

Spatial model

James T. Thorson

```
library(tinyVAST)
library(pdp) # approx = TRUE gives effects for average of other covariates
library(lattice)
library(visreg)
library(fmesher)
set.seed(101)
```

tinyVAST is an R package for fitting vector autoregressive spatio-temporal (VAST) models using a minimal and user-friendly interface. We here show how it can fit spatial autoregressive model

```
# Simulate
n_x = n_y = 25
n_t = 10
R = exp(-0.4 * abs(outer(1:n_x, 1:n_y, FUN="-"))) )
z = mvtnorm::rmvnorm(1, sigma=kronecker(R,R) )
t = sample(1:n_t, replace=TRUE, size=length(z))
Data = data.frame( expand.grid(x=1:n_x, y=1:n_y), t=t, z=as.vector(z) + cos(t/n_t*2*pi))
Data$n = Data$z + rnorm(nrow(Data), sd=1)

# Add columns for multivariate and temporal dimensions
Data$time = 1
Data$var = "n"

# make mesh
mesh = fm_mesh_2d( Data[,c('x','y')], n=100 )

# fit model
out = fit( data = Data,
           formula = n ~ s(t),
           spatial_graph = mesh,
           quiet = TRUE,
           sem = "" )
```

We can then calculate the area-weighted total abundance:

```
# Predicted sample-weighted total
integrate_output(out)
#>           Estimate           Std. Error Est. (bias.correct) Std. (bias.correct)
#>      -104.15032           25.99485      -104.15032              NA
# integrate_output(out, apply.epsilon=TRUE )

# True (latent) sample-weighted total
sum( Data$z )
#> [1] -92.89517
```

Percent deviance explained

We can compute deviance residuals and percent-deviance explained:

```
R1 = sum( residuals(out, type="deviance")^2 )

# tinyVAST null model with just a single intercept
null = fit( data = Data, formula = n ~ 1, quiet = TRUE )
R0 = sum( residuals(null, type="deviance")^2 )

# Percent deviance explained
1 - R1/R0
#> [1] 0.7063608
```

We can then compare this with the PDE reported by mgcv

```
mygam = gam( n ~ s(t) + s(x,y), data=Data ) #
summary(mygam)$dev.expl
#> [1] 0.3517756
```

where this comparison shows that using the SPDE method in tinyVAST results in higher percent-deviance-explained. This reduced performance for splines relative to the SPDE method presumably arises due to the reduced rank of the spline basis expansion, and the better match for the Matern function (in the SPDE method) relative to the true (simulated) exponential semivariogram.

It is then easy to confirm that mgcv and tinyVAST give (essentially) identical PDE when excluding the spatial term from each.

```
out_reduced = fit( data = Data, formula = n ~ s(t), quiet = TRUE, sem = "" )
R1_reduced = sum( residuals(out_reduced, type="deviance")^2 )
1 - R1_reduced/R0
#> [1] 0.1721612

#
mygam_reduced = gam( n ~ s(t), data=Data ) #
summary(mygam_reduced)$dev.expl
#> [1] 0.1713967
```

Visualize spatial response

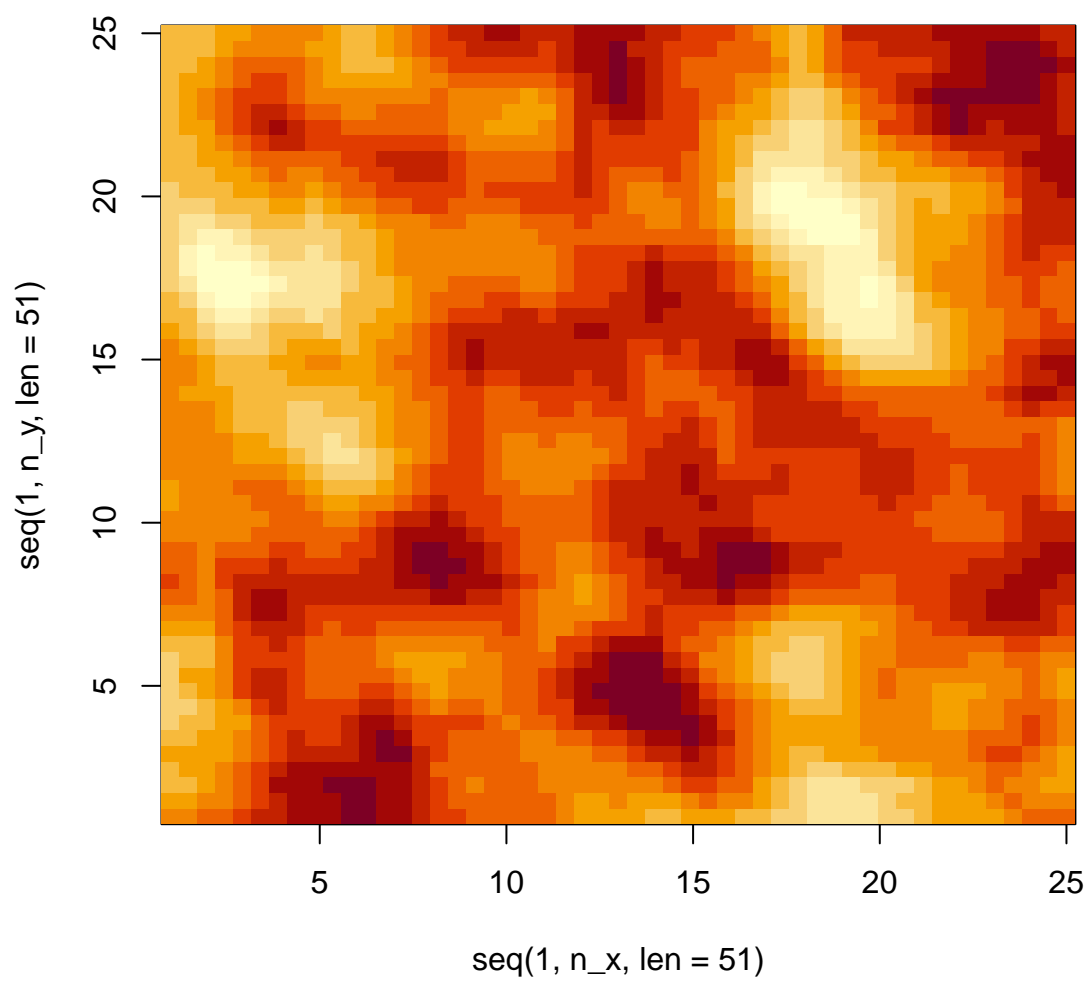
tinyVAST then has a standard predict function:

```
predict(out, newdata=data.frame(x=1, y=1, time=1, t=1, var="n") )
#> [1] 0.2978245
```

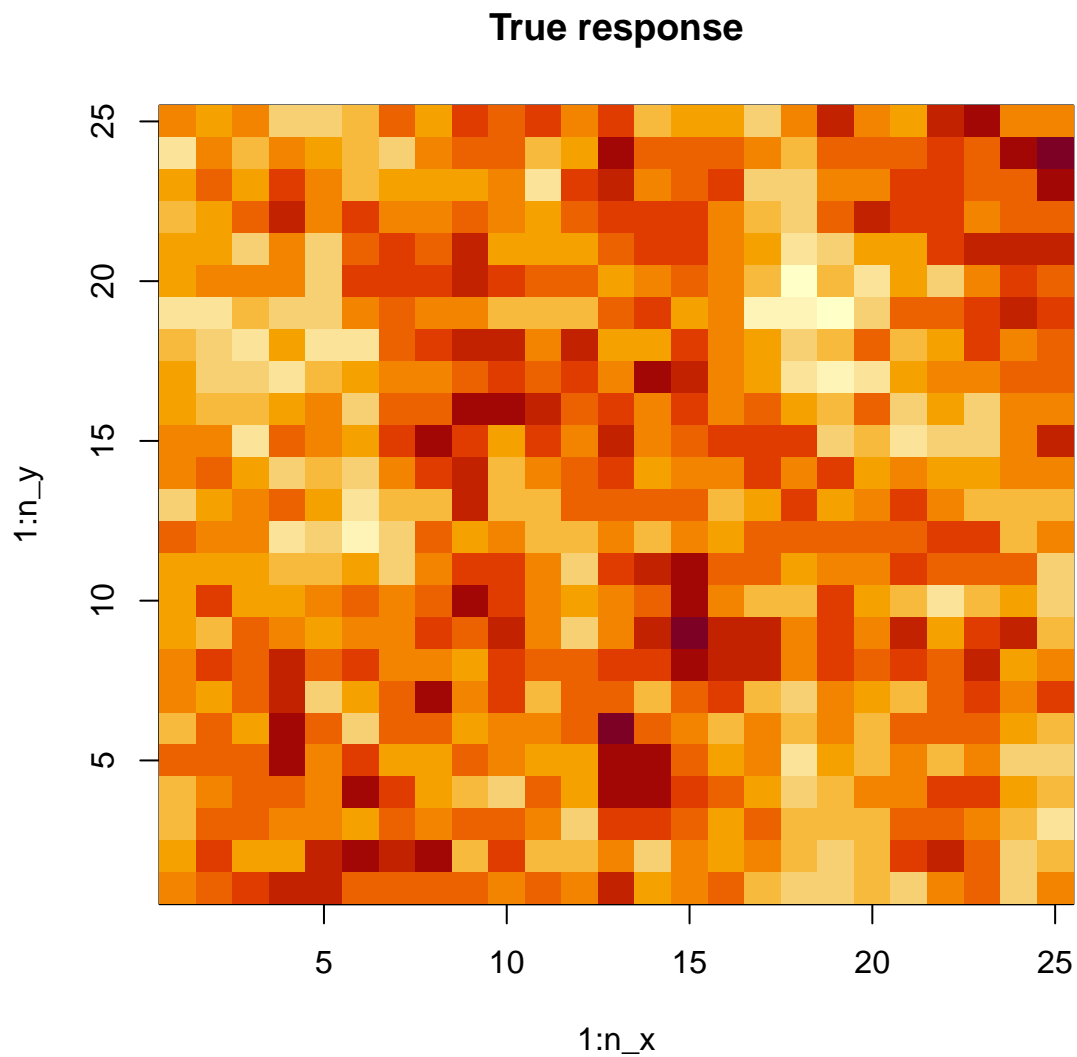
and this is used to compute the spatial response

```
# Prediction grid
pred = outer( seq(1,n_x,len=51),
              seq(1,n_y,len=51),
              FUN=\(x,y) predict(out,newdata=data.frame(x=x,y=y,t=1,time=1,var="n")) )
image( x=seq(1,n_x,len=51), y=seq(1,n_y,len=51), z=pred, main="Predicted response" )
```

Predicted response



```
# True value  
image( x=1:n_x, y=1:n_y, z=matrix(Data$z,ncol=n_y), main="True response" )
```



We can also compute the marginal time effect

```
# compute partial dependence plot
Partial = partial( object = out,
  pred.var = "t",
  pred.fun = \(object,newdata) predict(object,newdata),
  train = Data,
  approx = TRUE )

# Lattice plots as default option
plotPartial( Partial )
```

