PARTS Simulations

Clay Morrow

3/10/2021

require("mvtnorm")  
library(dplyr)  
library(ggplot2)

# Reworked models

The overall response will be simulated with the following process:

$$ x\_{i}(t) = \beta\_{k, i}u\_{k, i} + (\phi\_{m, i}u\_{m,i})t + \gamma\_{k, i} w\_{k, i}(t) + \varepsilon\_{i}(t) \tag{1} $$

Where:

* is the variable of interest (e.g. NDVI) in pixel ;
* is a model matrix of spatially variable predictors (e.g. landclass);
* are spatio-temporally variable predictors (e.g. temperature);
* , , and are the effects of the predictors. Each of these can be fixed OR randomly distributed in space (see below); and
* is the spatio-temporally correlated error term.

(and ) is generated with an ARMA process:

$$\varepsilon\_{i}(t) = \delta\_{i}(t) + \sum\_{j = 1}^{p}{\rho\_{j}\varepsilon\_{i}(t)} + \sum\_{j = 1}^{q}{\theta\_{j}\varepsilon\_{i}(t)} \tag{2}$$

where:

* is the AR(p) process;
* $\sum\_{j = 1}^{q}{\theta\_{j}\varepsilon\_{i}(t)$ is the MA(q) process; and
* are spatially correlated errors;

Spatial correlation in the ARMA terms is generated with an exponential power function:

$$\Sigma\_{\delta\_{i}} = I\eta\_i + (\sigma^{2} - \mu\_{\eta\_i})exp{(-d\_{i, j}/r)^a} \tag{3}$$

where:

* is an Identity matrix multiplied by the nugget , which may be fixed OR variable in space: ;
* is the sill (variance) of (I’ll stick with unit variance);
* is the distance between pixels and ;
* is the range of the spatial auto-correlation (in pixels); and
* is the shape parameter for the auto-correlation function

Other spatial correlations will be generated the Matern Cross-covariance function:

$$R(d\_{i, j}; v, c) = 2^{1-v} \Gamma(v)^{-1}(c||d\_{i, j}||)^{v} K\_{v}(c||d\_{i, j}||) \tag{4}$$

where :

* is the scale parameter (similar to range )
* is the smoothness parameter. When , is equivalent to and
* is the gamma function, and
* is a modified Bessel function of the second kind

**Note:** In practice, this function is very useful for generating smooth spatial correlations and waves (see $\S$ “Spatially variable parameter” below). But may need some help understanding the math and what it’s actually doing! It may be worth while to also generate with Matern correlations?

# Simulation Plans

Clay, sorry about this, but I think you are going to have to do it differently. I think you are going to have to use much bigger maps, say 10000, 20000, 40000, and 80000 pixels, and do the simulations using partitions (maybe 2000 pixels each). In other words, you are going to have to use datasets that require PARTS in order to demonstrate its robustness. This means two things:

1. You are going to have to be much more selective in the cases you consider for simulations.

2. You can't use the Matern matricies. Instead, for the residuals, I think you'll have to taper. For the coefficients, I'd use a 2D sine wave.

I haven't thought this through in much detail, but this is my starting list for what I would do:

x(t) = b1u1 + b2u2 + b3u3 + (p1u1+p2u2+p3u4)t + (g1+g2u5)w1(t)+ e(t)

u1-u2: land classes, set up in a regular grid, with squares of cover classes varying in size (1, 4, 16, ... pixels)

u3-u5: unmeasured variables given by a 2D sine wave plus spatially uncorrelated random variables -- These unmeasured variables are used to make the coefficients themselves random; this simplifies the model

All coefficients constant: b1, b2, b3, p1, p2, p3, g1, g2

The errors are ARMA(1,1), like you have. Because the analyses of the NDVI data show spatial structure in rho, I would let rho and theta following 2D sine waves.

For delta in the ARMA, I would use a tapered matrix (so you can simulate a 80000 pixel map)

ada and range are fixed.

Simulation studies:

1. For T=1 (i.e., only spatial data), make power curves for maps of 10000, 20000, 40000, and 80000 pixels over values of b1, testing H0:b1=0 and H0:b1=b2. Also plot the estimates of b1 with standard errors. From then on just use maps with 10000 pixels.

2. For T=30, do power curves testing H0:p1=0 and H0:p1=p2 while varying the size of the patches of u1 and u2.

3. For T=30, do power curves testing H0:g1=0 while setting p1=p2=p3=0 and varying g2 (assuming the spatial mean of u5 is zero)

4. Do a table of type I error rates H0:p1=0 and H0:p1=p2 to look at the effects of

i. w1

ii. u4

iii. u3

iv. range

v. spatial structure in rho

## Data Generation

In order to rigorously test the robustness of the PARTS method, we will simulate 1000 sets of predictor variables and and varying the effects that each of those sets have on . Therfore, will be simulated for each unique set of effect parameter combinations. The map for all simulations will be the same size: pixels (or some other size, dependant on speed). will contain the Euclidean distance between points and will be constant throughout the simulations.

will represent land cover class designations, with 4 levels (), coded as dummy variables (i.e. ). It will be generating by discretizing (by quartiles) a continuous variable produced with the Matern covariance function. and will be selected to generate a desired spatial pattern and will be constant throughout the simulations. will have no nugget and unit variance (i.e. , ).

will be a single () spatio-temporal variable representing temperature. It will be generated as an ARMA(1, 1) model with Matern covariance structure. and will be selected to generate a desired spatial pattern (different from ) and will be constant throughout the simulations. and will also be predetermined and constant. will have unit variance. will have no nugget and unit variance (i.e. , ).

will then be generated, for 30 time points (), by varying the following parameters on each of the 1000 data sets. This will allow us to compare the effect of the parameter independent of the specific predictor values.

Effects parameters , , and , will take on one of 3 levels: 2 fixed values (i.e. ) and one spatially distributed random variable (i.e. ). The AR parameter will also have 2 levels (e.g.  and ), the MA parameter will have 2 levels (e.g ), the spatial nugget will have 2 levels (i.e. ), and the spatial correlation range parameter will have 2 levels (i.e. ). The spatial correlation shape parameter will remain constant througout the simulations.

With this specification there are 432 unique parameter combinations which would yield 432,000 simulations of . It may be best to also hold constant or at least restrict it to 2 fixed values, since we aren’t actually interested in our estimate of an the intercept parameter. Doing so would reduce the total simulations to 72,000. It may also be acceptable to hold constant, further reducing the total to 36,000 simulations. But, we may also decide to add a level of that is spatially variable.

## Model Fitting

For each simulation of , we will apply the PARTS method. We will use AR\_REML to fit a time series to each pixel and then will fit a GLS to slope parameters for the effect of time. We will then compare the overall estimate of the effect of time to the true values . If was spatially variable, we will compare to the mean . We will also estimate type I error rates for the case when .

## Example Code

This is some code I ran to test out the spatial correlation functions

First we’ll set the initial parameters:

#---  
## Initial Parameters  
#---  
  
set.seed(3654)  
  
# Sim pars ----  
burn.in = 0  
sim.iters = 1  
  
# Map pars ----  
map.cols = 20  
map.rows = 10  
npix = map.rows \* map.cols # should be a square  
  
# ARMA pars ----  
rhos = c(0.6, 0.2)# AR parameter(s): must sum to less than 1  
thetas = c(.4) # MA parameter: must be less than 1  
  
# Temporal pars ----  
ntime = 30 # number of time points  
  
# varcov pars ----  
cormod = "stable" # 'stable' correlation model: varcov = nug+sill\*exp(-(d/scale)^shape)  
dist.meth = "Eucl" # Euclidean distance  
sill = 1 # sill (variance)  
r = .25 # range of spatial autocor, relative to width of map (scale parameter)  
a = 1 # shape of spatial autocor function (shape parameter)  
nug = .2 # fixed nugget  
  
# spatial nugget pars ----  
nug.cormod = "matern"  
nug.distmeth = "Eucl"  
nug.r = .2  
nug.s = 20  
constrain.nug = c(0, 1) # constrain the spatial nugget to these values  
  
# effect parameters ----  
beta = .2 # spatial predictors  
phi = .2 # spatio-temporal predictors  
gamma = .5 # time trends  
  
# Additinal parameter calculations ----  
  
## Assign coordinates to pixels  
coords = as.matrix(expand.grid(x = as.double(1:map.cols),   
 y = as.double(1:map.rows)))  
  
## spatial correlation  
psill = sill - nug # partial sill: if nug is spatially variable, use mean(nug)  
range = r \* map.cols # range in pixels  
  
## spatially variable parameters  
nug.range = nug.r \* map.cols  
nug.smooth = nug.s  
  
  
# Sanity checks ----  
stopifnot(sum(rhos) < 1)  
stopifnot(nrow(coords) == npix)  
  
# parameter table ----  
  
fixed.pars = expand.grid(range = range, a = a, nug = nug, fixed.pars = TRUE)  
  
  
corpars = list(scale = range, power = a, # parameter list for Covmatrix()  
 sill = psill, nugget = nug, mean = 0)  
nugpars = list(scale = nug.range, smooth = nug.smooth,   
 mean = 0, sill = 1, nugget = 0)

Then:

1. Calculate the spatial covariance matrix with the function Covmatrix from package CompRandFld.

# Calculate varcov matrix (Sigma) ----  
## Covmatrix() - easier to modify specs  
covmat = CompRandFld::Covmatrix(coordx = coords,   
 distance = dist.meth,  
 corrmodel = cormod,  
 param = corpars)$covmatrix  
# invcholV = remotePARTS::invert\_chol(covmat)  
  
# ## Tony's  
# d = as.matrix(stats::dist(coords, method = "euc"))  
# d.scale = (d - min(d)) / (max(d) - min(d)) # rescale to (0, 1)  
# covmat2 = diag(rep(nug, npix)) + (1 - nug) \* exp(-(d/r)^a)  
# invcholV2 = remotePARTS::invert\_chol(covmat2)  
# ## Same answer?  
# all((covmat2 - covmat) == 0) # TRUE! - also no speed change

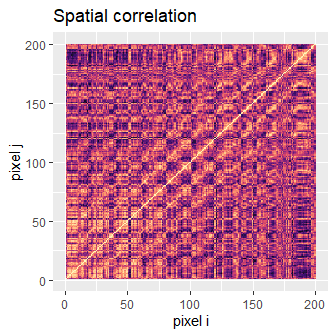
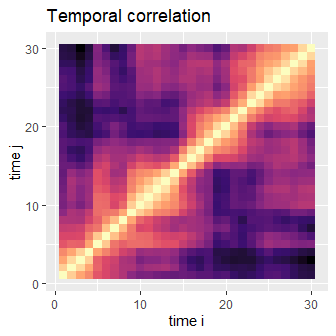
1. generate with rmvmnorm().

# Generate spatially correlated errors (delta) ----  
del = mvtnorm::rmvnorm(n = ntime, sigma = covmat, method = "chol")

1. generate .

# Generate ARMA errors (epsilon) ----  
## apply arima.sim() to each column of del  
eps = t(apply(X = del, MARGIN = 2,   
 function(e){  
 arima.sim(n = ntime,   
 model = list(ar = rhos, ma = thetas),   
 innov = e)  
 }))

Let’s visualize the correlations among in space and time:

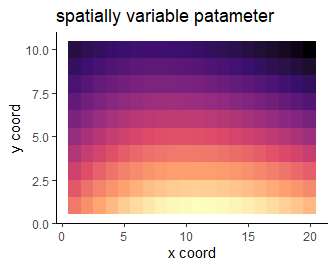


1. Add the errors to the rest of the sim model

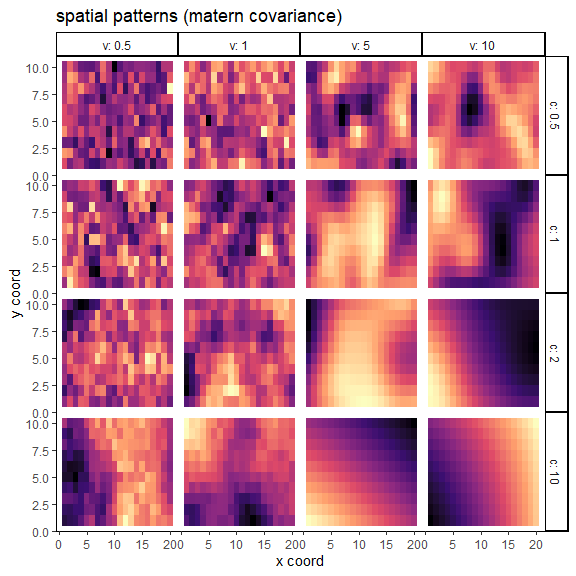
# beta, phi, gamma: user defined effects parameters (possibly spatially variable)  
# u: spatially generated predictors (via mattern function)  
# m: spatio-temporally generated predictors (mattern & ARMA functions)  
  
x = beta\*u + phi\*u\*time + gamma\*w + eps   
  
#Note: proper matrix multiplication will be used

# Spatially variability: Matern Covariance

To demonstrate how we can simulate a waved spatial pattern in our predictors and effects parameters, I’ll now generate a spatially variable nugget using Matern cross-covariance. When and are both large, you get a wave across the map:

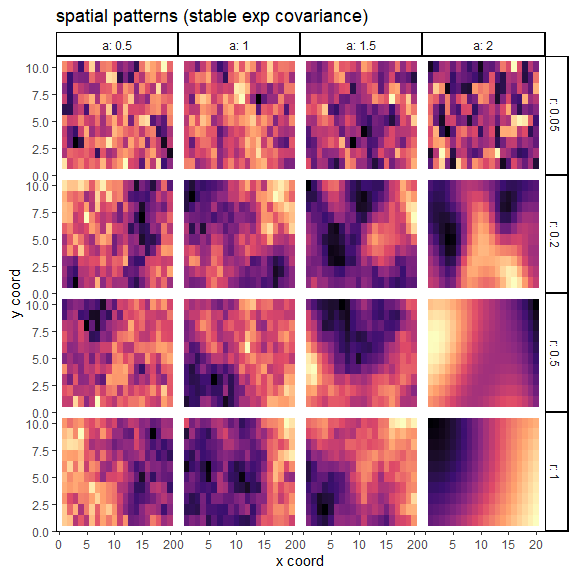


The Matern function is ideal for simulating spatial variables and spatial parameters because the resultant patterns can be **substantially** changed by varying and . To demonstrate this, I’ll vary and . and



whereas the stable exponential function has more limited range (). Though, the Gaussian version () can generate interesting and realistic patterns:

r.tmp = c(.05, 0.2, .5, 1)\*map.cols  
  
a.tmp = c(.5, 1, 1.5, 2)  
  
spatcor.tab = data.frame(NULL)  
  
for (a in a.tmp) for (r in r.tmp){  
   
 nugpars = list(scale = r, power = a,   
 mean = 0, sill = 1, nugget = 0)  
   
 nug.vcov = CompRandFld::Covmatrix(coords,   
 distance = dist.meth,   
 corrmodel = "stable",   
 param = nugpars)$covmatrix  
 # image(nug.vcov)  
 spat.nug <- scales::rescale(mvtnorm::rmvnorm(n = 1, sigma = nug.vcov), to = constrain.nug)  
 tmp = data.frame(coords, nug = as.vector(spat.nug), r = r/map.cols, a = a)  
 spatcor.tab = rbind(spatcor.tab, tmp)  
   
 # nug.corpars = list(scale = range, power = a, sill = psill, nugget = 0, mean = 0)  
 # covmat = CompRandFld::Covmatrix(coordx = coords,   
 # distance = dist.meth,  
 # corrmodel = cormod,  
 # param = nug.corpars)$covmatrix  
 #   
 # covmat = diag(as.vector(spat.nug)) + (1-mean(as.vector(spat.nug)))\*covmat  
 # del = mvtnorm::rmvnorm(n = ntime, sigma = covmat, method = "chol")  
 # eps = t(apply(X = del, MARGIN = 2,   
 # function(e){  
 # arima.sim(n = ntime,   
 # model = list(ar = rhos, ma = thetas),   
 # innov = e)  
 # }))  
   
}  
  
ggplot(data = spatcor.tab, aes(x = x, y = y, fill = nug)) +  
 theme\_classic() +   
 theme(legend.position = "none",  
 panel.spacing = unit(0, "lines")) +  
 geom\_tile() +   
 scale\_fill\_viridis\_c(option = "magma") +  
 facet\_grid(r ~ a, labeller = "label\_both") +  
 labs(x = "x coord", y = "y coord", title = "spatial patterns (stable exp covariance)")



The matern covariance structure seems to have been developed to generate more bioligically realistic spatial patterns.