Package 'STrollR'

September 3, 2020

Title Correct Standard Errors for Computing Spatial and Temporal Correlation post-estimation

Version 0.2.0

Date 2020-09-03

Author Jordan Adamson [aut, cre]

Maintainer Jordan Adamson <jordan.m.adamson@gmail.com>

Description A computationally efficient way to calculate covariance matrices that are corrected for spatial and temporal correlation.

License MIT + file LICENSE

URL <https://github.com/Jadamso/STrollR>

Encoding UTF-8

LazyData true

Published 2020-09-03

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports Matrix, spam, spam64, raster, sp, spdep

R topics documented:

`bartlettSparse` *Weighting Kernel*

Description

Weighting Kernel

Usage

```
bartlettSparse(d, dmax)
```

Arguments

`d`	distance vector
`dmax`	maximumm distance

Value

vector of bartlett weights

`df2stack` *Convert list of dataframes with to rasterstack*

Description

Convert list of dataframes with to rasterstack

Usage

```
df2stack(sim_i, DF)
```

Arguments

`sim_i`	which simulation
`DF`	list of dataframe

Value

rasterstack

KNN *K nearest neighbours Calculate the number of neighbours within a neighbourhood.*

Description

K nearest neighbours Calculate the number of neighbours within a neighbourhood.

Usage

```
KNN(w, h = w, type = "Moore")
```

Arguments

w	number of neighbours wide (east-west).
h	number of neighbours long (north-south).
type	type of neighbourhood; "Moore" or "VonNeumann"

Value

the number of nearest neighbours

Examples

```
KNN( 4 )
```

make_spacetime_data *Create Multivariate SpaceTime Data*

Description

Create Multivariate SpaceTime Data

Usage

```
make_spacetime_data(
  dimS,
  dimT,
  K,
  xpars = c(0, 1),
  theta = 1:K,
  space_groups = round(sqrt(dimS)),
  time_groups = round(sqrt(dimT)),
  error_type = "distance",
  error_scale = 1,
  xy_mat = expand.grid(Loc_X = seq(0, 1, length.out = dimS), Loc_Y = seq(1, 0,
    length.out = dimS)),
  ws_mat = exp(-1 * as.matrix(dist(xy_mat))),
  wt_mat = toeplitz((2/3)^(0:(dimT - 1))),
  wst_mat = kronecker(wt_mat, ws_mat),
  verbose = F
)
```

Arguments

dimS,	dimT spatial dimension and temporal dimension
K	number of covariates
xpars	generate each Xvariable from uniform(xpar1,xpar2)
theta	parameter vector for Y=X%*%theta
space_groups,	time_groups number of contiguous groups (e.g. countries, time-regimes)
error_type	c('focal', 'distance')
error_scale	variance of errors (normally distributed)
xy_mat	matrix specifing xy locations
ws_mat	matrix specifing correlation structer of error_type='distance'
wt_mat	matrix specifing correlation structer of error_type='distance'
wst_mat=kronecker(wt_mat,	ws_mat)

Details

'focal' creates errors based on $e = u + \rho v$ 'distance' creates errors base on drawing e from a multivariate normal with covariance matrix wst_mat

Value

dataframe

Examples

```
DFst <- make_spacetime_data(11,6,2)
DFst[DFst$Space_ID==1,]
```

---

make_space_data *make_space_data*

Description

Create Multivariate Spatial Data

Usage

```
make_space_data(
  dimS,
  K,
  t_id = NA,
  space_groups = round(sqrt(dimS)),
  xpars = c(0, 1),
  theta = 1:K,
  error_type = "focal",
  error_scale = 1,
  sar_factor = 0.5,
```

```
 wf_mat = rbind(rep(sar_factor, 3), c(sar_factor, 0, sar_factor), rep(sar_factor, 3)),
  xy_mat = expand.grid(Loc_X = seq(0, 1, length.out = dimS), Loc_Y = seq(1, 0,
    length.out = dimS)),
  ws_mat = exp(-as.matrix(dist(xy_mat)))
)

make_space_data.raster(
  dimS,
  K,
  t_id = NA,
  space_groups = round(sqrt(dimS)),
  xpars = c(0, 1),
  theta = 1:K,
  error_type = "focal",
  error_scale = 1,
  sar_factor = 0.5,
 wf_mat = rbind(rep(sar_factor, 3), c(sar_factor, 0, sar_factor), rep(sar_factor, 3)),
  xy_mat = expand.grid(Loc_X = seq(0, 1, length.out = dimS), Loc_Y = seq(1, 0,
    length.out = dimS)),
  ws_mat = exp(-as.matrix(dist(xy_mat)))
)
```

Arguments

dimS	spatial dimension
K	number of covariates
t_id	Time ID
space_groups	number of contiguous groups (e.g. countries)
xpars	generate each Xvariable from uniform(xpar1,xpar2)
theta	parameter vector for Y=X%*%theta
error_type	c('focal', 'distance', 'spherical')
error_scale	variance of errors (normally distributed)
sar_factor	parameter for error_type='focal'
wf_mat	matrix specifing weights to smooth when error_type='focal'
xy_mat	matrix specifing xy locations
ws_mat	matrix specifing correlation structer of error_type='distance'

Details

'focal' creates errors based on $e = u + \rho v$ 'distance' creates errors base on drawing e from a multivariate normal with covariance matrix ws_mat

Value

dataframe

Functions

- `make_space_data.raster`: make_space_data.df uses raster functions instead of matrices (primarily for transparent debugging)

Examples

```
set.seed(3)
DFs  <- make_space_data.raster(10,2,error_type='distance')
set.seed(3)
DFs2 <- make_space_data(10,2,error_type='distance')
```

`make_time_data` *Create Multivariate Temporal Data*

Description

Create Multivariate Temporal Data

Usage

```
make_time_data(
  dimT,
  K,
  s_id = NA,
  time_groups = round(sqrt(dimT)),
  xpars = c(0, 1),
  theta = 1:K,
  error_type = "focal",
  error_scale = 1,
  ar_factor = 2/3,
  wt_mat = toeplitz(c(1, ar_factor, ar_factor^2, rep(0, dimT - 3))),
  xy_mat = data.frame(Loc_X = NA, Loc_Y = NA)
)
```

Arguments

`dimT`	temporal dimension
`K`	number of covariates
`s_id`	Spatial ID
`time_groups`	number of contiguous groups (e.g. time-regimes)
`xpars`	generate each Xvariable from uniform(xpar1,xpar2)
`theta`	parameter vector for Y=X%*%theta
`error_type`	c('focal', 'distance', 'spherical')
`error_scale`	variance of errors (normally distributed)
`ar_factor`	parameter for error_type='focal'
`wt_mat`	matrix specifing correlation structer of error_type='distance'
`xy_mat`	matrix specifing xy locations (defaults to NA)

Details

'focal' creates errors based on $e = u + \rho v$ 'distance' creates errors base on drawing e from a multivariate normal with covariance matrix wt_mat

Value

dataframe

Examples

```
DFt  <- make_time_data(6,2)
```

---

mkGif *Create Gifs*

Description

Create Gifs

Usage

```
mkGif(
  DFlist,
  ti,
  fdir = "~/Desktop/Packages/STrollR/STsim/",
  pname = "STvarX",
  ind = 1,
  vw = FALSE
)
```

Arguments

DFlist	
ti	number of time periods
fdir, pname	directory and file
ind	which simulation
vw	view output

Value

list of rasterstacks

---

model.frame.i *Model Frame with Regression Intercept*

Description

Model Frame with Regression Intercept

Usage

```
## S3 method for class 'i'
model.frame(reg, check_int = T)
```

NEIGH *Calculate the weights objects*

Description

Calculate the weights objects

Usage

```
NEIGH(
  coord_sp,
  neigh = 1,
  knn = TRUE,
  adj = FALSE,
  dnn = FALSE,
  rast = FALSE,
  vario = FALSE,
  tracer = TRUE,
  tr_type = "mult",
  tr_m = 20,
  tr_p = 16,
  symm = TRUE,
  symm_check = TRUE,
  SAVE = NA,
  write_gwt = F
)
```

Arguments

coord_sp	matrix of coordinates or a SpatialPoints object
neigh	number of neighbours to use in calculation
knn	calculate weights using knn approach
adj	calculate vonneumann weights (see VonNeumann)
dnn	dnn approach unsupported
rast	raster approach unsupported
vario	is coord_sp a weights matrix?
tracer	create trace matrix objects?
tr_type	type of trace matrix
tr_m	trace matrix m
tr_p	trace matrix p
symm	make weights symmetric
symm_check	check for symmetric weights matrix
SAVE	filename to save to, NA (default) returns as object
write_gwt	create GWT objects used in spdep or sphet

Value

filename of saved objects, or returns objects if SAVE=NA

sim2stack *Convert simulation to rasterstack*

Description

Convert simulation to rasterstack

Usage

```
sim2stack(e_spt, nsim, xyt)
```

Arguments

e_spt	matrix of draws from spam.mvtnorm (each row a realization of a simulation)
xyt	lattice structure
number	of simulations

Value

list of rasterstacks

var2stack *Convert Dataframe with 1 variable to raster for one realization*

Description

Convert Dataframe with 1 variable to raster for one realization

Usage

```
var2stack(df_i, sim_i)
```

Arguments

df_i	dataframe
sim_i	which simulation

Value

raster

varioJ *Variogram Calculation*

Description

Variogram Calculation

Usage

```
varioJ(
  coords,
  cuttoff,
  E,
  latlon = FALSE,
  indices = FALSE,
  clean = FALSE,
  verbose = FALSE
)
```

Arguments

coords	coordinate matrix
cuttoff	cutoff from which to calculate variogram
E	vector of values (i.e. OLS residuals) associated coords
latlon	coordinates are lon,lat or x,y
indices	return indices?
clean	unused currently
verbose	

Value

data.frame of dij and (ei-ej)^2

vcovSHAC *Calculate a SHAC (Spatial Heteroskedastic and Autocorrelation Consistent) Variance Covariance Matrix*

Description

Calculate a SHAC (Spatial Heteroskedastic and Autocorrelation Consistent) Variance Covariance Matrix

Usage

```
vcovSHAC(
  reg,
  wmat,
  method = "bruteforce_ll",
  cutoff_s,
  loc_lat,
  loc_lon,
  loc_y,
  loc_x,
  add_hc = T,
  add_cluster = F,
  add_hac = F,
  verbose = FALSE,
  manual_dist = T,
  ...
)

vcovSHACsep(
  reg,
  wmat,
  method = "bruteforce_llt",
  cutoff_s,
  cutoff_t,
  loc_lat,
  loc_lon,
  loc_y,
  loc_x,
  loc_t,
  add_hc = T,
  add_cluster = F,
  add_hac = F,
  verbose = FALSE,
  manual_dist = T,
  ...
)

vcovSTHAC(
  reg,
  wmat,
  method = "bruteforce_llt",
  cutoff_s,
  cutoff_t,
  loc_lat,
  loc_lon,
  loc_y,
  loc_x,
  loc_t,
  add_hc = T,
  add_cluster = F,
  add_hac = F,
  verbose = FALSE,
```

```
  manual_dist = T,
  ...
)
```

Arguments

reg	an 'lm' object
wmat	weights matrix
method	c(rolled', semirolled', 'bruteforce_ll', 'bruteforce_xy')
cutoff_s	include weights up to cutoff_s. Required to be in km if method='bruteforce_ll' or in map-units if 'bruteforce_xy'.
loc_lat, loc_lon	required if method='bruteforce_ll'
loc_y, loc_x	required if method='bruteforce_xy'
add_hc, add_cluster, add_hac	logical for adding HC correction (default=T), Cluster correction (default=F), HAC correction (default=F)
verbose	show messages
options	passed to sandwich

Value

covariance matrix

Functions

- vcovSHACsep: similar to vcovSTHAC, but treats spatial autocorrelation seperately within each time period
- vcovSTHAC: Space and Time HAC. Also weights the time-dimension according to bartlett kernel (i.e., weight= K(space)*K(time)) for bartlett-kernel K.

VonNeumann *Compute VonNeumann Neighbours*

Description

Compute VonNeumann Neighbours

Usage

```
VonNeumann(coord_sp, directions = 4)
```

Arguments

coord_sp	SpatialPoints object or coordinate-matrix
directions	see adjacent

Value

sparse weights matrix

weight_mat *Compute Sparse Spatial Weights Matrix*

Description

Compute Sparse Spatial Weights Matrix

Usage

```
weight_mat(
  XY_Mat,
  latlon = FALSE,
  cutoff_s,
  cutoff_km2angles = FALSE,
  cutoff_angles2km = FALSE,
  verbose = F
)

weight_mat.df(
  DFs,
  xy_names = c("Loc_X", "Loc_Y"),
  latlon = FALSE,
  cutoff_s,
  cutoff_km2angles = FALSE,
  cutoff_angles2km = FALSE,
  verbose = F
)
```

Arguments

XY_Mat	matrix of coordinates (either lat,lon or x,y).
latlon	does XY_Mat have lat,lon coordinates?
cutoff_s	use distances up cutoff_s. (if using generic map units, must specify latlon=cutoff_km2angles=cutoff_a
cutoff_km2angles,	
	cutoff_angles2km convert cutoff_s from km to angles or vice-versa?
verbose	print messages

Value

the number of nearest neighbours

Functions

- `weight_mat.df`: weight_mat.df is a wrapper of weight_mat for dataframes

Examples

```
weight_mat(expand.grid( list(x=1:10, y=1:10)), cutoff_s=.5)
```

XOmegaX *XOmegaX Meat Matrix Calculations*

Description

XOmegaX Meat Matrix Calculations

Usage

```
XOmegaX_rolled(X, WMAT, E)

XOmegaX_semirolled(X, WMAT, E)

XOmegaX_bruteforce_ll(X, E, loc_lat, loc_lon, cutoff_s, manual_dist = T)

XOmegaX_bruteforce_xy(X, E, loc_y, loc_x, cutoff_s, manual_dist = T)

XOmegaX_bruteforce_llt(
  X,
  E,
  loc_lat,
  loc_lon,
  loc_t,
  cutoff_s,
  cutoff_t,
  manual_dist = T
)

XOmegaX_bruteforce_xyt(
  X,
  E,
  loc_y,
  loc_x,
  loc_t,
  cutoff_s,
  cutoff_t,
  manual_dist = T
)
```

Arguments

X	design matrix
WMAT	weighting matrix (preferably sparse sparse)
E	vector of residuals
loc_lat,	loc_lon spatial-coordinate vectors
cutoff_s	how far to extend bartlett kernel in spatial dimention (in km)
loc_y,	loc_x spatial-coordinate vectors
loc_t	time-coordinate vector
cutoff_t	how far to extend bartlett kernel in time dimension (in time units)

Details

XOmegaX_rolled Fully is for Sparse-Matrix Meat-Matrix Calculation (Whole Matrix At Once) XOmegaX_semirolled is for Sparse-Matrix Meat-Matrix Calculation (One Row At A Time) XOmegaX_bruteforce_ll is for latlon data. XOmegaX_bruteforce_llt is for latlon and time data. XOmegaX_bruteforce_xy is for projected data. XOmegaX_bruteforce_xyt is for projected and time data. XOmegaX_bruteforce_*t weights the time-dimension according to bartlett kernel

Value

object to be used in vcov* functions

Index