Comparison with mgcv

James T. Thorson

```
library(tinyVAST)
library(pdp)  # approx = TRUE gives effects for average of other covariates
library(lattice)
library(visreg)
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 replicate analysis using splines specified via `mgcv`

```
# Simulate
n_obs = 100
x = rnorm(n_obs)
group = sample( x=1:5, size=n_obs, replace=TRUE )
w = runif(n_obs, min=0, max=2)
z = 1 + x^2 + cos((w+group/5)*2*pi) + rnorm(5)[group]
a = exp(0.1*rnorm(n_obs))
y = z + a + rnorm(n_obs, sd=0.2)
Data = data.frame( x=x, y=y, w=w, z=z, group=factor(group), a=a )

# fit model
Formula = y ~ 1 + s(group, bs="re") + poly(x, 2, raw=TRUE) + s(w, by=group, bs="ts") # + offset(a)
myfit = fit( data = Data,
             formula = Formula,
             control = tinyVASTcontrol(quiet = TRUE) )
#>   0:     203.05622:  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
#>   1:     169.01209: 0.135764 -0.249607 0.941036 -0.156186 0.00715342 0.00653215 0.00491299 0.0034276
#>   2:     154.58731: 0.276830 -0.0156281 0.885657 -0.241085 0.0203700 0.0183906 0.0153712 0.00863582
#>   3:     144.64961: 0.301061 -0.0687604 0.755722 -0.252891 0.0171256 0.0202658 0.0101849 -0.0007092
#>   4:     143.67919: 0.329785 -0.0872183 0.830346 -0.268468 0.0171551 0.0236459 0.00967984 -0.005017
#>   5:     143.33998: 0.386681 -0.0334875 0.781206 -0.295537 0.0187298 0.0304190 0.00979083 -0.012795
#>   6:     142.55274: 0.464104 -0.0776590 0.809452 -0.330371 0.0221494 0.0401762 0.0107420 -0.0218526
#>   7:     141.85159: 0.644330 -0.0394940 0.769053 -0.391307 0.0299119 0.0657102 0.0122694 -0.0477634
#>   8:     141.13148:  1.01054 0.0164950 0.875855 -0.363873 0.0525960 0.142024 0.0164551 -0.124308 -0.
#>   9:     139.15342:  1.35393 -0.126483 0.781086 -0.313169 0.0781600 0.218021 0.0221701 -0.197728 -0.
#>  10:     137.66665:  2.18063 -0.130802 0.834568 0.232767 0.164960 0.482018 0.0420680 -0.455735 -0.23
#>  11:     136.88279:  2.17448 -0.0616404 0.756545 0.239204 0.165565 0.484775 0.0419148 -0.459494 -0.2
#>  12:     136.55807:  2.20011 -0.0754085 0.783391 0.310602 0.179581 0.526448 0.0444905 -0.501574 -0.2
#>  13:     136.41381:  2.14163 -0.0728860 0.803487 0.373332 0.189250 0.555457 0.0455661 -0.532239 -0.2
#>  14:     135.60214:  2.30159 -0.0645607 0.812072 0.976197 0.301441 0.899539 0.0622010 -0.892943 -0.5
#>  15:     131.53003:  1.82667 -0.0692446 0.854637  2.29915 0.864818  2.66227 0.109285 -2.85334 -1.988
#>  16:     130.63258:  2.00763 -0.0828194 0.835951  2.26501  1.36485  4.07796 -0.0366790 -4.57120 -3.6
#>  17:     129.56862:  2.09055 -0.0535076 0.803475  1.13845  1.37442  3.67626 -0.342398 -3.30292 -2.59
#>  18:     128.35474:  1.94162 -0.0501979 0.834193  1.41292  1.71426  4.25441 -0.804949 -3.71980 -2.89
#>  19:     126.60005:  1.75628 -0.0510832 0.875651  1.77507  2.59665  5.42472 -2.61730 -4.22690 -3.325
```

```
#>  20:     125.87610:  1.83438 -0.0477974 0.888438  1.58593  3.17972  5.67650 -3.55072 -3.98472 -3.307
#>  21:     125.48636:  2.01608 -0.0488549 0.855907  1.34958  3.51038  5.19044 -2.97446 -3.48515 -2.919
#>  22:     125.14556:  1.97144 -0.0441069 0.858927  1.40540  4.08483  5.33531 -3.30204 -3.69678 -2.963
#>  23:     124.90498:  1.94146 -0.0450360 0.863212  1.46838  4.77623  5.28542 -3.43999 -3.76299 -3.000
#>  24:     124.75722:  1.90886 -0.0407195 0.866271  1.53115  6.33187  5.10622 -3.44618 -3.84081 -3.068
#>  25:     124.68753:  1.91580 -0.0443943 0.867519  1.51318  5.79511  5.03539 -3.27615 -3.78153 -3.035
#>  26:     124.67998:  1.92422 -0.0433796 0.865367  1.50256  5.85842  5.02453 -3.21649 -3.76508 -3.036
#>  27:     124.67927:  1.92574 -0.0440787 0.865318  1.50206  5.90544  5.01432 -3.19095 -3.75707 -3.024
#>  28:     124.67919:  1.92546 -0.0435713 0.865396  1.50268  5.90043  5.02386 -3.19091 -3.75524 -3.029
#>  29:     124.67918:  1.92542 -0.0437228 0.865331  1.50237  5.89826  5.02873 -3.19168 -3.75615 -3.028
#>  30:     124.67918:  1.92534 -0.0437454 0.865392  1.50250  5.89922  5.03149 -3.19141 -3.75591 -3.027
#>  31:     124.67918:  1.92534 -0.0437446 0.865399  1.50250  5.89956  5.03206 -3.19120 -3.75573 -3.027
#>  32:     124.67918:  1.92534 -0.0437416 0.865401  1.50248  5.89957  5.03206 -3.19120 -3.75576 -3.027
```

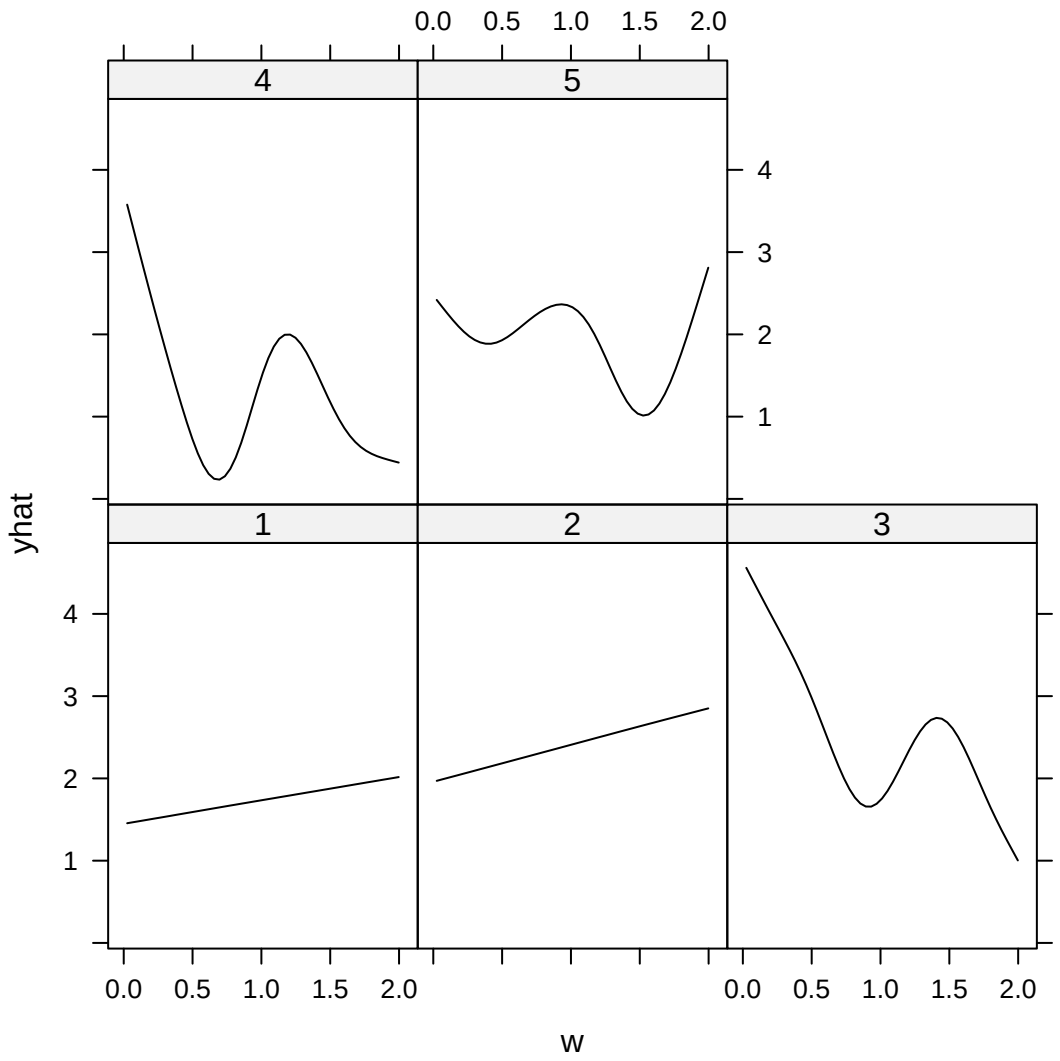
`tinyVAST` then has a standard `predict` function:

```
predict(myfit, newdata=data.frame(x=0, y=1, w=0.4, group=2, a=1) )
#> [1] 2.13754
```

and this is used to compute partial-dependence plots using package `pdp`

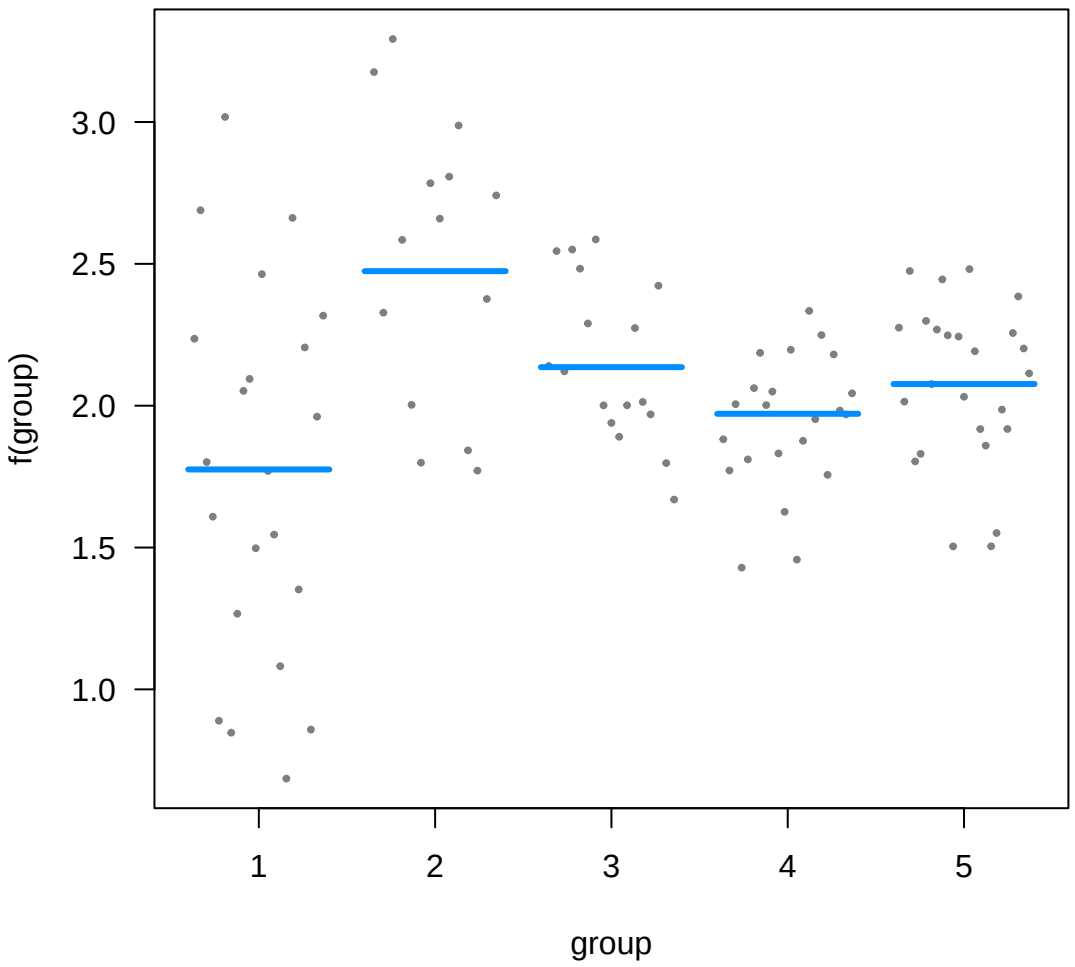
```
# compute partial dependence plot
Partial = partial( object = myfit,
                   pred.var = c("w","group"),
                   pred.fun = \(object,newdata) predict(object,newdata),
                   train = Data,
                   approx = TRUE )

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

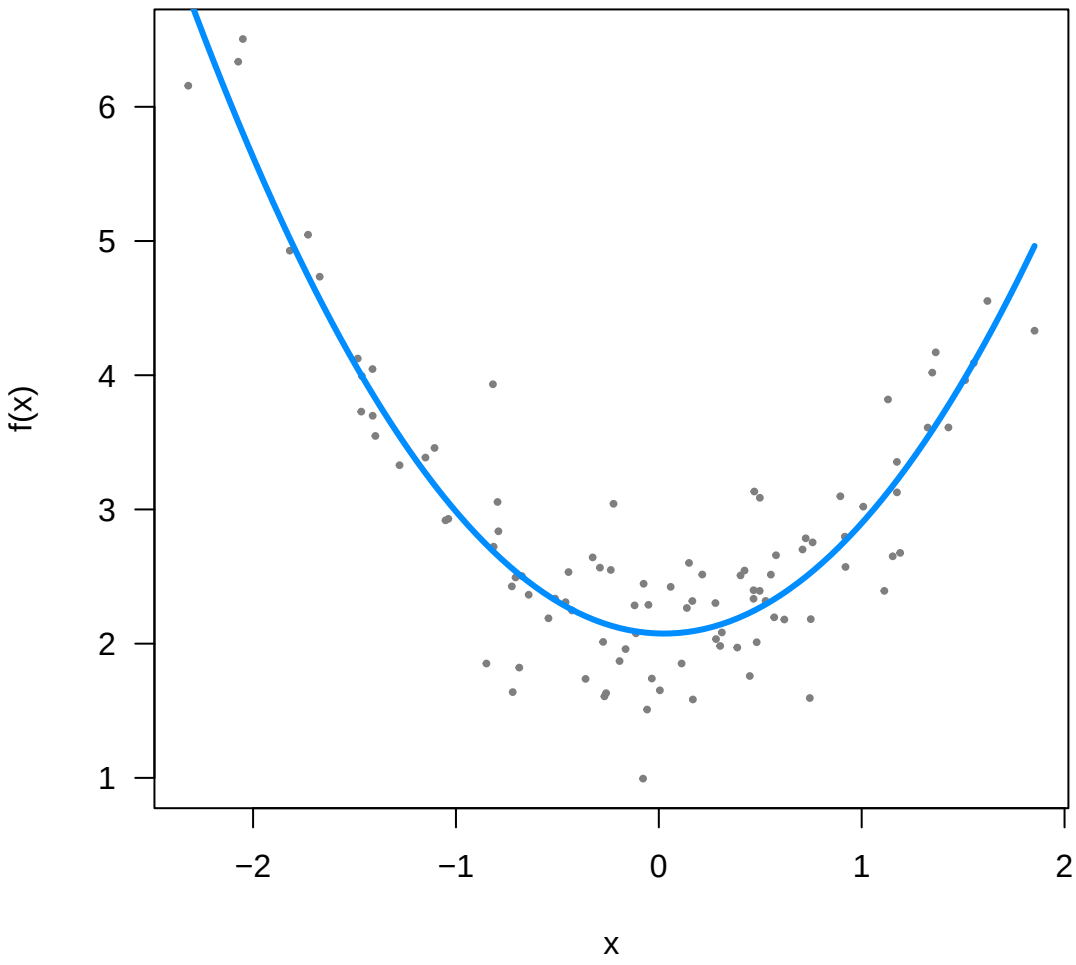


Alternatively, we can use `visreg` to visualize output:

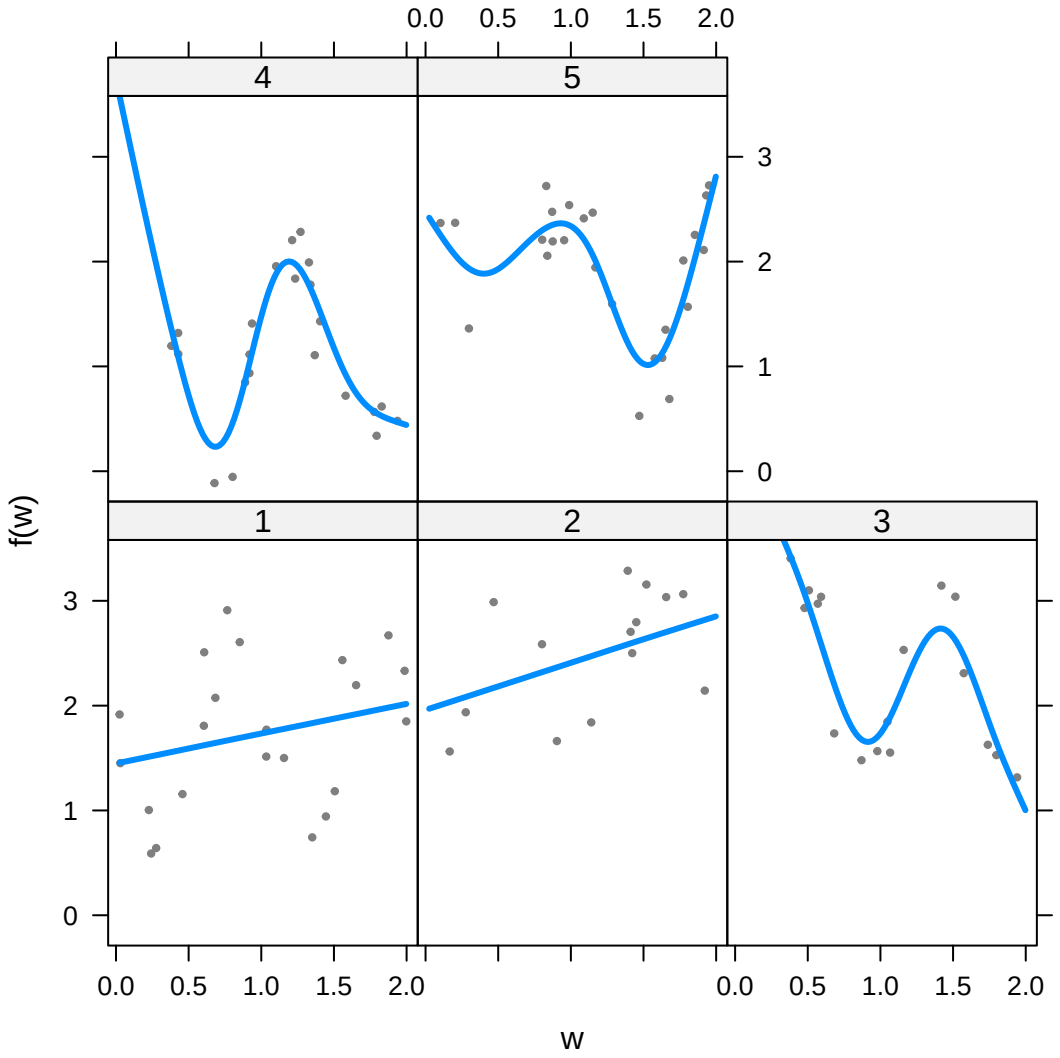
```
visreg(myfit, xvar="group")
```




```
visreg(myfit, xvar="x")
```




```
visreg(myfit, xvar="w", by="group")
```



Alternatively, we can calculate derived quantities via Monte Carlo integration of the estimated density function:

```
# Predicted sample-weighted total
integrate_output(myfit)
#>            Estimate          Std. Error Est. (bias.correct) Std. (bias.correct)
#>          261.165751            4.614605          261.165751                  NA

# True (latent) sample-weighted total
sum( Data$z )
#> [1] 164.0811
```

Similarly, we can fit a grouped 2D spline

```
# Simulate
R = exp(-0.4 * abs(outer(1:10, 1:10, FUN="-")) )
z = mvtnorm::rmvnorm(3, sigma=kronecker(R,R) )
```

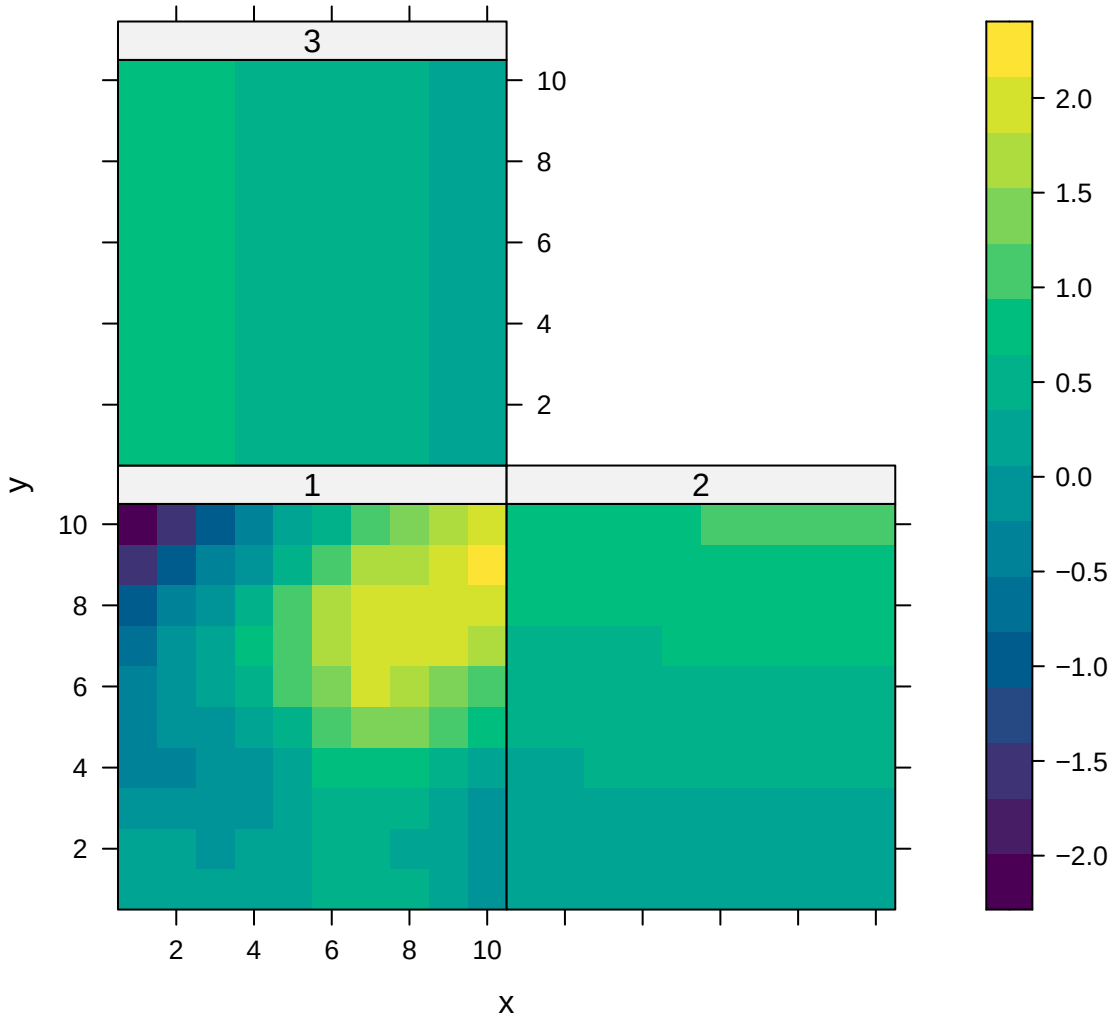
```r
Data = data.frame( expand.grid(x=1:10, y=1:10, group=1:3), z=as.vector(t(z)))
Data$n = Data$z + rnorm(nrow(Data), sd=0.1)
Data$group = factor(Data$group)

# fit model
Formula = n ~ s(x, y, by=group)
myfit = fit( data = Data,
             formula = Formula,
             quiet = TRUE )
#>   0:     531.96653:  0.00000  0.00000  0.00000  0.00000  0.00000
#>   1:     516.82610: 0.951316 -0.00327251 0.0164448 0.0177418 -0.307250
#>   2:     474.56818: 0.475091 -0.00858372 0.0195258 0.0218426 -0.155081
#>   3:     465.28394: 0.510657 -0.0113082 0.0238541 0.0268178 -0.314613
#>   4:     464.31487: 0.545318 -0.135757 0.0716749 0.0976087 -0.367330
#>   5:     464.16156: 0.507435 -0.142334 0.0743180 0.101637 -0.364554
#>   6:     464.00131: 0.522181 -0.170569 0.0858861 0.119086 -0.372061
#>   7:     463.96183: 0.519195 -0.181073 0.0900854 0.125589 -0.359694
#>   8:     463.92115: 0.523613 -0.191130 0.0947393 0.132509 -0.371606
#>   9:     463.82526: 0.514657 -0.218011 0.106283 0.150106 -0.362509
#>  10:     463.71619: 0.524370 -0.243662 0.119012 0.169351 -0.369116
#>  11:     463.38603: 0.508020 -0.361194 0.184295 0.267969 -0.380225
#>  12:     462.95472: 0.521381 -0.449625 0.261130 0.384448 -0.353348
#>  13:     461.57963: 0.489185 -0.723764  1.07603  1.58353 -0.319410
#>  14:     460.26168: 0.532133 -0.592353  1.99922  2.72712 -0.316847
#>  15:     459.17089: 0.535924 -0.563820  3.14827  3.65333 -0.303435
#>  16:     441.95683: 0.600744 0.164635  20.7772  19.3536 -0.0579487
#>  17:     402.34777: 0.785926 0.291587  77.3270  70.0944 0.340504
#>  18:     338.90553: 0.947474 0.113540  133.904  120.806 0.522978
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  19:     259.55611: 0.955220 0.0619610  156.531  141.095 0.330875
#>  20:     243.06043: 0.557186 0.0443723  201.761  181.678 -0.902422
#>  21:     46.813264: 0.525449 0.00780439  224.378  201.965 -0.145050
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  22:    -121.86960: 0.685697 -0.0965361  314.874  283.135 -0.190664
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  23:    -162.42582: 0.609426 -0.0862082  332.965  299.377 -0.247484
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  24:    -436.66720: 0.528586 -0.0424190  477.709  429.306 -0.392507
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  25:    -493.20226: 0.524405 -0.0363637  506.657  455.291 -0.359711
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  26:    -604.27800: 0.518825 -0.0275675  564.555  507.262 -0.299598
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  27:    -823.83973: 0.515800 -0.00840579  680.351  611.204 -0.271786
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  28:    -828.33454: 0.518084 -0.0101900  682.666  613.283 -0.281656
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  29:    -863.54332: 0.521735 -0.0143300  701.194  629.914 -0.296370
#> Warning in nlminb(start = opt$par, obj = obj$fn, gr = obj$gr, control = list(eval.max = control$eval
#>  30:    -864.27336: 0.518713 -0.0347388  701.563  630.246 -0.284977
#>  31:    -865.71539: 0.521671 -0.0403248  702.304  630.912 -0.296914
#>  32:    -866.01807: 0.519107 -0.0586382  702.451  631.044 -0.285698
#>  33:    -866.10404: 0.521580 -0.0640413  702.479  631.070 -0.297258
```

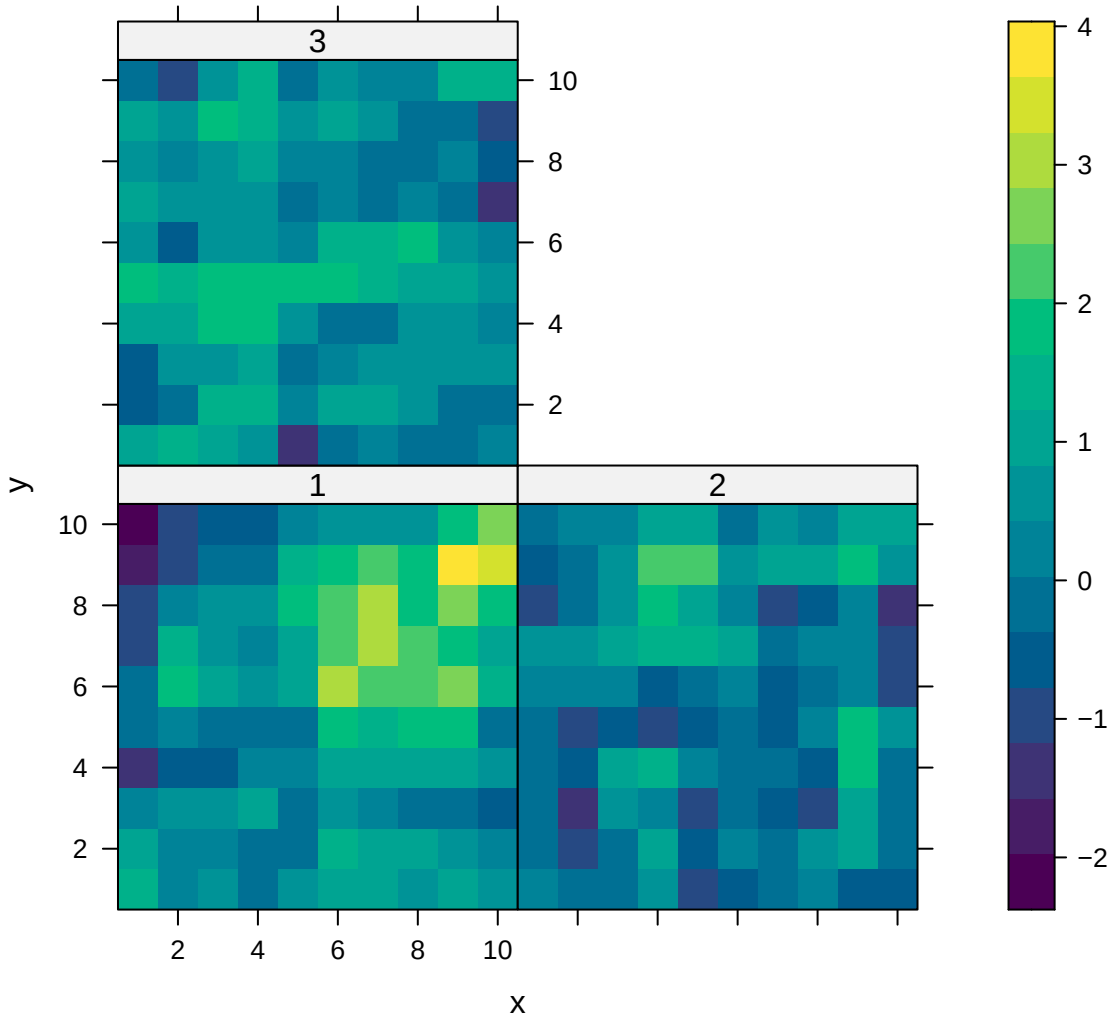
```
#>  34:    -866.12446: 0.520837 -0.0700062  702.482  631.072 -0.293490
#>  35:    -866.23478: 0.523454 -0.122488  702.505  631.095 -0.308752
#>  36:    -866.29525: 0.521312 -0.125978  702.506  631.097 -0.296894
#>  37:    -866.34479: 0.519909 -0.146110  702.516  631.107 -0.289572
#>  38:    -866.36157: 0.520645 -0.148439  702.517  631.108 -0.293709
#>  39:    -866.36426: 0.520727 -0.149178  702.518  631.108 -0.294157
#>  40:    -866.36940: 0.520863 -0.150713  702.519  631.109 -0.294898
#>  41:    -866.37041: 0.520879 -0.151035  702.519  631.109 -0.294988
#>  42:    -866.37241: 0.520909 -0.151682  702.519  631.110 -0.295154
#>  43:    -866.37281: 0.520914 -0.151812  702.519  631.110 -0.295182
#>  44:    -866.37289: 0.520915 -0.151838  702.519  631.110 -0.295187
#>  45:    -866.37291: 0.520915 -0.151843  702.519  631.110 -0.295188
#>  46:    -866.37303: 0.520917 -0.151885  702.519  631.110 -0.295196
#>  47:    -866.37304: 0.520917 -0.151885  702.519  631.110 -0.295197
#>  48:    -866.37304: 0.520917 -0.151887  702.519  631.110 -0.295197
#>  49:    -866.37304: 0.520917 -0.151887  702.519  631.110 -0.295197
#>  50:    -866.37304: 0.520917 -0.151888  702.519  631.110 -0.295197
#>  51:    -866.37304: 0.520917 -0.151888  702.519  631.110 -0.295197
#>  52:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  53:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  54:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  55:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  56:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  57:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  58:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  59:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  60:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  61:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  62:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  63:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  64:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  65:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  66:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  67:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197
#>  68:    -866.37305: 0.520917 -0.151889  702.519  631.110 -0.295197

# compute partial dependence plot
mypartial = partial( object = myfit,
                     pred.var = c("x","y","group"),
                     pred.fun = \(object,newdata) predict(object,newdata),
                     train = Data,
                     approx = TRUE )

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

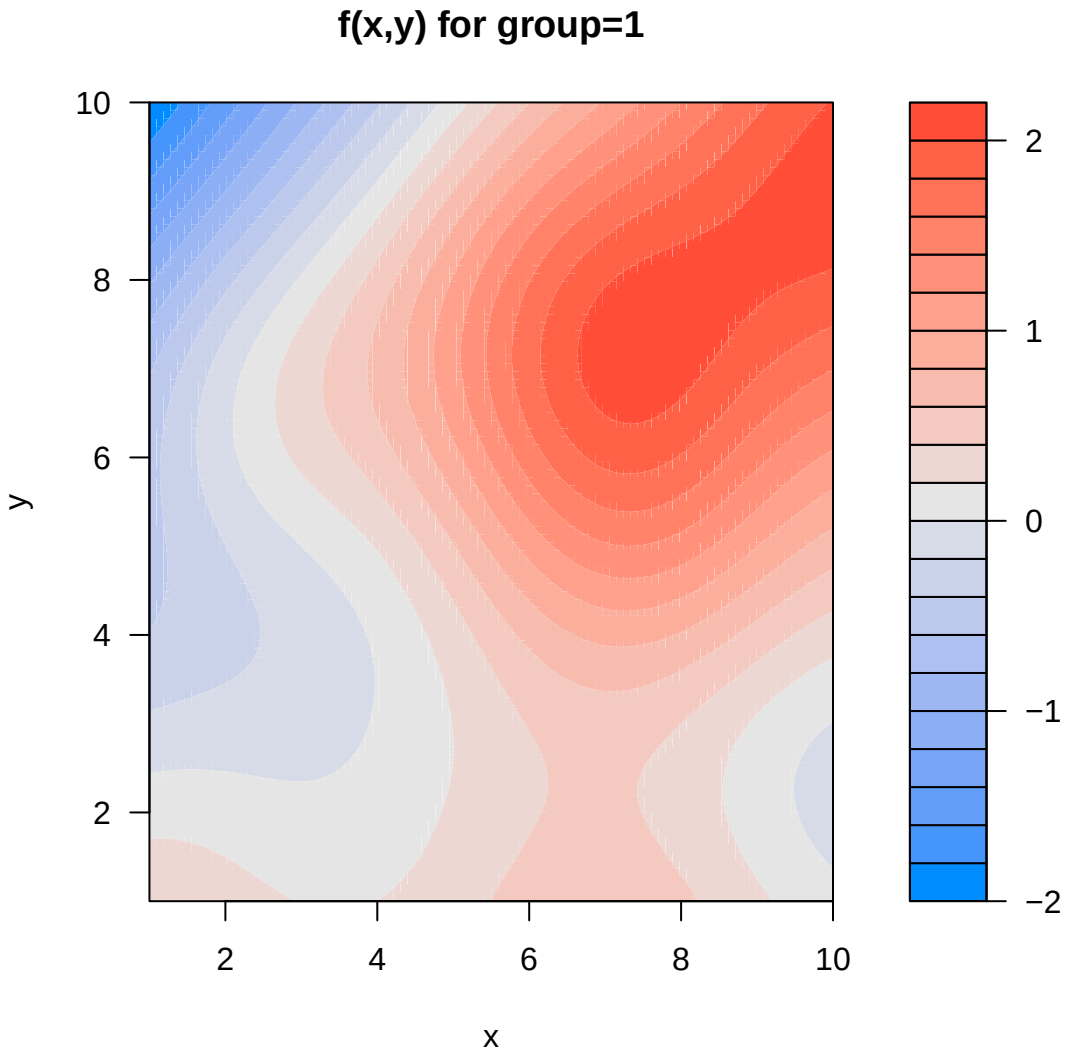



```
# Lattice plot of true values
mypartial$yhat = Data$z
plotPartial( mypartial )
```



We can again use `visreg` to visualize response surfaces, although it doesn't seem possible to extract a grouped spatial term, so we here show only a single term:

```
out = visreg2d( myfit, "x", "y", cond=list("group"=1), plot=FALSE )
plot( out, main="f(x,y) for group=1")
```


f(x,y) for group=1
10
8
6
4
2
y
2
4
6
8
10
x
2
1
0
−1
−2