spatialfusion: short demo

Craig Wang (craig.wang@math.uzh.ch)

June 21, 2019

This brief demo provides code and output for fitting spatial fusion models using R package **spatialfusion**. The first section analyze the built-in synthetic dataset with INLA implementation while the second section analyze a simulated dataset with Stan implementation. The `method` argument in `fusionData()` function decides on which implementation to use.

1. Spatial fusion modelling with INLA on built-in synthetic data

Load libraries

```r
library(spatialfusion)
```

```
## Loading required package: Rcpp

## Loading spatialfusion (version 0.6):
## - The compilation time for a Stan model can be up to 20s.
## - We recommend using INLA method for larger datasets (several thousand observations).
## - It is good practice to test your model on sub-sampled dataset first.
```

```r
library(tmap, quietly = T)
library(sp, quietly = T)
```

Load and view built-in synthetic data

```r
summary(dataGeo)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##        min       max
## x  8.39188  8.930003
## y 47.19461 47.646111
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Number of points: 200
## Data attributes:
##   lungfunction        covariate
##  Min.   :-14.1384   Min.   :-2.76510
##  1st Qu.: -2.6764   1st Qu.:-0.68487
##  Median :  1.1040   Median :-0.04320
##  Mean   :  0.9074   Mean   :-0.05798
##  3rd Qu.:  4.7340   3rd Qu.: 0.73442
##  Max.   : 17.9710   Max.   : 3.20051
```

```r
summary(dataLattice)
```

```
## Object of class SpatialPolygonsDataFrame
## Coordinates:
##         min       max
## x  8.360146  8.984447
```

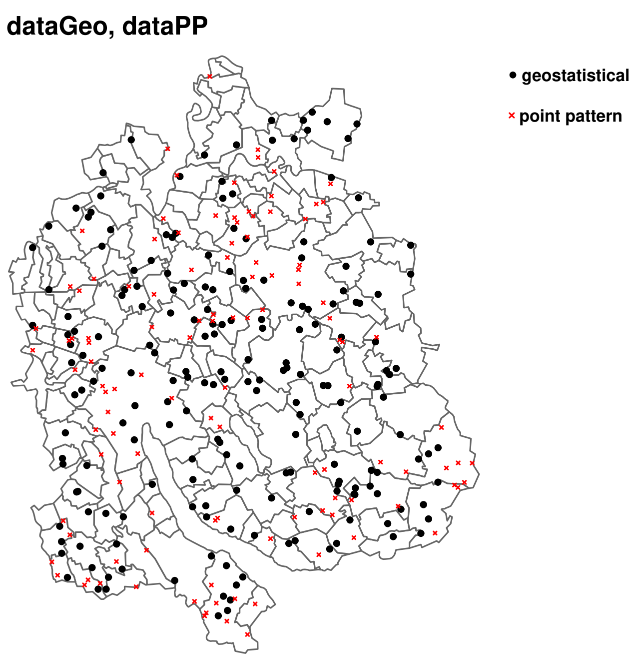
```
## y 47.161094 47.696279
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Data attributes:
##    mortality        covariate              pop                 mr
##  Min.   :  0.00   Min.   :-2.86427   Min.   :   362   Min.   : 0.0000
##  1st Qu.:  0.00   1st Qu.:-0.67110   1st Qu.:  1880   1st Qu.: 0.0000
##  Median :  1.00   Median :-0.05943   Median :  4431   Median : 0.2741
##  Mean   : 14.90   Mean   :-0.05240   Mean   :  9358   Mean   : 1.7206
##  3rd Qu.:  6.75   3rd Qu.: 0.52046   3rd Qu.:  7880   3rd Qu.: 1.4923
##  Max.   :540.00   Max.   : 2.07776   Max.   :413912   Max.   :26.3228
```

```r
summary(dataPP)
```

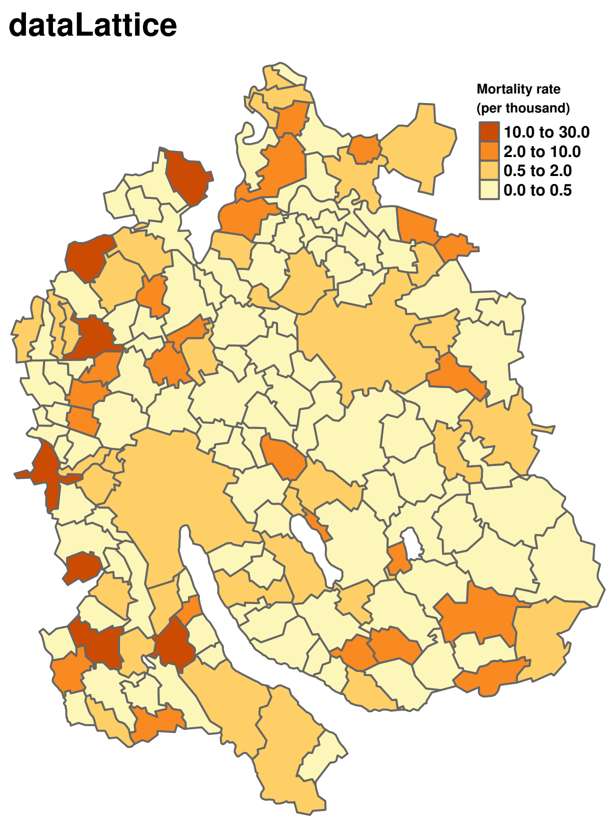
```
## Object of class SpatialPoints
## Coordinates:
##         min       max
## x  8.391983  8.975185
## y 47.177822 47.678063
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Number of points: 116
```

Plot data

```r
tm_shape(dataLattice) + tm_polygons(col = "white") +
  tm_shape(dataGeo) + tm_dots(size = 0.1) +
  tm_add_legend(type = "symbol", shape = 16, size = 0.3, col = "black", label = "geostatistical") +
  tm_shape(dataPP) + tm_symbols(col = "red", shape = 4, size = 0.02) +
  tm_add_legend(type = "symbol", shape = 4, size = 0.2, col = "red", label = "point pattern") +
  tm_layout(main.title = "dataGeo, dataPP", main.title.size = 1,
            frame = F, fontface = 2, legend.outside = T)
```



```r
tm_shape(dataLattice) +
  tm_fill(col = "mr", style = "fixed", breaks = c(0, 0.5, 2, 10, 30),
          title = "Mortality rate \n(per thousand)", legend.reverse = T) + tm_borders() +
  tm_layout(main.title = "dataLattice", main.title.size = 1, frame = F, fontface = 2,
            legend.position = c(0.77,0.8), legend.text.size = 0.5, legend.title.size = 0.5)
```



Data preparation

```r
dat <- fusionData(geo.data = dataGeo, geo.formula = lungfunction ~ covariate,
                  lattice.data = dataLattice,
                  lattice.formula = mortality ~ covariate + log(pop),
                  pp.data = dataPP, distributions = c("normal", "poisson"),
                  method = "INLA")

dat
```

```
## data object for spatial fusion modeling with INLA consisting of:
## - 1 geostatistical variable(s)
## - 1 lattice variable(s)
## - 1 point pattern variable(s)
##
## Provide this object as 'data' argument in fusion() to fit a spatial fusion model.
```

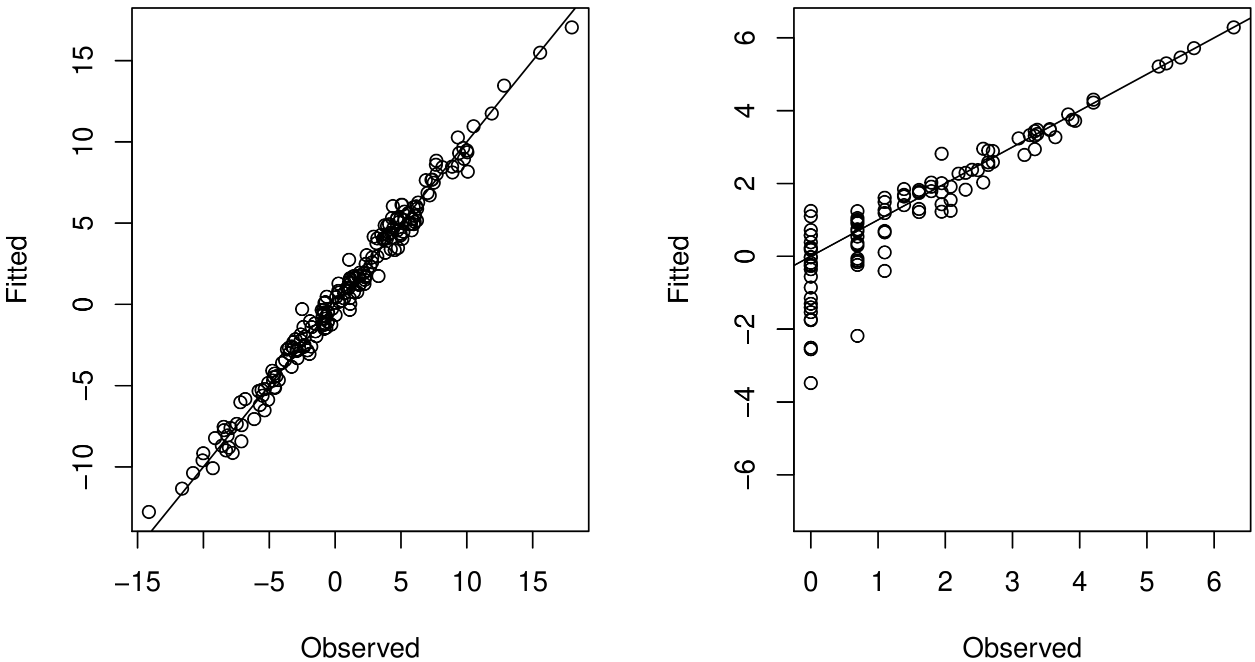
Fit a spatial fusion model

```r
mod <- fusion(data = dat, n.latent = 1, bans = matrix(c(0,0,0), ncol = 1),
              pp.offset = 400, prior.range = c(0.1, 0.5),
```

```
              prior.sigma = c(1, 0.5),  mesh.locs = dat$locs_point,
              mesh.max.edge = c(0.05, 0.5))
```

Inspect the fit

```
mod_fit <- fitted(mod, type = "link")
par(mfrow = c(1,2))
plot(dataGeo$lungfunction, mod_fit$point1,
     xlab = "Observed", ylab = "Fitted")
abline(0,1)
plot(log(dataLattice$mortality), mod_fit$area1,
     xlab = "Observed", ylab = "Fitted")
abline(0,1)
```



Check parameter estimates

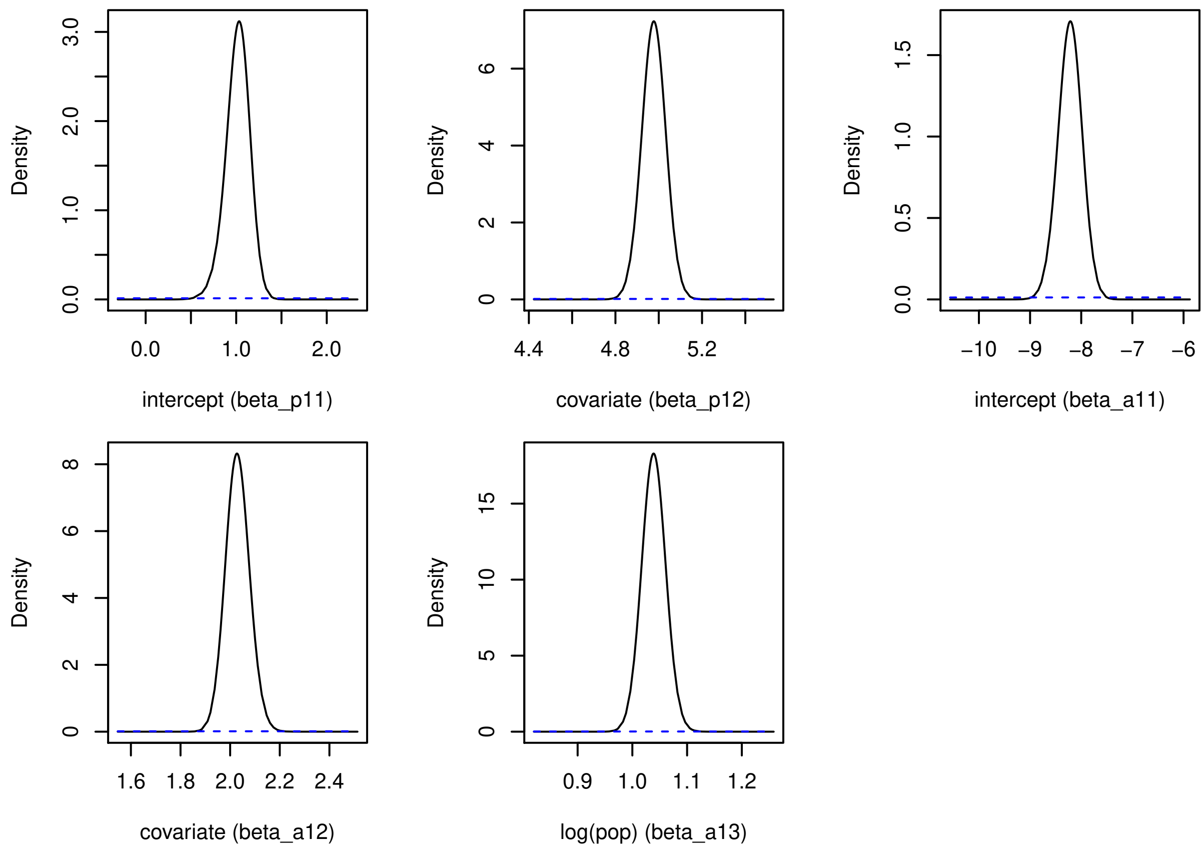
```
summary(mod, digits = 3)
```

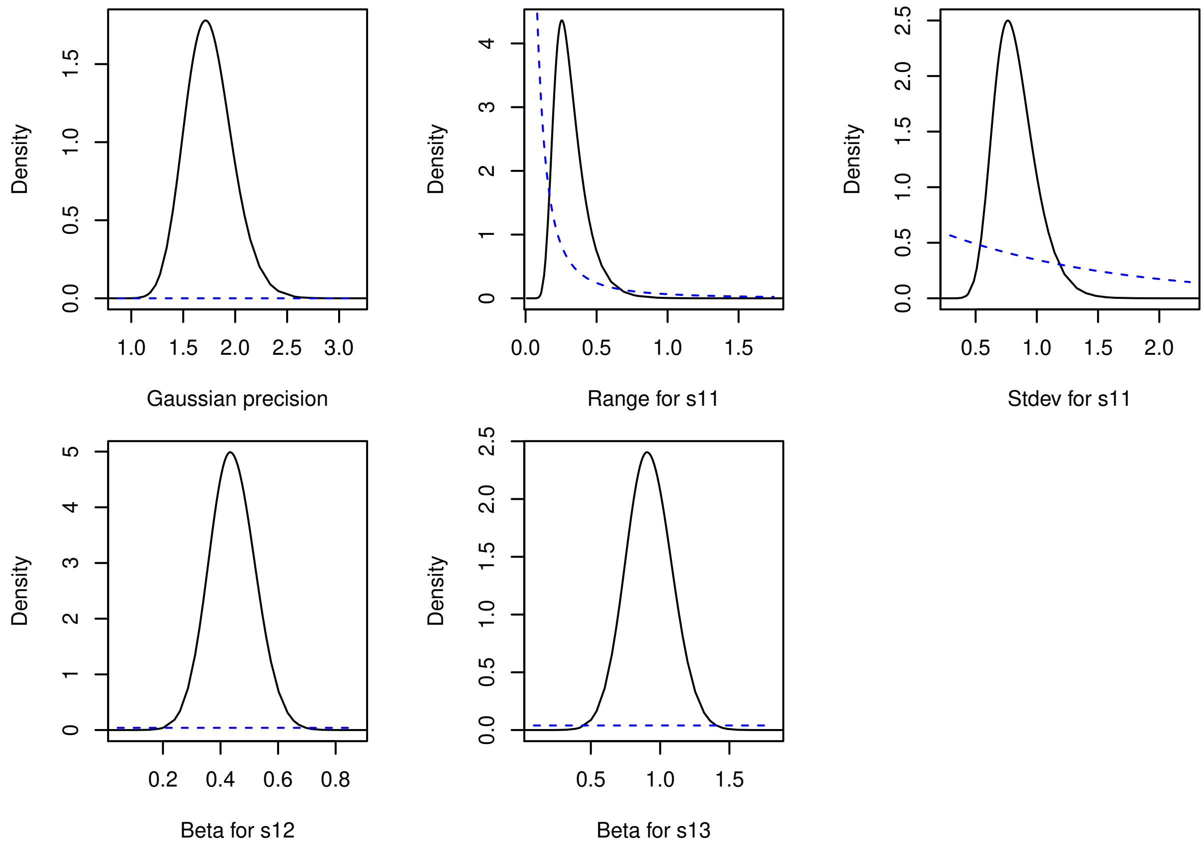
```
## Model:
## geostatistical formula: lungfunction ~ covariate
## lattice formula: mortality ~ covariate + log(pop)
## point pattern variables: 1
## latent process(es): 1
## --------------
## Fixed effect coefficients:
##                          mean     sd 0.025quant 0.5quant 0.975quant  mode
## intercept (beta_p11)   1.02 0.1320      0.735     1.02       1.26  1.03
## covariate (beta_p12)   4.98 0.0554      4.870     4.98       5.09  4.98
## intercept (beta_a11)  -8.22 0.2340     -8.690    -8.22      -7.77 -8.21
## covariate (beta_a12)   2.03 0.0483      1.940     2.03       2.13  2.03
## log(pop) (beta_a13)    1.04 0.0219      0.996     1.04       1.08  1.04
```

```
##
## Latent parameters:
##                      mean    sd 0.025quant 0.5quant 0.975quant  mode
## Gaussian precision 1.750 0.227      1.340    1.740      2.230 1.720
## Range for s11      0.317 0.116      0.157    0.294      0.603 0.256
## Stdev for s11      0.827 0.174      0.546    0.805      1.230 0.763
## Beta for s12       0.441 0.080      0.287    0.439      0.602 0.434
## Beta for s13       0.919 0.166      0.600    0.916      1.250 0.905
```

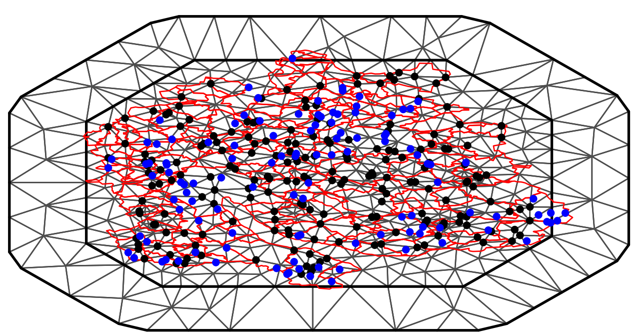
Diagnostic plots

```
plot(mod, interactive = FALSE)
```



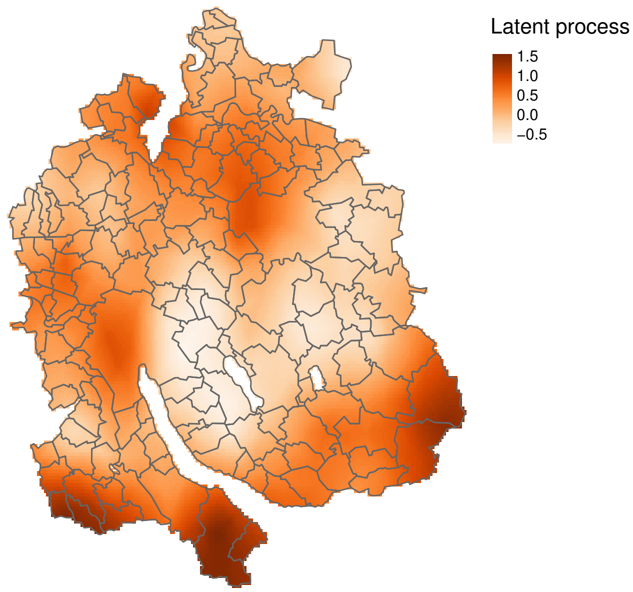



```r
plot(mod, posterior = FALSE)
```



Predict latent surface

```r
pred.locs <- spsample(dataDomain, 20000, type = "regular")
mod.pred <- predict(mod, pred.locs)
mod.pred.plot <- SpatialPointsDataFrame(coords = pred.locs, data = as.data.frame(mod.pred))
tm_shape(mod.pred.plot) +
  tm_symbols(col = "latent.s11", shape = 15, size = 0.05, style = "cont",
             midpoint = NA, legend.col.reverse = T, palette = "Oranges",
             title.col = "Latent process") +
  tm_shape(dataLattice) + tm_borders() +
  tm_layout(frame = FALSE, legend.outside = TRUE)
```



2. Spatial fusion modelling with Stan on simulated data

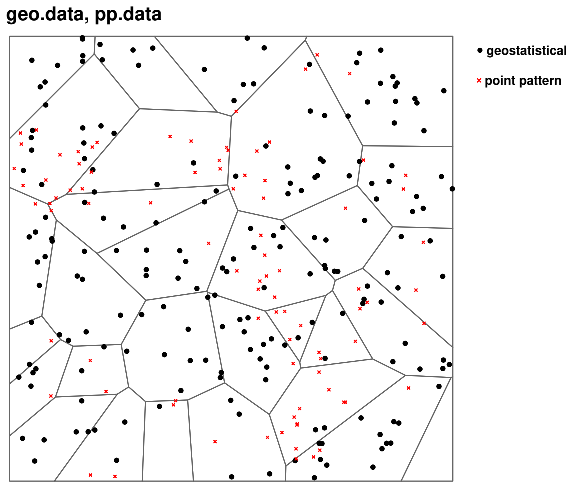
Simulate data

```
dat <- fusionSimulate(n.point = 200, n.area = 30, n.grid = 5, n.pred = 100,
                      psill = 1.5, phi = 1, nugget = 0, tau.sq = 0.2,
                      dimension = 10, domain = NULL, point.beta = list(rbind(1,5)),
                      area.beta = list(rbind(1, 1.5)), nvar.pp = 1,
                      distributions = c("normal","poisson"),
                      design.mat = matrix(c(2, 0.5, 1), ncol = 1),
                      pp.offset = 0.5, seed = 1)

geo.data <- SpatialPointsDataFrame(coords = dat$mrf[dat$sample.ind, c("x","y")],
                                   data = data.frame(cov.point = dat$dat$X_point[,2],
                                                     outcome = dat$dat$Y_point[[1]]),
                                   proj4string = CRS("+proj=longlat +ellps=WGS84"))

lattice.data <- SpatialPolygonsDataFrame(dat$poly,
                                         data = data.frame(outcome = dat$dat$Y_area[[1]],
                                                           cov.area = dat$dat$X_area[,2]))

pp.data <- dat$data$lgcp.coords[[1]]

lattice.data@proj4string <- pp.data@proj4string <- CRS("+proj=longlat +ellps=WGS84")
```

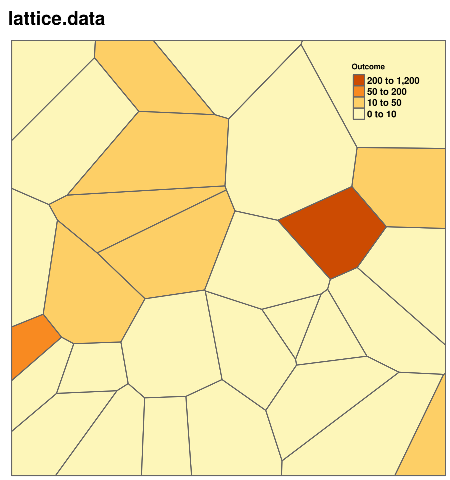
Plot data

```
tm_shape(lattice.data) + tm_polygons(col = "white") +
  tm_shape(geo.data) + tm_dots(size = 0.1) +
  tm_add_legend(type = "symbol", shape = 16, size = 0.3, col = "black", label = "geostatistical") +
  tm_shape(pp.data) + tm_symbols(col = "red", shape = 4, size = 0.02) +
  tm_add_legend(type = "symbol", shape = 4, size = 0.2, col = "red", label = "point pattern") +
  tm_layout(main.title = "geo.data, pp.data", main.title.size = 1,
```

```
            frame = F, fontface = 2, legend.outside = T)
```




```
tm_shape(lattice.data) +
  tm_fill(col="outcome", style="fixed", breaks=c(0, 10, 50, 200, 1200),
          title = "Outcome", legend.reverse = T) + tm_borders() +
  tm_layout(main.title="lattice.data", main.title.size = 1, frame = F, fontface = 2,
            legend.position = c(0.77,0.8), legend.text.size = 0.5, legend.title.size = 0.5)
```



Data preparation

```
dat2 <- fusionData(geo.data = geo.data, geo.formula = outcome ~ cov.point,
                   lattice.data = lattice.data, lattice.formula = outcome ~ cov.area,
                   pp.data = pp.data, distributions = c("normal","poisson"),
                   method = "Stan")
dat2
```

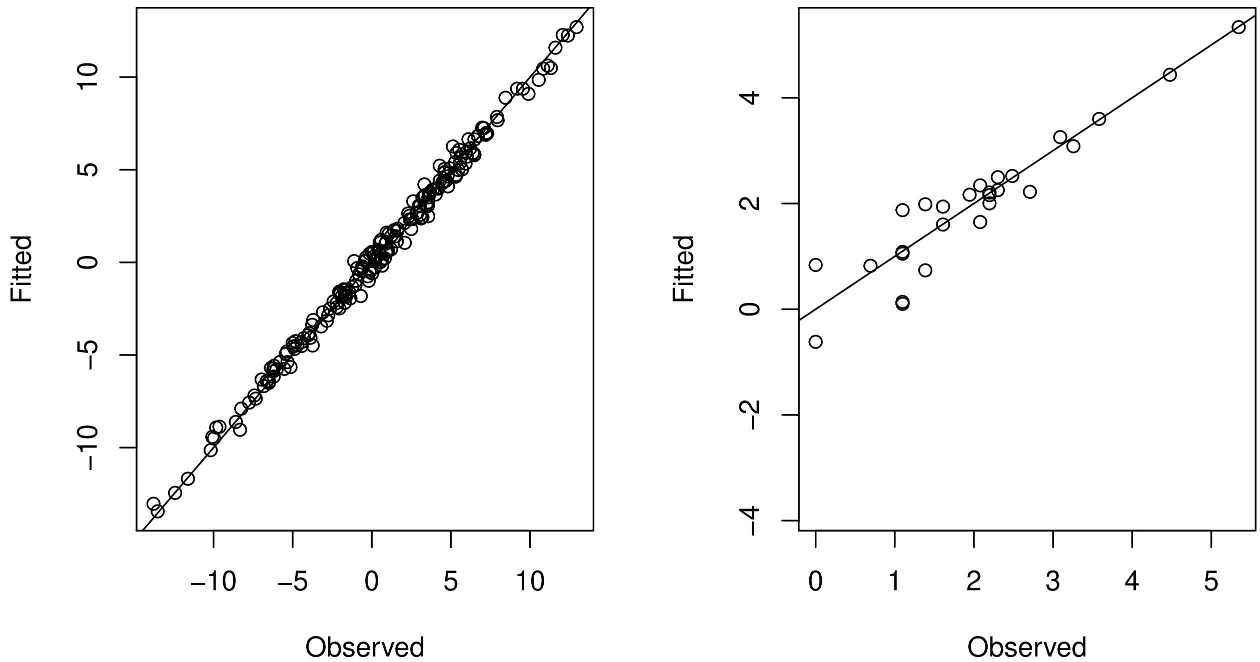
```
## data object for spatial fusion modeling with Stan consisting of:
## - 1 geostatistical variable(s), with 200 locations
## - 1 lattice variable(s), with 150 sampling point locations
## - 1 point pattern variable(s), with 100 gridded locations
##
## Provide this object as 'data' argument in fusion() to fit a spatial fusion model.
```

Fit a spatial fusion model

```r
mod <- fusion(data = dat2, n.latent = 1, bans = matrix(c(0,0,0), ncol = 1),
              pp.offset = 0.5, prior.phi = list(distr = "normal", pars = c(1, 1)))
```

Inspect the fit

```r
mod_fit <- fitted(mod, type = "link")
par(mfrow = c(1,2))
plot(dat$data$Y_point[[1]], mod_fit$point1, xlab = "Observed", ylab = "Fitted")
abline(0,1)
plot(log(dat$data$Y_area[[1]]), mod_fit$area1, xlab = "Observed", ylab = "Fitted")
abline(0,1)
```



Check parameter estimates

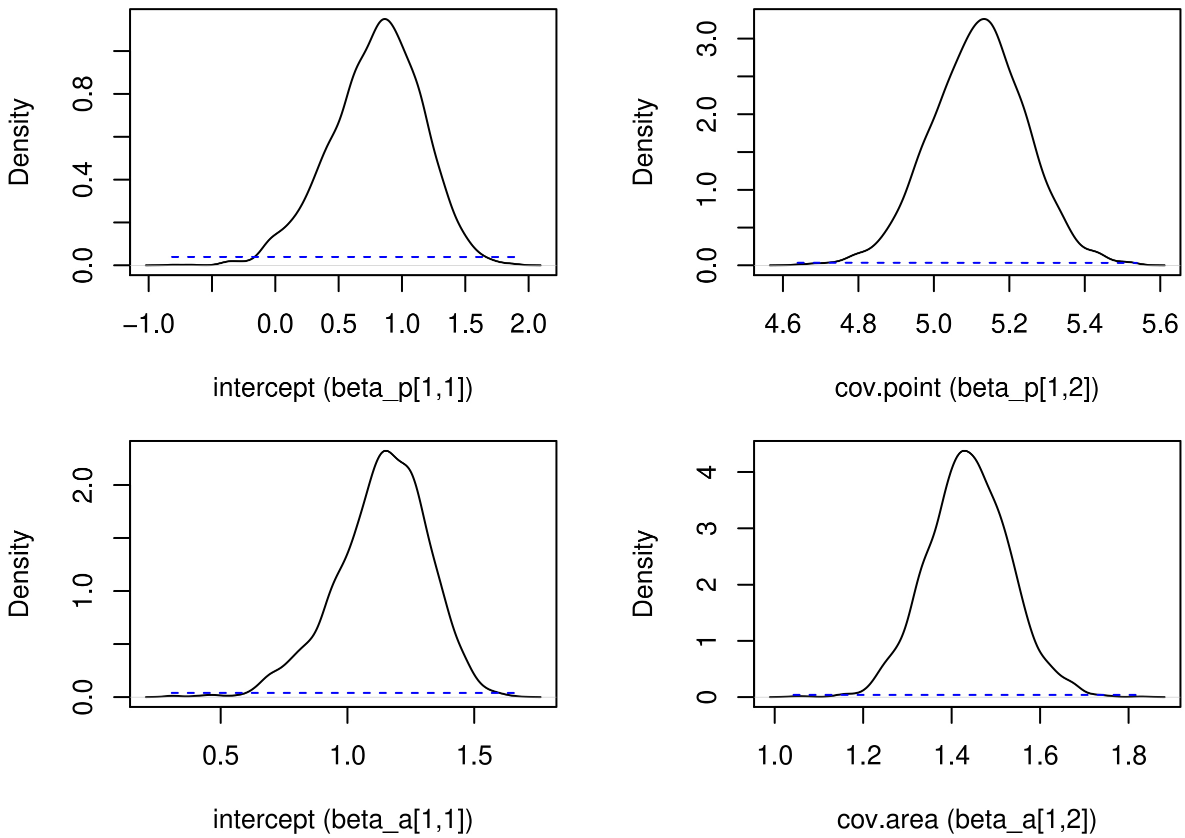
```r
summary(mod, digits = 2)
```

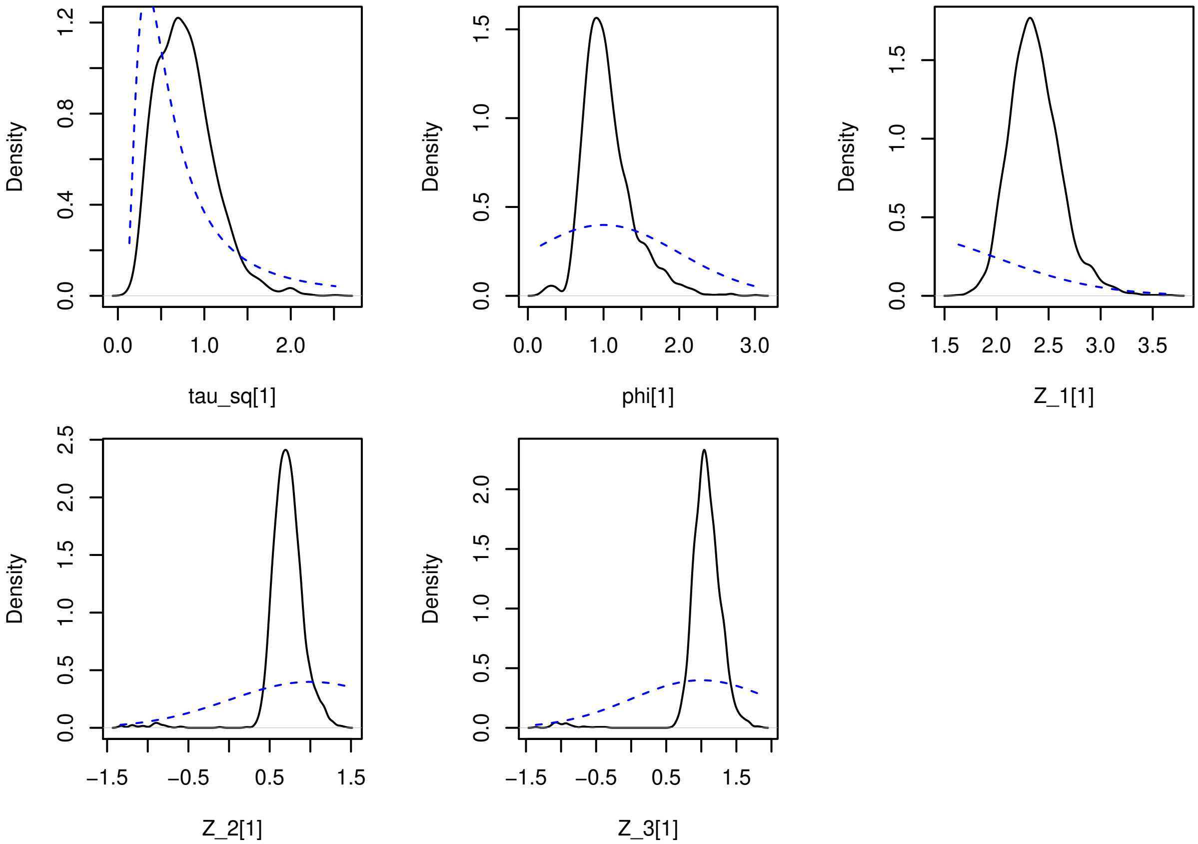
```
## Model:
## geostatistical formula: outcome ~ cov.point
## lattice formula: outcome ~ cov.area
## point pattern variables: 1
## latent process(es): 1
## ---------------
## Fixed effect coefficients:
##                          mean se_mean    sd  2.5%  25%  50% 75% 97.5% n_eff
## intercept (beta_p[1,1])  0.8  0.0180 0.360 0.027 0.57 0.82 1.0   1.4   380
## cov.point (beta_p[1,2])  5.1  0.0053 0.130 4.900 5.00 5.10 5.2   5.4   560
## intercept (beta_a[1,1])  1.1  0.0063 0.180 0.740 1.00 1.10 1.3   1.5   830
## cov.area (beta_a[1,2])   1.4  0.0025 0.093 1.300 1.40 1.40 1.5   1.6  1400
##                          Rhat
## intercept (beta_p[1,1])    1
```

```
## cov.point (beta_p[1,2])     1
## intercept (beta_a[1,1])     1
## cov.area (beta_a[1,2])      1
##
## Latent parameters:
##            mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## tau_sq[1] 0.78   0.039 0.33 0.29 0.53 0.75 0.97   1.5    71    1
## phi[1]    1.10   0.022 0.33 0.60 0.84 0.99 1.20   1.9   220    1
## Z_1[1]    2.40   0.017 0.25 2.00 2.20 2.40 2.50   2.9   210    1
## Z_2[1]    0.71   0.025 0.26 0.41 0.61 0.71 0.82   1.1   110    1
## Z_3[1]    1.10   0.034 0.31 0.73 0.95 1.10 1.20   1.5    81    1
```

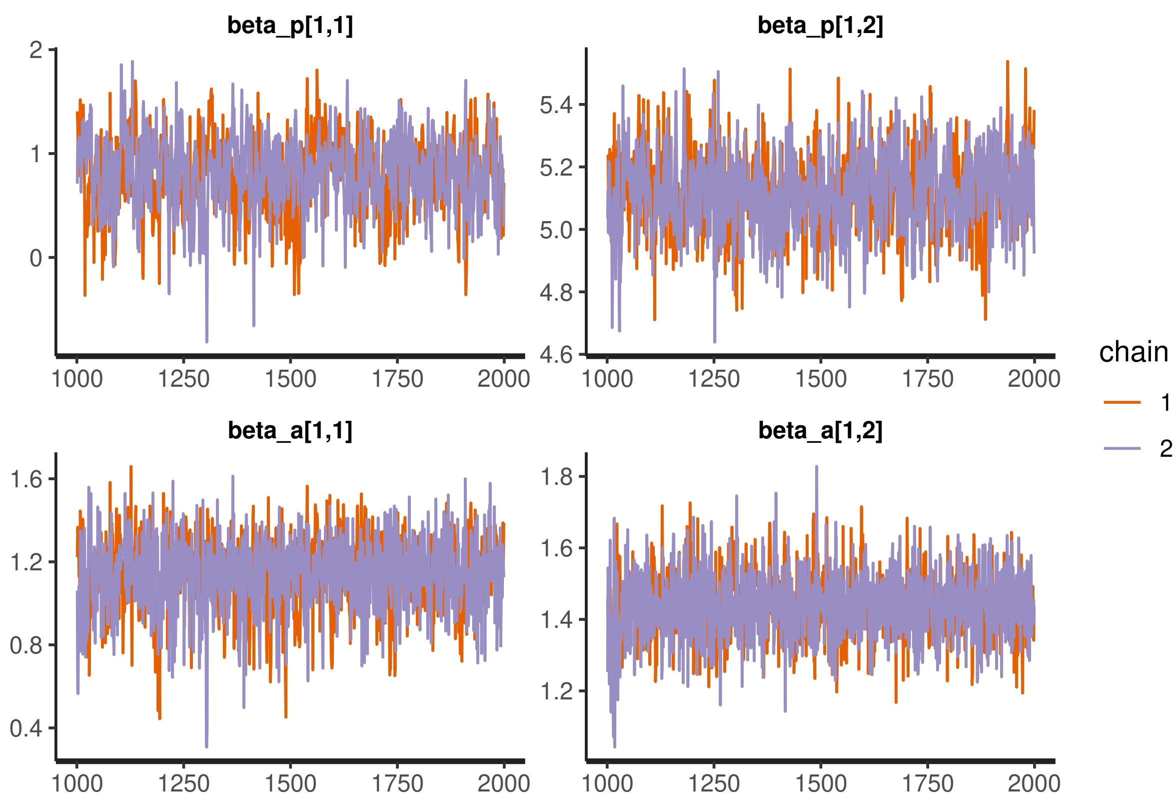
Diagnostic plots

```r
plot(mod, interactive = FALSE)
```

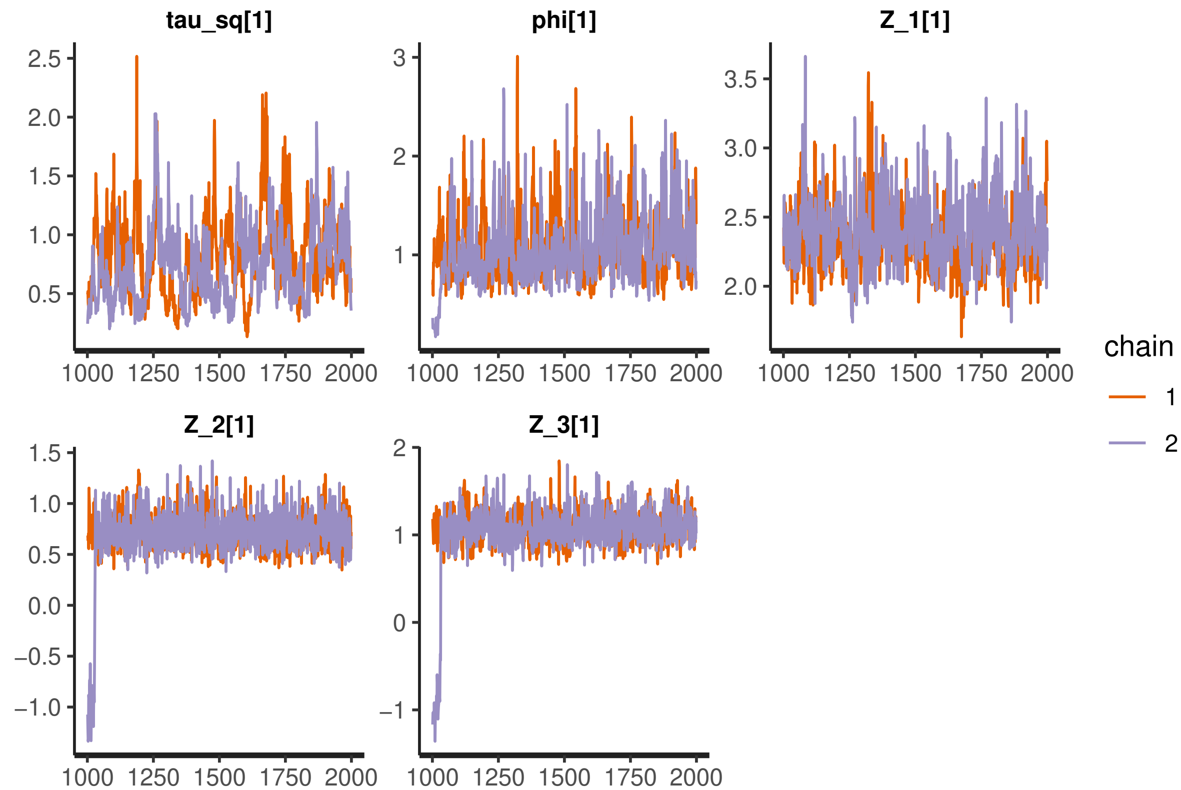





```
plot(mod, posterior = FALSE)
```

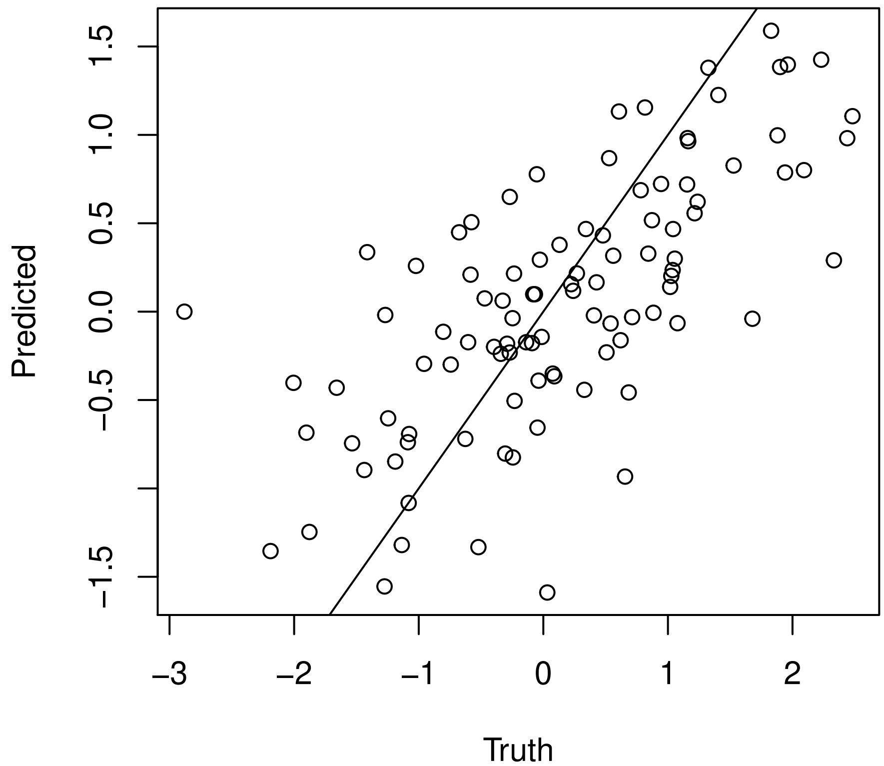


```
## Enter <return> to proceed ...
```



Predict latent surface and compare with simulated truth

```
mod.pred <- predict(mod, new.locs = dat$pred.loc, type = "summary")
par(mfrow = c(1,1))
plot(dat$mrf[dat$pred.ind, c("sim1")], mod.pred$latent1,
     xlab = "Truth", ylab = "Predicted")
abline(0,1)
```


Predicted
Truth
1.5
1.0
0.5
0.0
−0.5
−1.5
−3
−2
−1
0
1
2