Various GLGM examples

Patrick Brown

February 9, 2018

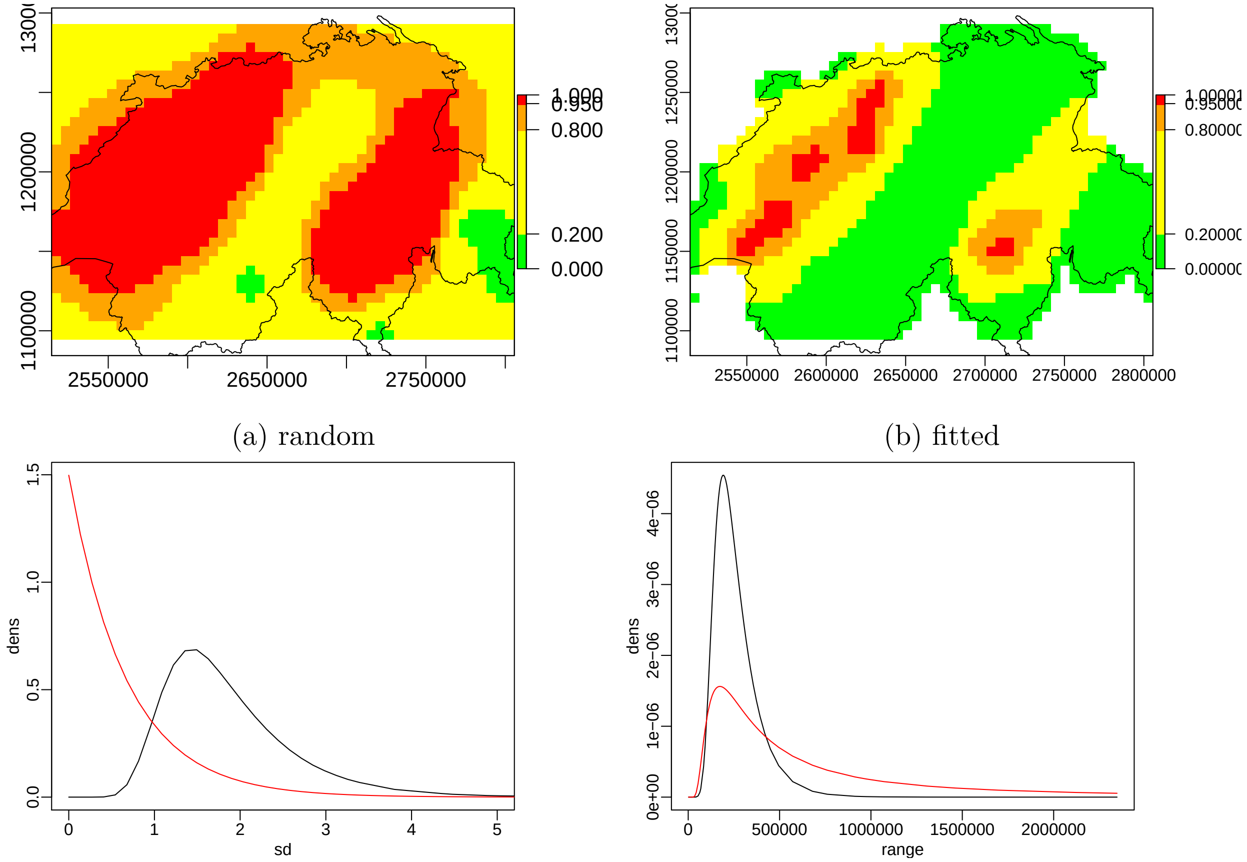
```
library('mapmisc')

## Loading required package:  sp
## Loading required package:  raster
## map images will be cached in  /tmp/RtmpOd1dCO/mapmiscCache

library("geostatsp")

## Loading required package:  Matrix

data('swissRain')

havePackages = c(
  'INLA' = requireNamespace('INLA', quietly=TRUE)
)

print(havePackages)

## INLA
## TRUE

swissRain$lograin = log(swissRain$rain)
swissAltitudeCrop = raster::mask(swissAltitude,swissBorder)
```

number of cells... smaller is faster but less interesting

```
fact

## [1] 2

(Ncell = 25*fact)

## [1] 50
```

standard formula

```
if(all(havePackages)) {

  swissFit =  glgm(
    formula = lograin~ CHE_alt,
    data = swissRain,
    grid = Ncell,
    buffer = 10*1000,
    covariates=swissAltitudeCrop,
    family="gaussian",
    prior = list(
      sd=c(2, 0.05),
      sdObs = 1,
      range=c(500000, 0.5)),
    control.inla = list(strategy='gaussian')
  )
  knitr::kable(swissFit$parameters$summary, digits=3)


  swissExc = excProb(
    x=swissFit,  random=TRUE,
    threshold=0)

  plot(swissExc, breaks = c(0, 0.2, 0.8, 0.95, 1.00001),
    col=c('green','yellow','orange','red'))

  plot(swissBorder, add=TRUE)


  swissExcP = excProb(
    swissFit$inla$marginals.predict, 3,
      template=swissFit$raster)
  plot(swissExcP, breaks = c(0, 0.2, 0.8, 0.95, 1.00001),
    col=c('green','yellow','orange','red'))
  plot(swissBorder, add=TRUE)

  matplot(
    swissFit$parameters$sd$posterior[,'x'],
    swissFit$parameters$sd$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='sd', ylab='dens', xlim = c(0,5))

  matplot(
    swissFit$parameters$range$posterior[,'x'],
    swissFit$parameters$range$posterior[,c('y','prior')],
```



(a) random

(b) fitted

(c) sd

(d) range

Figure 1: Swiss rain as in help file

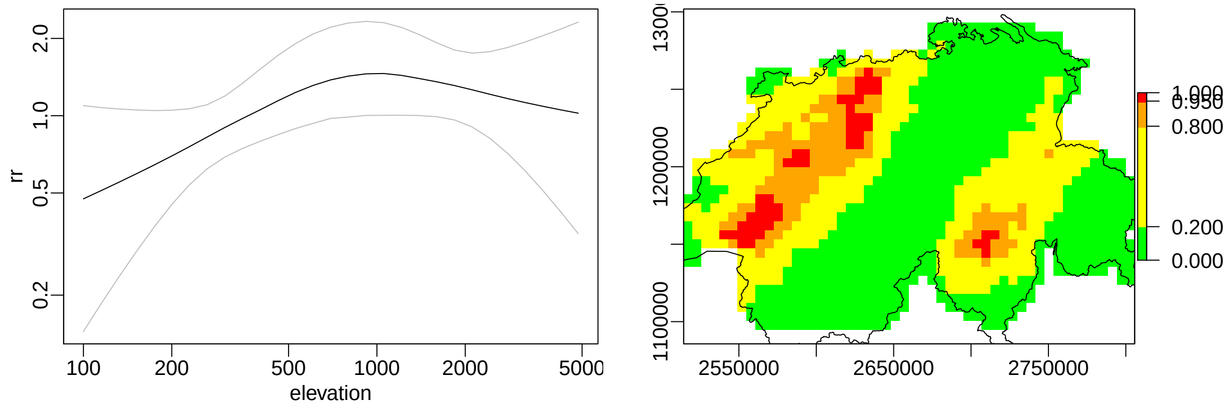
```
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')
}
```

non-parametric elevation effect

```
altSeq = exp(seq(
    log(100), log(5000),
    by = log(2)/5))

swissAltCut = raster::cut(
  swissAltitudeCrop,
  breaks=altSeq
)
names(swissAltCut) = 'bqrnt'

if(all(havePackages)) {
  swissFitNp = glgm(
    formula = lograin ~ f(bqrnt, model = 'rw2', scale.model=TRUE,
      values = 1:length(altSeq),
      prior = 'pc.prec', param = c(0.1, 0.01)),
    data=swissRain,
    grid = Ncell,
    covariates=swissAltCut,
    family="gaussian", buffer=20000,
    prior=list(
      sd=c(u = 0.5, alpha = 0.1),
      range=c(50000,500000),
      sdObs = c(u=1, alpha=0.4)),
    control.inla=list(strategy='gaussian')
  )
  knitr::kable(swissFitNp$parameters$summary, digits=3)

  matplot(
    altSeq,
    exp(swissFitNp$inla$summary.random$bqrnt[,
      c('0.025quant', '0.975quant', '0.5quant')]),
    log='xy',
    xlab ='elevation', ylab='rr',
  type='l',
    lty = 1,
    col=c('grey','grey','black')
    )

  swissExcP = excProb(swissFitNp$inla$marginals.predict,
    3, template=swissFitNp$raster)
  plot(swissExcP, breaks = c(0, 0.2, 0.8, 0.95, 1.00001),
    col=c('green','yellow','orange','red'))
  plot(swissBorder, add=TRUE)

}
```

intercept only, named response variable. legacy priors

```
if(all(havePackages)) {
  swissFit =  glgm("lograin", swissRain, Ncell,
    covariates=swissAltitude, family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000), sdObs = 2),
    control.inla=list(strategy='gaussian')
  )
```



(a) elevation effect

(b) fitted

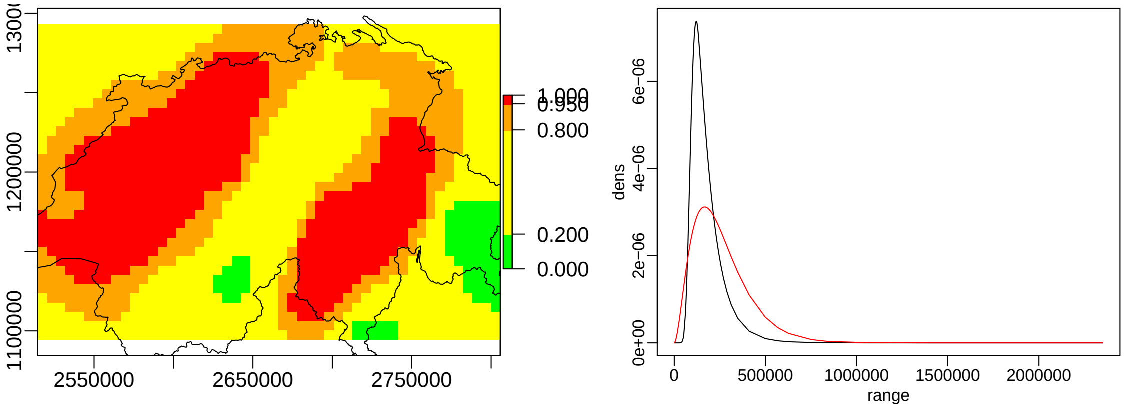
Figure 2: Swiss rain elevation rw2

```
  knitr::kable(swissFit$parameters$summary[,c(1, 3:5, 8)], digits=4)
}
```

	mean	0.025quant	0.5quant	0.975quant	meanExp
(Intercept)	2.2578	1.5896	2.2658	2.8827	9.9531
CHE alt	-0.0001	-0.0004	-0.0001	0.0002	1.0035
range/1000	172.0648	72.0418	149.5736	401.3490	NA
sdNugget	0.3748	0.2740	0.3744	0.4958	NA
sd	1.2471	0.6866	1.1676	2.4257	NA

intercept only, add a covariate just to confuse glgm.

```
if(all(havePackages)) {

  swissFit =  glgm(
    formula=lograin~1,
    data=swissRain,
    grid=Ncell,
    covariates=swissAltitude,
    family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla=list(strategy= 'gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma", param=c(.1, .1))))
  )

  knitr::kable(swissFit$parameters$summary[,c(1, 3:5, 8)], digits=3)

  swissExc = excProb(
    swissFit$inla$marginals.random$space, 0,
    template=swissFit$raster)
  plot(swissExc, breaks = c(0, 0.2, 0.8, 0.95, 1.00001),
```



(a) exc prob

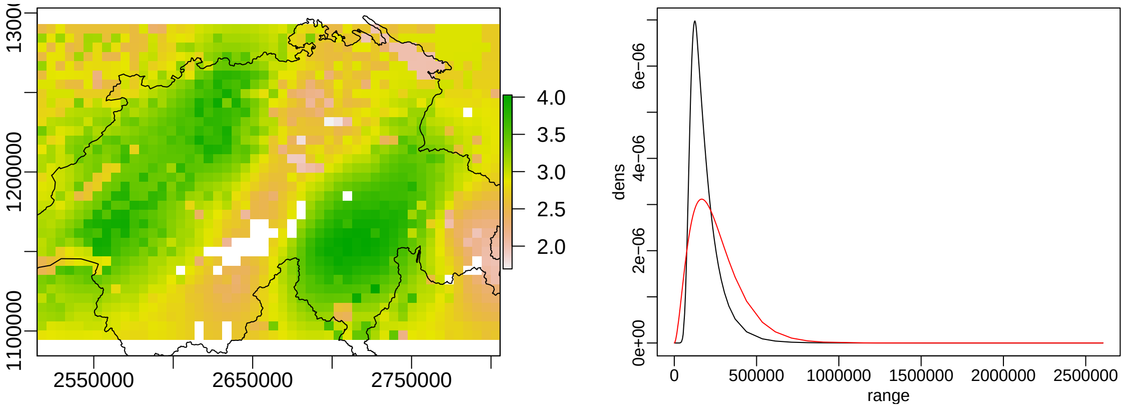
(b) range

Figure 3: Swiss intercept only

```
    col=c('green','yellow','orange','red'))
  plot(swissBorder, add=TRUE)

    matplot(
    swissFit$parameters$range$posterior[,'x'],
    swissFit$parameters$range$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')

}
```

covariates are in data

```
newdat = swissRain
newdat$elev = extract(swissAltitude, swissRain)
if(all(havePackages)) {
  swissFit =  glgm(lograin~ elev + land,
    newdat, Ncell,
    covariates=list(land=swissLandType),
    family="gaussian", buffer=40000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
  )
  knitr::kable(swissFit$parameters$summary, digits=3)

  plot(swissFit$raster[['predict.mean']])
   plot(swissBorder, add=TRUE)
```



(a) predict.mean

(b) range

Figure 4: covaraites in data

```
    matplot(
    swissFit$parameters$range$posterior[,'x'],
    swissFit$parameters$range$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')

}
```

formula, named list elements

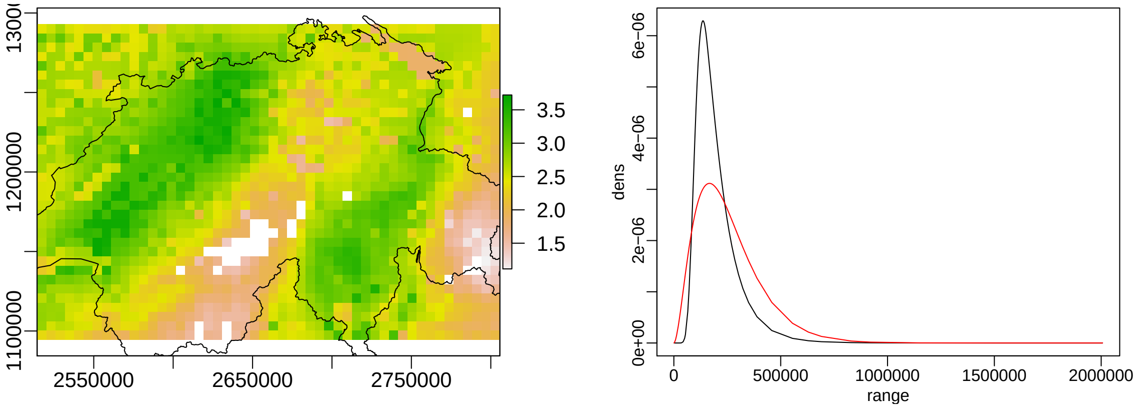
```
if(all(havePackages)) {

  swissFit =  glgm(lograin~ elev,
    swissRain, Ncell,
    covariates=list(elev=swissAltitude),
    family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.mode=list(theta=c(1.9,0.15,2.6),restart=TRUE),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
  )
  swissFit$parameters$summary[,c(1,3,5)]
}

##                      mean    0.025quant   0.975quant
## (Intercept)  2.253108e+00  1.5858154604 2.877664e+00
## elev        -8.339353e-05 -0.0003827184 2.147521e-04
## range/1000   1.746615e+02 72.8998567207 4.086975e+02
## sdNugget     3.817998e-01  0.2834201614 4.999517e-01
## sd           1.254819e+00  0.6794445314 2.479189e+00
```

categorical covariates

```
if(all(havePackages)) {
  swissFit =  glgm(
    formula = lograin ~ elev + factor(land),
    data = swissRain, grid = Ncell,
    covariates=list(elev=swissAltitude,land=swissLandType),
    family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla=list(strategy='gaussian'),
    control.family=list(hyper=list(
      prec=list(prior="loggamma",
          param=c(.1, .1))))
  )

  knitr::kable(swissFit$parameters$summary[,c(1,3,5)], digits=3)


  plot(swissFit$raster[['predict.mean']])
   plot(swissBorder, add=TRUE)

    matplot(
    swissFit$parameters$range$posterior[,'x'],
    swissFit$parameters$range$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')

}
```

put some missing values in covaritates also dont put factor() in formula

```
temp = values(swissAltitude)
temp[seq(10000,12000)] = NA
values(swissAltitude) = temp
if(all(havePackages)) {

  swissFitMissing =  glgm(rain ~ elev + land,swissRain,  Ncell,
    covariates=list(elev=swissAltitude,land=swissLandType),
    family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
```



(a) map

(b) range

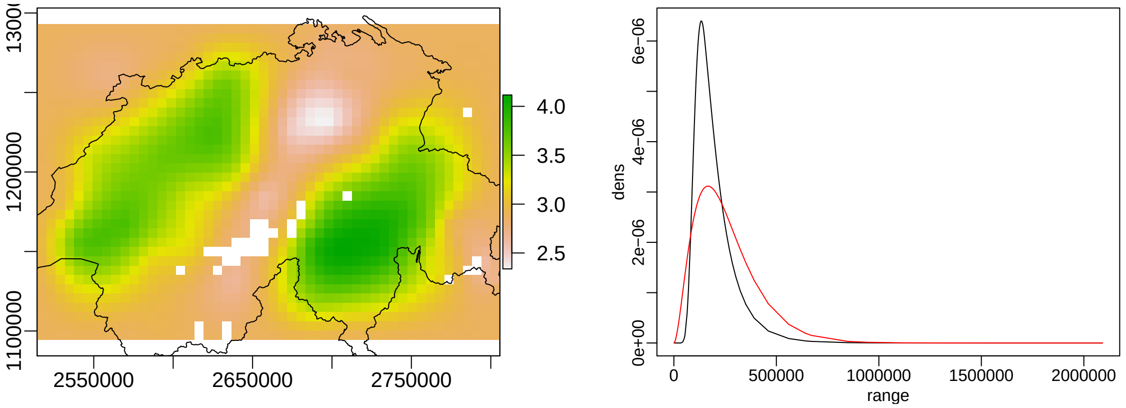
Figure 5: categorical covariates

```
  )
  knitr::kable(swissFitMissing$parameters$summary[,1:5], digits=3)

}
```

	mean	sd	0.025quant	0.5quant	0.975quant
(Intercept)	20.138	4.942	10.086	20.260	29.525
elev	-0.004	0.003	-0.009	-0.004	0.001
landMixed forests	-3.358	2.039	-7.394	-3.351	0.637
landGrasslands	-3.713	2.981	-9.587	-3.712	2.148
landCroplands	-5.073	2.439	-9.902	-5.065	-0.294
landUrban and built-up	-8.862	4.195	-17.107	-8.866	-0.604
landEvergreen needleleaf forest	-5.457	3.957	-13.304	-5.437	2.277
landWater bodies	-11.742	5.094	-21.829	-11.722	-1.775
landDeciduous needleleaf forest	-5.335	4.976	-15.159	-5.326	4.429
landDeciduous broadleaf forest	6.518	5.240	-3.745	6.501	16.865
landOpen shrublands	-7.119	7.162	-21.262	-7.103	6.921
landPermanent Wetlands	-16.294	8.055	-32.111	-16.306	-0.428
range/1000	160.112	78.197	71.532	139.660	365.812
sdNugget	4.627	0.954	3.258	4.606	6.367
sd	16.616	35.686	9.727	15.605	30.984

covariates are in data, interactions

```
newdat = swissRain
newdat$elev = extract(swissAltitude, swissRain)
if(all(havePackages)) {

  swissFit =  glgm(
    formula = lograin~ elev : land,
    data=newdat,
```



(a) map

(b) range

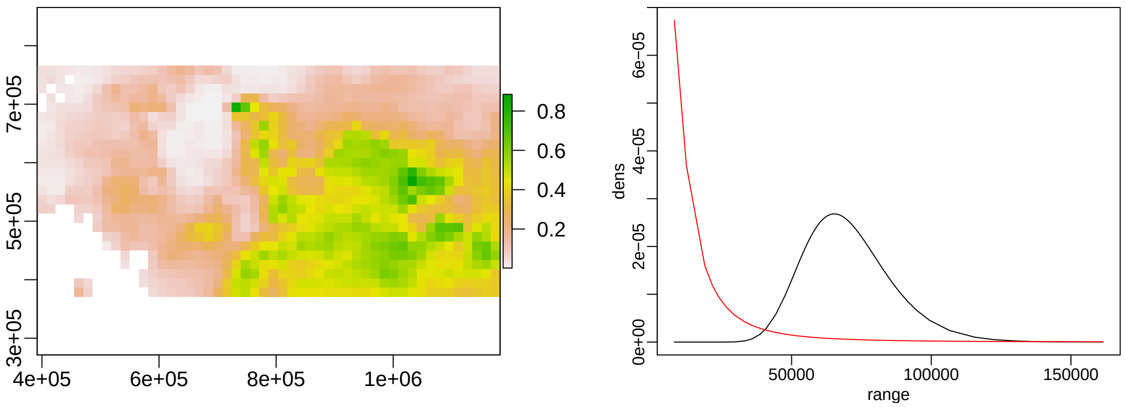
Figure 6: interactions

```
    grid=squareRaster(swissRain,50),
    covariates=list(land=swissLandType),
    family="gaussian", buffer=0,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
  )

  knitr::kable(swissFit$parameters$summary, digits=3)

  plot(swissFit$raster[['predict.mean']])
     plot(swissBorder, add=TRUE)

    matplot(
    swissFit$parameters$range$posterior[,'x'],
    swissFit$parameters$range$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')

}
```

these tests are time consuming, so only run them if the `fact` variable is set to a value above 1.

```
  data('loaloa')
  rcl = rbind(
    # wedlands and mixed forests to forest
    c(5,2),c(11,2),
# savannas to woody savannas
```

```
    c(9,8),
    # croplands and urban changed to crop/natural mosaid
    c(12,14),c(13,14))
  ltLoaR = reclassify(ltLoa, rcl)
  levels(ltLoaR) = levels(ltLoa)


  elevationLoa = elevationLoa - 750
  elevLow = reclassify(elevationLoa, c(0, Inf, 0))
  elevHigh = reclassify(elevationLoa, c(-Inf, 0, 0))

  covList = list(elLow = elevLow, elHigh = elevHigh,
    land = ltLoaR, evi=eviLoa)

if(all(havePackages) & fact > 1) {


  loaFit = glgm(
    y ~ land + evi + elHigh + elLow +
      f(villageID, prior = 'pc.prec', param = c(log(2), 0.5),
       model="iid"),
    loaloa,
    Ncell,
    covariates=covList,
    family="binomial", Ntrials = loaloa$N,
    shape=2, buffer=25000,
    prior = list(
      sd=log(2),
      range = list(prior = 'invgamma', param = c(shape=2,rate=1))),
    control.inla = list(strategy='gaussian')
    )

  loaFit$par$summary[,c(1,3,5)]

  plot(loaFit$raster[['predict.exp']])

    matplot(
      loaFit$parameters$range$posterior[,'x'],
      loaFit$parameters$range$posterior[,c('y','prior')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')

}
```



(a) predict

(b) range

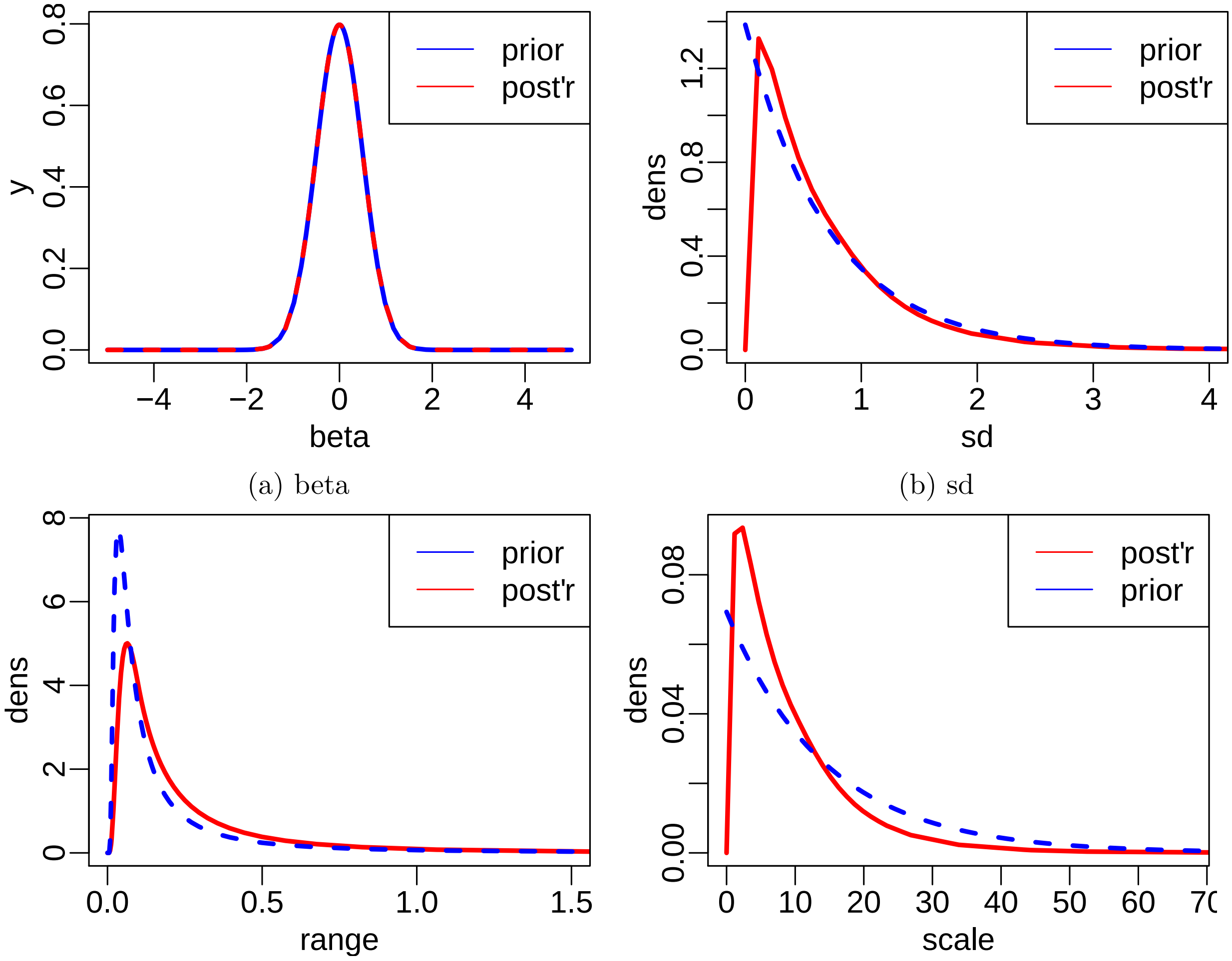
Figure 7: categorical

```
if(all(havePackages) &  fact > 1 ) {

# prior for observation standard deviation
  swissFit =  glgm( formula="lograin",data=swissRain, grid=Ncell,
    covariates=swissAltitude, family="gaussian", buffer=20000,
    prior=list(sd=0.5, range=200000, sdObs=1),
    control.inla = list(strategy='gaussian')
  )
}
```

a model with little data, posterior should be same as prior

```
  data2 = SpatialPointsDataFrame(cbind(c(1,0), c(0,1)),
    data=data.frame(y=c(0,0), offset=c(-50,-50), x=c(-1,1)))

if(all(havePackages) & fact > 1) {

resNoData = res = glgm(
  data=data2, grid=Ncell,
    formula=y~1 + x+offset(offset),
    prior = list(sd=0.5, range=0.1),
    family="poisson",
    buffer=0.5,
    control.fixed=list(
      mean.intercept=0, prec.intercept=1,
      mean=0,prec=4),
    control.mode = list(theta = c(0.651, 1.61), restart=TRUE),
    control.inla = list(strategy='gaussian')
  )
# beta
  plot(res$inla$marginals.fixed[['x']], col='blue', type='l',
    xlab='beta',lwd=3)
  xseq = res$inla$marginals.fixed[['x']][,'x']
  lines(xseq, dnorm(xseq, 0, 1/2),col='red',lty=2,lwd=3)
  legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))

# sd
    matplot(
      res$parameters$sd$posterior[,'x'],
      res$parameters$sd$posterior[,c('y','prior')],
      xlim = c(0, 4),
      type='l', col=c('red','blue'),xlab='sd',lwd=3, ylab='dens')
    legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))


# range
    matplot(
      res$parameters$range$posterior[,'x'],
      res$parameters$range$posterior[,c('y','prior')],
      xlim = c(0, 1.5),
      type='l', col=c('red','blue'),xlab='range',lwd=3, ylab='dens')
    legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))

        matplot(
      res$parameters$scale$posterior[,'x'],
      res$parameters$scale$posterior[,c('y','prior')],
      xlim = c(0, 2/res$parameters$summary['range','0.025quant']),
#      ylim = c(0, 10^(-3)), xlim = c(0,1000),
      type='l', col=c('red','blue'),xlab='scale',lwd=3, ylab='dens')
    legend("topright", col=c("red","blue"),lty=1,legend=c("post'r","prior"))

}
```

```
if(all(havePackages) & fact > 1) {


  resQuantile = res = glgm(
    data=data2,
    grid=25,
    formula=y~1 + x+offset(offset),
    prior = list(
      sd=c(lower=0.2, upper=2),
```



(a) beta

(b) sd

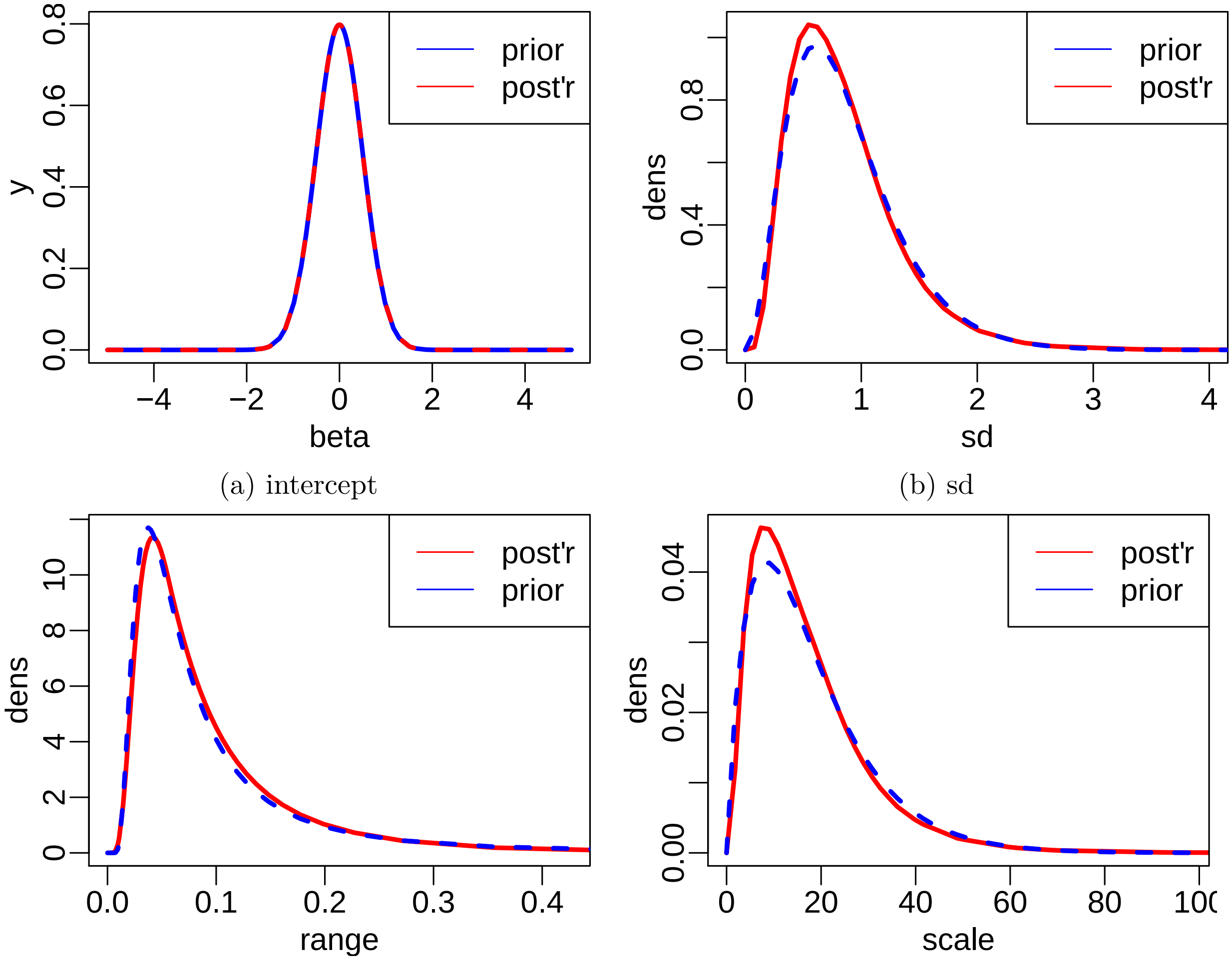
(c) range

(d) scale

Figure 8: no data, pc priors

```r
   range=c(lower=0.02, upper=0.5)),
  family="poisson", buffer=1,
  control.fixed=list(
   mean.intercept=0, prec.intercept=1,
   mean=0,prec=4),
  control.inla = list(strategy='gaussian')
 )

# beta
 plot(res$inla$marginals.fixed[['x']], col='blue', type='l',
  xlab='beta',lwd=3)
 xseq = res$inla$marginals.fixed[['x']][,'x']
 lines(xseq, dnorm(xseq, 0, 1/2),col='red',lty=2,lwd=3)
 legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))

# sd
  matplot(
   res$parameters$sd$posterior[,'x'],
   res$parameters$sd$posterior[,c('y','prior')],
   xlim = c(0, 4),
   type='l', col=c('red','blue'),xlab='sd',lwd=3, ylab='dens')
  legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))


# range
  matplot(
   res$parameters$range$posterior[,'x'],
   res$parameters$range$posterior[,c('y','prior')],
   xlim = c(0, 1.2*res$parameters$summary['range','0.975quant']),
#   xlim = c(0, 1), ylim = c(0,5),
   type='l', col=c('red','blue'),xlab='range',lwd=3, ylab='dens')
  legend("topright", col=c("red","blue"),lty=1,legend=c("post'r","prior"))

# scale
  matplot(
   res$parameters$scale$posterior[,'x'],
   res$parameters$scale$posterior[,c('y','prior')],
   xlim = c(0, 2/res$parameters$summary['range','0.025quant']),
#   ylim = c(0, 10^(-3)), xlim = c(0,1000),
   type='l', col=c('red','blue'),xlab='scale',lwd=3, ylab='dens')
  legend("topright", col=c("red","blue"),lty=1,legend=c("post'r","prior"))

}
```

No data, legacy priors



(a) intercept

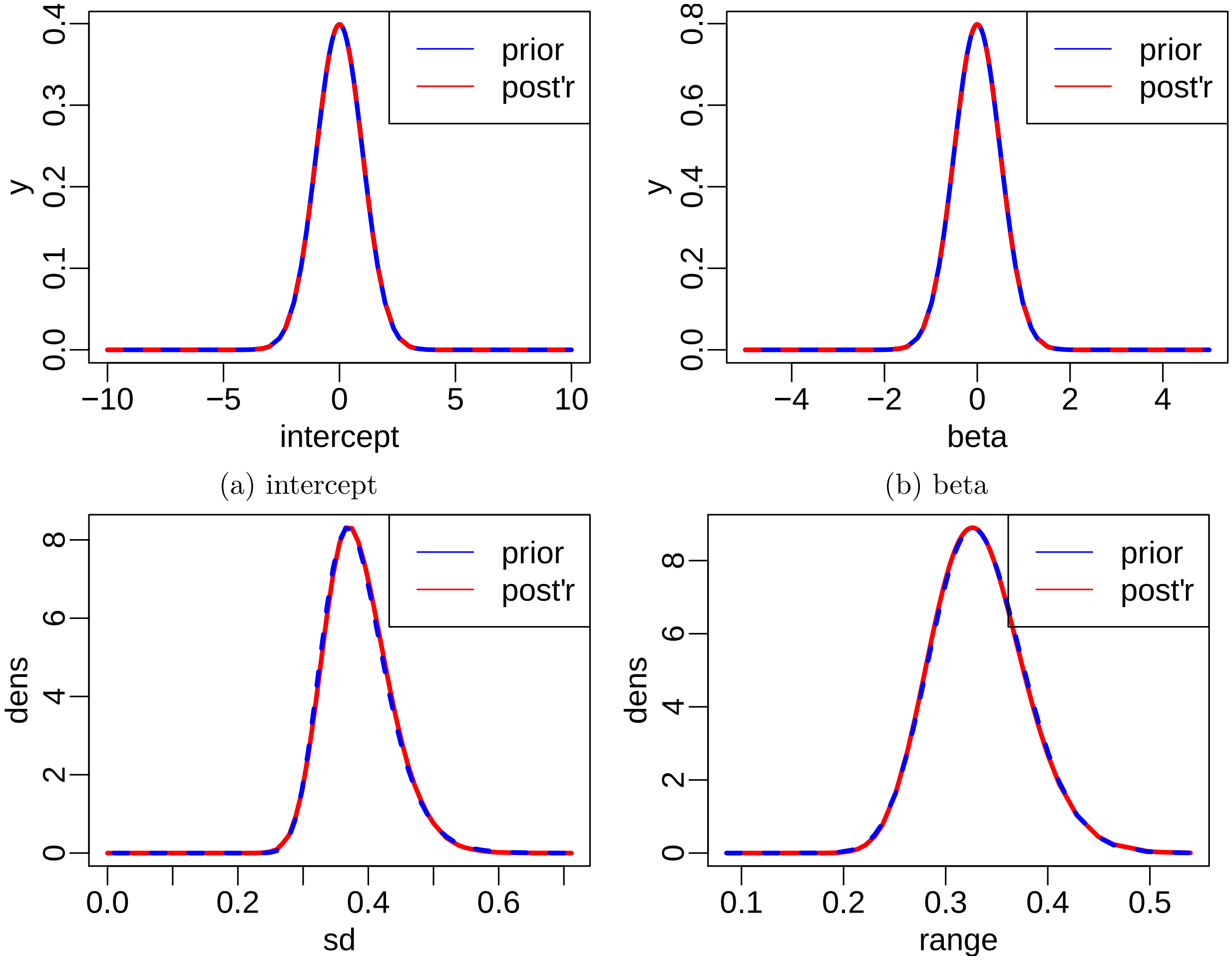
(b) sd

(c) range

(d) scale

Figure 9: no data quantile priors

```
if(all(havePackages) & fact > 1) {

resLegacy = res = glgm(data=data2,
    grid=20,
    formula=y~1 + x+offset(offset),
    priorCI = list(
      sd=c(lower=0.3,upper=0.5),
      range=c(lower=0.25, upper=0.4)),
    family="poisson",
    buffer=0.5,
    control.fixed=list(
      mean.intercept=0,
      prec.intercept=1,
      mean=0, prec=4),
    control.inla = list(strategy='gaussian'),
    control.mode=list(theta=c(2, 2),restart=TRUE)
  )


# intercept
  plot(res$inla$marginals.fixed[['(Intercept)']], col='blue', type='l',
    xlab='intercept',lwd=3)
  xseq = res$inla$marginals.fixed[['(Intercept)']][,'x']
  lines(xseq, dnorm(xseq, 0, 1),col='red',lty=2,lwd=3)
  legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))

# beta
  plot(res$inla$marginals.fixed[['x']], col='blue', type='l',
    xlab='beta',lwd=3)
  xseq = res$inla$marginals.fixed[['x']][,'x']
  lines(xseq, dnorm(xseq, 0, 1/2),col='red',lty=2,lwd=3)
  legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))

# sd
    matplot(
      res$parameters$sd$posterior[,'x'],
      res$parameters$sd$posterior[,c('y','prior')],
      type='l', col=c('red','blue'),xlab='sd',lwd=3, ylab='dens')
    legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))


# range
    matplot(
      res$parameters$range$posterior[,'x'],
```



(a) intercept

(b) beta

(c) sd

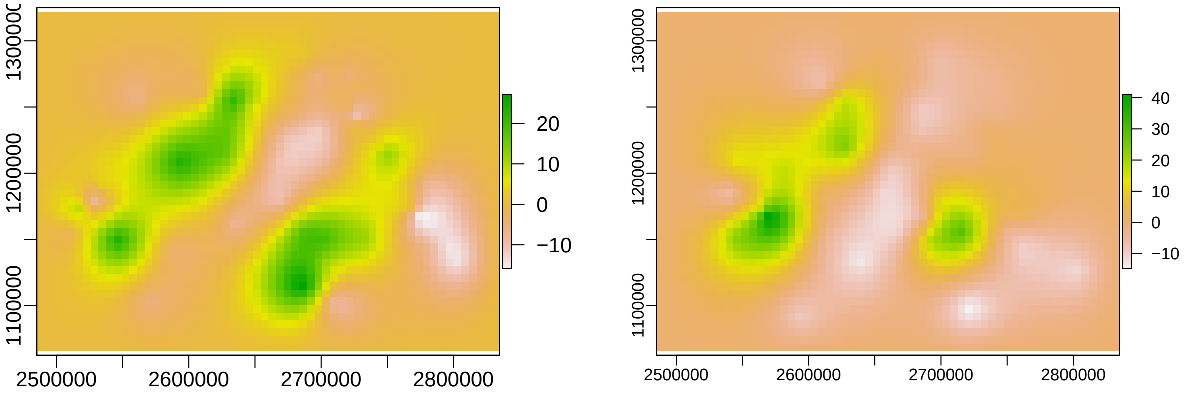
(d) range

Figure 10: No data, legacy priors

```
      res$parameters$range$posterior[,c('y','prior')],
      type='l', col=c('red','blue'),xlab='range',lwd=3, ylab='dens')
    legend("topright", col=c("blue","red"),lty=1,legend=c("prior","post'r"))
}
```

specifying spatial formula

```
swissRain$group = 1+rbinom(length(swissRain), 1, 0.5)
theGrid = squareRaster(swissRain, Ncell, buffer=30*1000)
if(all(havePackages)) {
  swissFit = glgm(
    formula = rain ~ 1,
    data=swissRain,
    grid=theGrid,
    family="gaussian",
```



(a) one

(b) two

Figure 11: spatial formula provided

```
    spaceFormula = ~ f(space, model='matern2d',
      nrow = nrow(theGrid), ncol = ncol(theGrid),
      nu = 1, replicate = group),
      control.inla = list(strategy='gaussian')
)

  swissFit$rasterTwo = setValues(
    swissFit$raster[[1:2]],
    as.matrix(swissFit$inla$summary.random$space[
      ncell(theGrid)+values(swissFit$raster[['space']]),
      c('mean','0.5quant')]))
  plot(swissFit$raster[['random.mean']])

  plot(swissFit$rasterTwo[['mean']])
}
```