Package 'latentcor'

August 9, 2022

Type Package

Title Fast Computation of Latent Correlations for Mixed Data

Version 2.0.0

Maintainer Mingze Huang <mingzehuang@gmail.com>

Description The first stand-alone R package for computation of latent correlation that takes into account all variable types (continuous/binary/ordinal/zero-inflated),
comes with an optimized memory footprint, and is computationally efficient, essentially making latent correlation estimation almost as fast as rank-based correlation estimation.
The estimation is based on latent copula Gaussian models.
For continuous/binary types, see Fan, J., Liu, H., Ning, Y., and Zou, H. (2017).
For ternary type, see Quan X., Booth J.G. and Wells M.T. (2018) <arXiv:1809.06255>.
For truncated type or zero-
inflated type, see Yoon G., Carroll R.J. and Gaynanova I. (2020) <doi:10.1093/biomet/asaa007>.
For approximation method of computation, see Yoon G., Müller C.L. and Gaynanova I. (2021) <doi:10.1080/10618600.2021.1882468>. The latter method uses multilinear interpolation originally implemented in the R package <https://cran.r-project.org/package=chebpol>.

Depends R (>= 3.0.0)

Imports stats, pcaPP, fMultivar, mnormt, Matrix, MASS, heatmaply, ggplot2, plotly, graphics, geometry, doFuture, foreach, future, doRNG, microbenchmark

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.1

Suggests rmarkdown, markdown, knitr, testthat (>= 3.0.0), lattice, cubature, plot3D, covr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Mingze Huang [aut, cre] (<https://orcid.org/0000-0003-3919-1564>), Grace Yoon [aut] (<https://orcid.org/0000-0003-3263-1352>), Christian Müller [aut] (<https://orcid.org/0000-0002-3821-7083>), Irina Gaynanova [aut] (<https://orcid.org/0000-0002-4116-0268>)

Repository CRAN

Date/Publication 2022-08-09 05:20:02 UTC

R topics documented:

`evaluation` *Numerical evaluation for different estimation methods.*

Description

Speed and accuracy comparison of two different estimation methods.

Usage

```
evaluation(
  genfun,
  estfun_1,
  estfun_2,
  grid_list,
  nrep = 100,
  showplot = FALSE,
  cores = detectCores(),
  ...
)
```

Arguments

`genfun`	A data generation function.
`estfun_1`	A function for first estimation method.
`estfun_2`	A function for second estimation method.
`grid_list`	A list for grid points to be evaluated (each element of list is a vector represents ticklabels on a dimension). The number of list elements are the dimension of function inputs.
`nrep`	Number of replications in simulation.
`showplot`	Logical indicator. `showplot = TRUE` generates the heatmaps of output arrays. NULL if `showplot = FALSE`.

cores	The numbers of cores (threads) of your machine to conduct parallel computing.
...	Other inputs for data generation or estimation functions to be passed through.

Value

evaluation returns

- meanAE_1: An array for mean absolute error of first estimation method.
- meanAE_2: An array for mean absolute error of second estimation method.
- medianAE_1: An array for median absolute error of first estimation method.
- medianAE_2: An array for median absolute error of second estimation method.
- maxAE_1: An array for maximum absolute error of first estimation method.
- maxAE_2: An array for maximum absolute error of second estimation method.
- meanAE_diff: An array for mean absolute error of difference between two estimations.
- medianAE_diff: An array for median absolute error of difference between two estimations.
- maxAE_diff: An array for maximum absolute error of difference between two estimations.
- mediantime_1: An array for median time of first estimation method.
- mediantime_2: An array for median time of second estimation method.
- plot_meanAE_1: A plot for mean absolute error of first estimation method.
- plot_meanAE_2: A plot for mean absolute error of second estimation method.
- plot_medianAE_1: A plot for median absolute error of first estimation method.
- plot_medianAE_2: A plot for median absolute error of second estimation method.
- plot_maxAE_1: A plot for maximum absolute error of first estimation method.
- plot_maxAE_2: A plot for maximum absolute error of second estimation method.
- plot_meanAE_diff: A plot for mean absolute error of difference between two estimations.
- plot_medianAE_diff: A plot for median absolute error of difference between two estimations.
- plot_maxAE_diff: A plot for maximum absolute error of difference between two estimations.
- plot_mediantime_1: A plot for median time of first estimation method.
- plot_mediantime_2: A plot for median time of second estimation method.

---

gen_data *Mixed type simulation data generator*

Description

Generates data of mixed types from the latent Gaussian copula model.

Usage

```
gen_data(
  n = 100,
  types = c("ter", "con"),
  rhos = 0.5,
  copulas = "no",
  XP = NULL,
  showplot = FALSE
)
```

Arguments

n	A positive integer indicating the sample size. The default value is 100.
types	A vector indicating the type of each variable, could be "con" (continuous), "bin" (binary), "tru" (truncated) or "ter" (ternary). The number of variables is determined based on the length of types, that is p = length(types). The default value is c("ter", "con") which creates two variables: the first one is ternary, the second one is continuous.
rhos	A vector with lower-triangular elements of desired correlation matrix, e.g. rhos = c(.3, .5, .7) means the correlation matrix is matrix(c(1, .3, .5, .3, 1, .7, .5, .7, 1), 3, 3). If only a scalar is supplied (length(rhos) = 1), then equi-correlation matrix is assumed with all pairwise correlations being equal to rhos. The default value is 0.5 which means correlations between any two variables are 0.5.
copulas	A vector indicating the copula transformation f for each of the p variables, e.g. U = f(Z). Each element can take value "no" (f is identity), "expo" (exponential transformation) or "cube" (cubic transformation). If the vector has length 1, then the same transformation is applied to all p variables. The default value is "no": no copula transformation for any of the variables.
XP	A list of length p indicating proportion of zeros (for binary and truncated), and proportions of zeros and ones (for ternary) for each of the variables. For continuous variable, NA should be supplied. If NULL, the following values are automatically generated as elements of XP list for the corresponding data types: For continuous variable, the corresponding value is NA; for binary or truncated variable, the corresponding value is a number between 0 and 1 representing the proportion of zeros, the default value is 0.5; for ternary variable, the corresponding value is a pair of numbers between 0 and 1, the first number indicates the the proportion of zeros, the second number indicates the proportion of ones. The sum of a pair of numbers should be between 0 and 1, the default value is c(0.3, 0.5).
showplot	Logical indicator. If TRUE, generates the plot of the data when number of variables p is no more than 3. The default value is FALSE.

Value

gen_data returns a list containing

- X: Generated data matrix (n by p) of observed variables.
- plotX: Visualization of the data matrix X. Histogram if p=1. 2D Scatter plot if p=2. 3D scatter plot if p=3. Returns NULL if `showplot = FALSE`.

References

Fan J., Liu H., Ning Y. and Zou H. (2017) "High dimensional semiparametric latent graphicalmodel for mixed data" doi:10.1111/rssb.12168.

Yoon G., Carroll R.J. and Gaynanova I. (2020) "Sparse semiparametric canonical correlation analysis for data of mixed types" doi:10.1093/biomet/asaa007.

Examples

```
# Generate single continuous variable with exponential transformation (always greater than 0)
# and show histogram.
simdata = gen_data(n = 100, copulas = "expo", types = "con", showplot = FALSE)
X = simdata$X; plotX = simdata$plotX
# Generate a pair of variables (ternary and continuous) with default proportions
# and without copula transformation.
simdata = gen_data()
X = simdata$X
# Generate 3 variables (binary, ternary and truncated)
# corresponding copulas for each variables are "no" (no transformation),
# "cube" (cube transformation) and "cube" (cube transformation).
# binary variable has 30% of zeros, ternary variable has 20% of zeros
# and 40% of ones, truncated variable has 50% of zeros.
# Then show the 3D scatter plot (data points project on either 0 or 1 on Axis X1;
# on 0, 1 or 2 on Axas X2; on positive domain on Axis X3)
simdata = gen_data(n = 100, rhos = c(.3, .4, .5), copulas = c("no", "cube", "cube"),
          types = c("bin", "ter", "tru"), XP = list(.3, c(.2, .4), .5), showplot = TRUE)
X = simdata$X; plotX = simdata$plotX
# Check the proportion of zeros for the binary variable.
sum(simdata$X[ , 1] == 0)
# Check the proportion of zeros and ones for the ternary variable.
sum(simdata$X[ , 2] == 0); sum(simdata$X[ , 2] == 1)
# Check the proportion of zeros for the truncated variable.
sum(simdata$X[ , 3] == 0)
```

---

get_types *Automatically determine types of each variable (continuous/binary/ternary/truncated) in a data matrix.*

Description

Automatically determine types of each variable (continuous/binary/ternary/truncated) in a data matrix.

Usage

```
get_types(X, tru_prop = 0.05)
```

Arguments

X	A numeric data matrix (n by p), where n is number of samples, and p is number of variables. Missing values (NA) are allowed.
tru_prop	A scalar between 0 and 1 indicating the minimal proportion of zeros that should be present in a variable to be treated as "tru" (truncated type or zero-inflated) rather than as "con" (continuous type). The default value is 0.05 (any variable with more than 5% of zero values among n samples is treated as truncated or zero-inflated)

Value

get_types returns

- types: A vector of length p indicating the type of each of the p variables in X. Each element is one of "con" (continuous), "bin" (binary), "ter" (ternary) or "tru" (truncated).

Examples

```
X = gen_data(types = c("ter", "con"))$X
get_types(X)
```

---

interpolant *Evaluate an interpolant in a point*

Description

An interpolant is a function returned by ipol which has prespecified values in some points, and which fills in between with some reasonable values.

Arguments

x	The argument of the function. A function of more then one variable takes a vector. x can also be a matrix of column vectors.
threads	The number of threads to use for evaluation. All interpolants created by ipol are parallelized. If given a matrix argument x, the vectors can be evaluated in parallel.
...	Other parameters. Currently used for simplex linear interpolants with the logical argument The "multilinear" interpolant also has the argument blend=c("linear","cubic","sigmoi where a blending function can be chosen.

Value

A numeric. If more than one point was evaluated, a vector.

Author(s)

Simen Gaure

`interpolation` *Parallel version of multilinear interpolation generator for function approximation*

Description

Parallel version of multilinear interpolation generator for function approximation

The primary method is `ipol` which dispatches to some other method. All the generated interpolants accept as an argument a matrix of column vectors. The generated functions also accept an argument `threads=getOption('ipol.threads')` to utilize more than one CPU if a matrix of column vectors is evaluated. The option `ipol.threads` is initialized from the environment variable `IPOL_THREADS` upon loading of the package. It defaults to `1`.

Usage

```
interpolation(evalfun, grid_list, cores = detectCores(), int = FALSE, ...)
```

Arguments

`evalfun`	The objective function to be approximated.
`grid_list`	A list for grid points (each element of list is a vector represents ticklabels on a dimension). The number of list elements are the dimension of function inputs.
`cores`	The numbers of cores (threads) of your machine to conduct parallel computing.
`int`	Logical indicator. `int = TRUE` interpolant value multiplied by 10^7 then convert to interger to save memory. Original interpolant if `int = FALSE`.
`...`	Other inputs for objective functions to be passed through.

Details

The interpolants are ordinary R-objects and can be saved with `save()` and loaded later with `load()` or serialized/unserialized with other tools, just like any R-object. However, they contain calls to functions in the package, and while the author will make efforts to ensure that generated interpolants are compatible with future versions of **ipol**, I can issue no such absolute guarantee.

Value

`interpolation` returns

- value: A list of of length p corresponding to each variable. Returns NA for continuous variable; proportion of zeros for binary/truncated variables; the cumulative proportions of zeros and ones (e.g. first value is proportion of zeros, second value is proportion of zeros and ones) for ternary variable.
- interpolant: An interpolant function generated by `chebpol::chebpol` for interplation.

See Also

ipol, interpolant

Examples

```
grid_list = list(seq(-0.5, 0.5, by = 0.5), seq(-0.5, 0.5, by = 0.5))
objfun = function(x, y) {x^2 + sqrt(y)}
evalfun = function(X) {objfun(X[1], X[2])}
value = interpolation(evalfun = evalfun, grid_list = grid_list)$value
interpolant = interpolation(evalfun = evalfun, grid_list = grid_list)$interpolant
```

`ipol` *Create an interpolating function from given values. Several interpolation methods are supported.*

Description

Create an interpolating function from given values. Several interpolation methods are supported.

Usage

```
ipol(val, grid = NULL, ...)
```

Arguments

`val`	array or function. Function values on a grid.
`grid`	list. Each element is a vector of ordered grid-points for a dimension.
`...`	Further arguments to the function, if `is.function(val)`. And some extra arguments for interpolant creation described in section Details.

Value

A `function(x, threads=getOption('chebpol.threads'))` defined on a hypercube, an interpolant for the given function. The argument `x` can be a matrix of column vectors which are evaluated in parallel in a number of threads. The function yields values for arguments outside the hypercube as well, though it will typically be a poor approximation. `threads` is an integer specifying the number of parallel threads which should be used when evaluating a matrix of column vectors.

Author(s)

Simen Gaure

Examples

```
## evenly spaced grid-points
su <- seq(0,1,length.out=10)
## irregularly spaced grid-points
s <- su^3
## create approximation on the irregularly spaced grid
ml1 <- ipol(exp(s), grid=list(s))
## test it, since exp is convex, the linear approximation lies above
```

```
## the exp between the grid points
ml1(su) - exp(su)

## multi dimensional approximation
f <- function(x) 10/(1+25*mean(x^2))
# a 3-dimensional 10x10x10 grid, first and third coordinate are non-uniform
grid <- list(s, su, sort(1-s))

# make multilinear spline.
ml2 <- ipol(array(apply(expand.grid(grid), 1, f), c(10, 10, 10)), grid=grid)
# make 7 points in R3 to test them on
m <- matrix(runif(3*7),3)
rbind(true=apply(m,2,f), ml=ml2(m))
```

`latentcor` *Estimate latent correlation for mixed types.*

Description

Estimation of latent correlation matrix from observed data of (possibly) mixed types (continuous/binary/truncated/ternary) based on the latent Gaussian copula model. Missing values (NA) are allowed. The estimation is based on pairwise complete observations.

Usage

```
latentcor(
  X,
  types = NULL,
  method = c("approx", "original"),
  use.nearPD = TRUE,
  nu = 0.001,
  tol = 1e-08,
  ratio = 0.9,
  showplot = FALSE
)
```

Arguments

`X`	A numeric matrix or numeric data frame (n by p), where n is number of samples, and p is number of variables. Missing values (NA) are allowed, in which case the estimation is based on pairwise complete observations.
`types`	A vector of length p indicating the type of each of the p variables in `X`. Each element must be one of `"con"` (continuous), `"bin"` (binary), `"ter"` (ternary) or `"tru"` (truncated). If the vector has length 1, then all p variables are assumed to be of the same type that is supplied. The default value is `NULL`, and the variable types are determined automatically using function `get_types`. As automatic determination of variable types takes extra time, it is recommended to supply the types explicitly when they are known in advance.

method	The calculation method for latent correlations. Either "original" or "approx". If method = "approx", multilinear approximation method is used, which is much faster than the original method, see Yoon et al. (2021) for timing comparisons for various variable types. If method = "original", optimization of the bridge inverse function is used. The default is "approx".
use.nearPD	Logical indicator. use.nearPD = TRUE gets nearest positive definite matrix for the estimated latent correlation matrix with shrinkage adjustment by nu. Output R is the same as Rpointwise if use.nearPD = FALSE. Default value is TRUE.
nu	Shrinkage parameter for the correlation matrix, must be between 0 and 1. Guarantees that the minimal eigenvalue of returned correlation matrix is greater or equal to nu. When nu = 0, no shrinkage is performed, the returned correlation matrix will be semi-positive definite but not necessarily strictly positive definite. When nu = 1, the identity matrix is returned (not recommended). The default (recommended) value is 0.001.
tol	When method = "original", specifies the desired accuracy of the bridge function inversion via uniroot optimization and is passed to optimize. The default value is 1e-8. When method = "approx", this parameter is ignored.
ratio	When method = "approx", specifies the boundary value for multilinear interpolation, must be between 0 and 1. When ratio = 0, no linear interpolation is performed (the slowest execution) which is equivalent to method = "original". When ratio = 1, linear interpolation is always performed (the fastest execution) but may lead to high approximation errors. The default (recommended) value is 0.9 which controls the approximation error and has fast execution, see Yoon et al. (2021) for details. When method = "original", this parameter is ignored.
showplot	Logical indicator. showplot = TRUE generates a ggplot object plotR with the heatmap of latent correlation matrix R. plotR = NULL if showplot = FALSE. Default value is FALSE.

Details

The function estimates latent correlation by calculating inverse bridge function (Fan et al., 2017) evaluated at the value of sample Kendall's tau ($\hat{\tau}$). The bridge function F connects Kendall's tau to latent correlation r so that $F(r) = E(\hat{\tau})$. The form of function F depends on the variable types (continuous/binary/truncated/ternary), but is exact. The exact form of inverse is not available, so has to be evaluated numerically for each pair of variables leading to Rpointwise.

When method = "original", the inversion is done numerically by solving

$$minimize_r(F(r) - \hat{\tau})^2$$

using optimize. The parameter tol is used to control the accuracy of the solution.

When method = "approx", the inversion is done approximately by interpolating previously calculated and stored values of $F^{-1}(\hat{\tau})$. This is significantly faster than the original method (Yoon et al., 2021) for binary/ternary/truncated cases, however the approximation errors may be non-negligible on some regions of the space. The parameter ratio controls the region where the interpolation is performed with default recommended value of 0.9 giving a good balance of accuracy and computational speed . Increasing the value of ratio may improve speed (but possibly sacrifice the accuracy), whereas decreasing the value of ratio my improve accuracy (but possibly sacrifice the speed). See Yoon et al. 2021 and vignette for more details.

In case the pointwise estimator Rpointwise is has negative eigenvalues, it is projected onto the space of positive semi-definite matrices using nearPD. The parameter nu further allows to perform additional shrinkage towards identity matrix (desirable in cases where the number of variables p is very large) as

$$R = (1 - \nu)\tilde{R} + \nu I,$$

where $\tilde{R}$ is Rpointwise after projection by nearPD.

Value

latentcor returns

- zratios: A list of of length p corresponding to each variable. Returns NA for continuous variable; proportion of zeros for binary/truncated variables; the cumulative proportions of zeros and ones (e.g. first value is proportion of zeros, second value is proportion of zeros and ones) for ternary variable.
- K: (p x p) Kendall Tau (Tau-a) Matrix for X
- R: (p x p) Estimated latent correlation matrix for X
- Rpointwise: (p x p) Point-wise estimates of latent correlations for X. This matrix is not guaranteed to be semi-positive definite. This is the original estimated latent correlation matrix without adjustment for positive-definiteness.
- plotR: Heatmap plot of latent correlation matrix R, NULL if showplot = FALSE

References

Fan J., Liu H., Ning Y. and Zou H. (2017) "High dimensional semiparametric latent graphical model for mixed data" doi:10.1111/rssb.12168.

Yoon G., Carroll R.J. and Gaynanova I. (2020) "Sparse semiparametric canonical correlation analysis for data of mixed types" doi:10.1093/biomet/asaa007.

Yoon G., Müller C.L., Gaynanova I. (2021) "Fast computation of latent correlations" doi:10.1080/10618600.2021.1882468.

Examples

```
# Example 1 - truncated data type, same type for all variables
# Generate data
X = gen_data(n = 300, types = rep("tru", 5))$X
# Estimate latent correlation matrix with original method and check the timing
start_time = proc.time()
R_org = latentcor(X = X, types = "tru", method = "original")$R
proc.time() - start_time
# Estimate latent correlation matrix with approximation method and check the timing
start_time = proc.time()
R_approx = latentcor(X = X, types = "tru", method = "approx")$R
proc.time() - start_time
# Heatmap for latent correlation matrix.
Heatmap_R_approx = latentcor(X = X, types = "tru", method = "approx",
                             showplot = TRUE)$plotR
```

```
# Example 2 - ternary/continuous case
X = gen_data()$X
# Estimate latent correlation matrix with original method
R_nc_org = latentcor(X = X, types = c("ter", "con"), method = "original")$R
# Estimate latent correlation matrix with aprroximation method
R_nc_approx = latentcor(X = X, types = c("ter", "con"), method = "approx")$R
```

Index