Z` has `NA` values, `sub_region` is ignored and a warning is produced.

A specific region can be kept constant during the Gibbs Sampler by using the `fixed_region` argument. Keeping a subset of pixels constant is useful when you want to sample in a specific region of the image conditional to the rest, for example, in texture synthesis problems.

Value

A `matrix` with the sampled field.

Note

As in any Gibbs Sampling scheme, a large number of cycles may be required to achieve the target distribution, specially for strong interaction systems.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08.

`rmrf2d_mc` for generating multiple points of a Markov Chain to be used in Monte-Carlo methods.

Examples

```
# Sample using specified lattice dimension
Z <- rmrf2d(c(150,150), mrfi(1), theta_potts)

#Sample using itial configuration

Z2 <- rmrf2d(Z, mrfi(1), theta_potts)

# View results
dplot(Z)
dplot(Z2)

# Using sub-regions
subreg <- matrix(TRUE, 150, 150)
subreg <- abs(row(subreg) - 75) + abs(col(subreg) - 75) <= 80
# view the sub-region
dplot(subreg)

Z3 <- rmrf2d(c(150,150), mrfi(1), theta_potts, sub_region = subreg)
dplot(Z3)

# Using fixed regions
fixreg <- matrix(as.logical(diag(150)), 150, 150)
# Set initial configuration: diagonal values are 0.
init_Z4 <- Z
init_Z4[fixreg] <- 0

Z4 <- rmrf2d(init_Z4, mrfi(1), theta_potts, fixed_region = fixreg)
dplot(Z4)

# Combine fixed regions and sub-regions
Z5 <- rmrf2d(init_Z4, mrfi(1), theta_potts,
fixed_region = fixreg, sub_region = subreg)
```

```
dplot(Z5)
```

`rmrf2d_mc` *Markov Chain sampling of MRFs for Monte-Carlo methods*

Description

Generates a Markov Chain of random fields and returns the sufficient statistics for each of the observations.

This function automatizes the process of generating a random sample of MRFs to be used in Monte-Carlo methods by wrapping `rmrf2d` and executing it multiple time while storing sufficient statistics instead of the entire lattice.

Usage

```
rmrf2d_mc(
  init_Z,
  mrfi,
  theta,
  family,
  nmc = 100,
  burnin = 100,
  cycles = 4,
  verbose = interactive()
)
```

Arguments

`init_Z`	One of two options: • A `matrix` object with the initial field configuration. Its valuesmust be integers in `{0,...,C}`. • A length 2 `numeric` vector with the lattice dimensions.
`mrfi`	A `mrfi` object representing the interaction structure.
`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`family`	The family of parameter restrictions to potentials. Families are: `'onepar'`, `'oneeach'`, `'absdif'`, `'dif'` or `'free'`. See `mrf2d-familiy`.
`nmc`	Number of samples to be stored.
`burnin`	Number of cycles iterated before start collecting sufficient statistics.
`cycles`	Number of cycles between collected samples.
`verbose`	`logical` indicating whether to print iteration number or not.

Value

A matrix where each row contains the vector of sufficient statistics for an observation.

Note

Fixed regions and incomplete lattices are not supported.

Author(s)

Victor Freguglia

Examples

```
rmrf2d_mc(c(80, 80), mrfi(1), theta_potts, family = "oneeach", nmc = 8)
```

---

smr_array *Summarized representation of theta arrays*

Description

`smr_array` creates a vector containing only the free parameters from an array given a restriction `family`. `exapand_array` is the reverse operation, expanding a complete array from the vector of sufficient statistics.

Usage

```
smr_array(theta, family)

expand_array(theta_vec, family, mrfi, C)
```

Arguments

`theta`	A 3-dimensional array describing potentials. Slices represent interacting positions, rows represent pixel values and columns represent neighbor values. As an example: `theta[1,3,2]` has the potential for the pair of values 0,2 observed in the second relative position of `mrfi`.
`family`	The family of parameter restrictions to potentials. Families are: `'onepar'`, `'oneeach'`, `'absdif'`, `'dif'` or `'free'`. See `mrf2d-familiy`.
`theta_vec`	A `numeric` vector with the free parameters of a potential array. It's dimension depends on the restriction `family`, `C` and the number of interacting positions on `mrfi`.
`mrfi`	A `mrfi` object representing the interaction structure.
`C`	The maximum value of the field.

Details

The order the parameters appear in the summarized vector matches the order in `smr_stat()`.

`vec_description()` provides a `data.frame` object describing which are the relative positions and interactions associated with each element of the summarized vector in the same order.

Value

`smr_array` returns a numeric vector with the free parameters of `theta`.

`expand_array` returns a three-dimensional `array` of potentials.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08

Examples

```
smr_array(theta_potts, "onepar")
smr_array(theta_potts, "oneeach")

expand_array(0.99, family = "onepar", mrfi = mrfi(1), C = 2)
expand_array(c(0.1, 0.2), family = "oneeach", mrfi = mrfi(1), C = 3)
```

---

smr_stat *Summary Statistics*

Description

Computes the summary count statistics of a field given an interaction structure and a restriction family.

- `cohist()` computes the co-ocurrence histogram.
- `smr_stat()` computes the co-ocurrence histogram, then converts it into a vector of sufficient statistics given a `family` of restrictions.

Usage

```
smr_stat(Z, mrfi, family)

cohist(Z, mrfi)

vec_description(mrfi, family, C)
```

Arguments

Z	A matrix object containing the observed MRF. NA values can be used to create a subregion of the lattice for non-rectangular data.
mrfi	A mrfi object representing the interaction structure.
family	The family of parameter restrictions to potentials. Families are: 'onepar', 'oneeach', 'absdif', 'dif' or 'free'. See mrf2d-familiy.
C	The maximum value of the field.

Details

The order the summarized counts appear in the summary vector matches the order in smr_array().

Value

A numeric vector with the summarized counts.

An array representing the co-ocurrence histogram of Z in the relative positions contained in mrfi. Each row and column corresponds a pair of values in (0, ..., C) and each slice corresponds to

A data.frame describing the relative position and interaction associated with each potential in the vector form in each row, in the same order.

Author(s)

Victor Freguglia

See Also

A paper with detailed description of the package can be found at doi: 10.18637/jss.v101.i08

Examples

```
smr_stat(Z_potts, mrfi(1), "onepar")
smr_stat(Z_potts, mrfi(1), "oneeach")

cohist(Z_potts, mrfi(1))
```

Z_potts *Example objects from* mrf2d

Description

Z_potts and theta_potts are example objects for mrf2d.

Z_potts is a matrix object containing an observed lattice of a 3 color (C = 2) Potts model.

theta_potts is the parameter array used to sample it, it consists of a configuration with one parameter (-1.0) and two relative positions (to be used with a nearest-neighbor structure).

Author(s)

Victor Freguglia

Examples

```
theta_potts
dplot(Z_potts)
```

Index