Various GLGM examples

Patrick Brown

September 22, 2023

This vignette is a bunch of examples, its primary purpose is to test the glgm function.

The data

```
library("geostatsp")

## Loading required package: Matrix

## Loading required package: terra

## terra 1.7.41

##
## Attaching package: 'terra'

## The following object is masked from 'package:knitr':
##
##     spin

data('swissRain')
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)
swissBorder = unwrap(swissBorder)
swissRain$lograin = log(swissRain$rain)

swissAltitudeCrop = mask(swissAltitude,swissBorder)
```

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

```
if(!exists('fact')) fact = 1
fact

## [1] 1

(Ncell = round(25*fact))
```

```
## [1] 25
```

model with standard formula

```
swissFit =  glgm(
    formula = lograin~ CHE_alt,
    data = swissRain,
    grid = Ncell,
    buffer = 10*1000,
    covariates=swissAltitudeCrop,
    family="gaussian",
    prior = list(
      sd=c(1,0.5),
      sdObs = 1,
      range=c(500000, 0.5)),
    control.inla = list(strategy='gaussian')
  )
```

parameters

```
if(length(swissFit$parameters)) {
    knitr::kable(swissFit$parameters$summary[,c(1,3,5)], digits=3)
} else {
    print("INLA was not run, install the INLA package to see results")
}
```

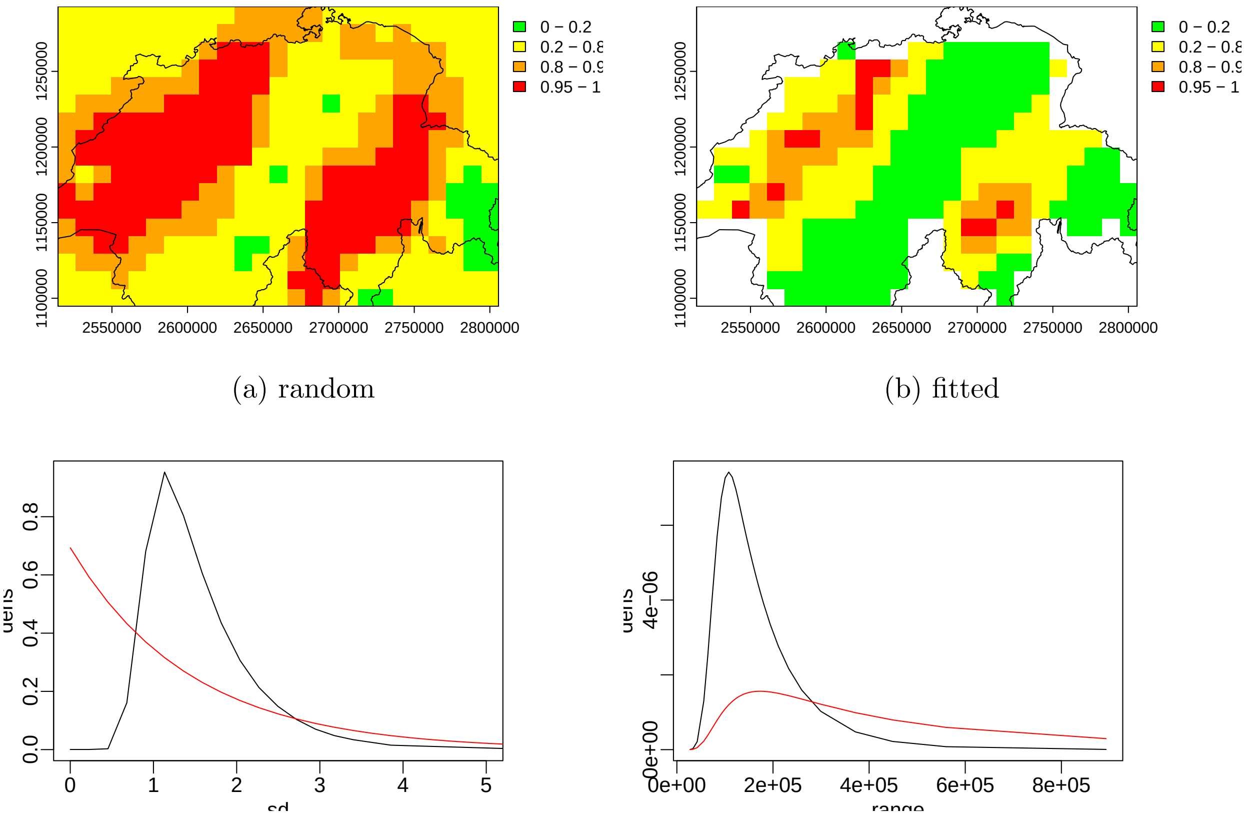
	mean	0.025quant	0.975quant
(Intercept)	2.262	1.584	2.904
CHE alt	0.000	0.000	0.000
range/1000	173.865	64.419	448.563
sdNugget	0.312	0.190	0.468
sd	1.554	0.746	3.258

Exceedance probabilities

```
if(length(swissFit$parameters)) {
  swissExc = excProb(
    x=swissFit,  random=TRUE,
    threshold=0)
}

if(length(swissFit$parameters)) {
  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')],
    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))
altMat = cbind(altSeq[-length(altSeq)], altSeq[-1], seq(1,length(altSeq)-1))

swissAltCut = classify(
  swissAltitudeCrop,
  altMat
)
names(swissAltCut) = 'bqrnt'


  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),
```



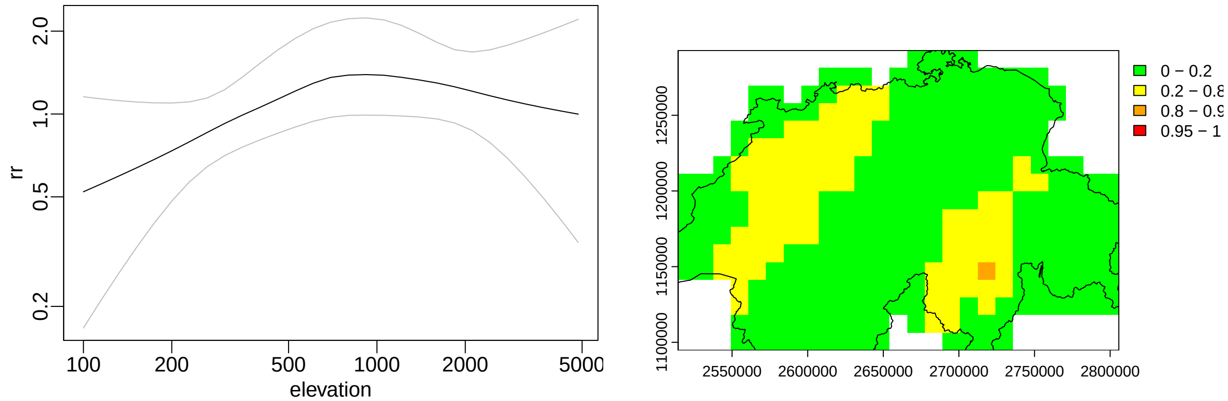
(a) random

(b) fitted

(c) sd

(d) range

Figure 1: Swiss rain as in help file



(a) elevation effect

(b) fitted

Figure 2: Swiss rain elevation rw2

```
      range=c(50000,500000),
      sdObs = c(u=1, alpha=0.4)),
    control.inla=list(strategy='gaussian')
  )

if(length(swissFitNp$parameters)) {
  knitr::kable(swissFitNp$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

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

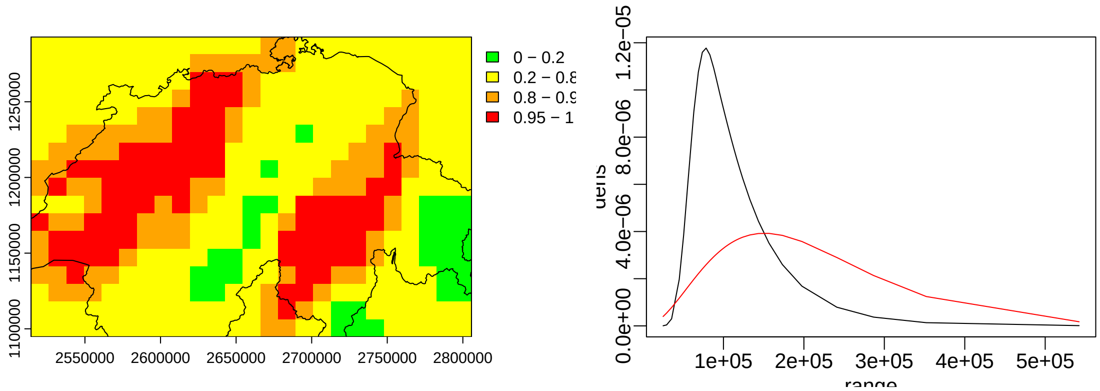
	mean	0.025quant	0.5quant	0.975quant	meanExp
(Intercept)	2.4817	1.7085	2.5130	3.0955	12.6072
CHE alt	-0.0001	-0.0004	-0.0001	0.0002	1.0127
range/1000	99.4377	43.7274	87.1941	231.7933	NA
sdNugget	0.3215	0.1989	0.3022	0.5028	NA
sd	0.9251	0.5925	0.8680	1.4268	NA

intercept only, add a covariate just to confuse glgm.

```
  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))))
  )

  if(length(swissFit$parameters)) {

  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),
    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, ID=FALSE)
```



(a) exc prob

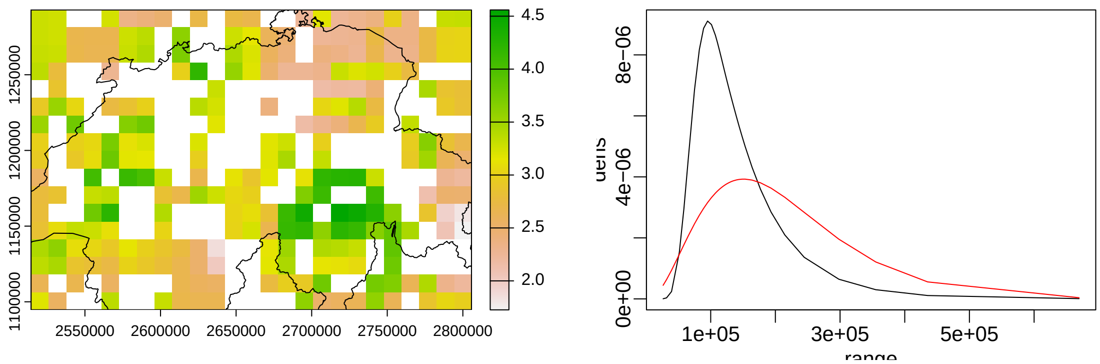
(b) range

Figure 3: Swiss intercept only

```
swissLandType = unwrap(swissLandType)
  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))))
  )

  if(length(swissFit$parameters)) {
  knitr::kable(swissFit$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')
}
```

formula, named list elements

```
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)),
```



(a) predict.mean

(b) range

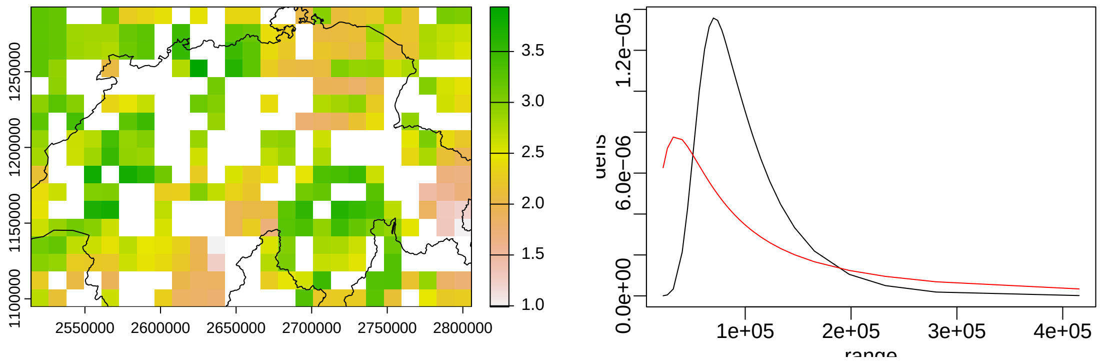
Figure 4: covaraites in data

```
    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))))
  )
if(length(swissFit$parameters))
    swissFit$parameters$summary[,c(1,3,5)]
```

```
##                         mean    0.025quant   0.975quant
## (Intercept)  2.456456e+00  1.6858671882 3.081686e+00
## elev        -9.737333e-05 -0.0003999165 2.049723e-04
## range/1000   1.164995e+02 48.7639685747 2.802470e+02
## sdNugget     3.482916e-01  0.2262350647 5.092120e-01
## sd           1.023735e+00  0.6178876079 1.669214e+00
```

categorical covariates

```
swissFit =  glgm(
    formula = lograin ~ elev + factor(land),
    data = swissRain, grid = Ncell,
    covariates=list(elev=swissAltitude,land=swissLandType),
    family="gaussian", buffer=20000,
    prior=list(sd=c(0.2, 0.5), range=c(100000,0.5)),
    control.inla=list(strategy='gaussian'),
    control.family=list(hyper=list(
      prec=list(prior="loggamma",
          param=c(.1, .1))))
  )
if(length(swissFit$parameters)) {

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



(a) map

(b) range

Figure 5: categorical covariates

```
  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

swissFitMissing =  glgm(rain ~ elev + land,swissRain,  Ncell,
    covariates=list(elev=swissAltitude,land=swissLandType),
    family="gaussian", buffer=20000,
    prior=list(sd=c(0.2, 0.5), range=c(100000,0.5)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
)
if(length(swissFitMissing$parameters))
      knitr::kable(swissFitMissing$parameters$summary[,1:5], digits=3)
```

	mean	sd	0.025quant	0.5quant	0.975quant
(Intercept)	27.184	3.245	20.792	27.188	33.550
elev	-0.005	0.003	-0.011	-0.005	0.002
landMixed forests	-4.250	3.252	-10.629	-4.255	2.156
landGrasslands	-3.330	4.894	-12.936	-3.335	6.305
landCroplands	-9.537	4.208	-17.791	-9.543	-1.248
landUrban and built-up	-8.066	5.473	-18.805	-8.073	2.711
landEvergreen needleleaf forest	-11.999	6.264	-24.285	-12.008	0.338
landWater bodies	-15.823	8.039	-31.580	-15.838	0.019
landDeciduous needleleaf forest	-8.986	8.002	-24.684	-8.996	6.772
landDeciduous broadleaf forest	8.308	7.998	-7.417	8.309	24.023
landOpen shrublands	-11.591	11.022	-33.197	-11.609	10.117
landPermanent Wetlands	-21.620	10.867	-42.887	-21.650	-0.182
range/1000	191.073	266.598	18.549	112.400	855.138
sdNugget	11.662	-3.056	9.647	11.213	13.147
sd	0.008	0.000	0.003	0.007	0.020

covariates in data, factors

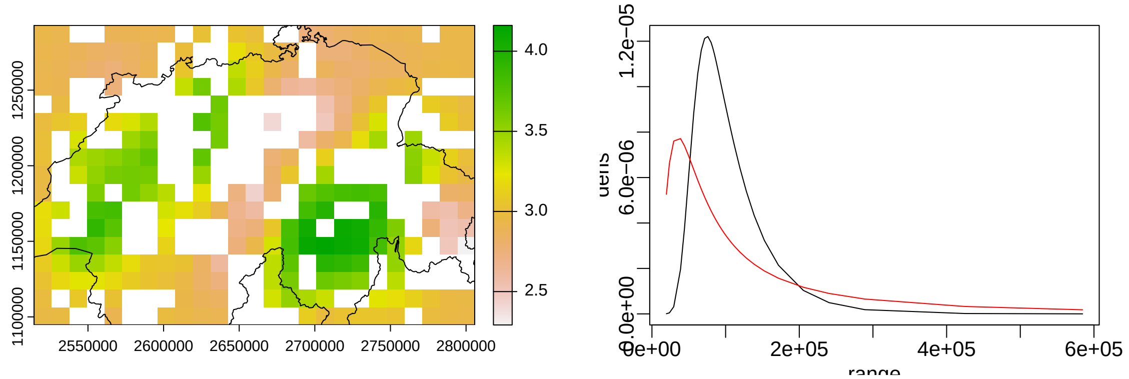
```
newdat = swissRain
newdat$landOrig = extract(swissLandType, swissRain, ID=FALSE)
newdat$landRel = relevel(newdat$landOrig, 'Mixed forests')

swissFit =  glgm(
    formula = lograin~ elev + landOrig,
    data=newdat,
    covariates=list(elev = swissAltitude),
    grid=squareRaster(swissRain,Ncell),
    family="gaussian", buffer=0,
    prior=list(sd=c(0.2, 0.5), range=c(100000,0.5)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
)

swissFitR =  glgm(
    formula = lograin~ elev + landRel,
    data=newdat,
    grid=squareRaster(swissRain,Ncell),
    covariates=list(elev = swissAltitude, landRel = swissLandType),
    family="gaussian", buffer=0,
    prior=list(sd=c(0.2, 0.5), range=c(100000,0.5)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
  )
```

```
levels(newdat$landOrig)

##  [1] "Water bodies"                       "Evergreen needleleaf forest"
##  [3] "Evergreen broadleaf forest"         "Deciduous needleleaf forest"
##  [5] "Deciduous broadleaf forest"         "Mixed forests"
##  [7] "Closed shrublands"                  "Open shrublands"
##  [9] "Woody savannas"                     "Savannas"
## [11] "Grasslands"                         "Permanent Wetlands"
## [13] "Croplands"                          "Urban and built-up"
## [15] "Cropland/natural vegetation mosaic" "Snow and ice"
## [17] "Barren or sparsely vegetated"

levels(newdat$landRel)

##  [1] "Mixed forests"                      "Water bodies"
##  [3] "Evergreen needleleaf forest"        "Evergreen broadleaf forest"
##  [5] "Deciduous needleleaf forest"        "Deciduous broadleaf forest"
##  [7] "Closed shrublands"                  "Open shrublands"
##  [9] "Woody savannas"                     "Savannas"
## [11] "Grasslands"                         "Permanent Wetlands"
## [13] "Croplands"                          "Urban and built-up"
## [15] "Cropland/natural vegetation mosaic" "Snow and ice"
## [17] "Barren or sparsely vegetated"

if(length(swissFit$parameters)) {
    levels(swissFit$inla$.args$data$landOrig)
    levels(swissFitR$inla$.args$data$landRel)
  }

##  [1] "Cropland/natural vegetation mosaic" "Mixed forests"
##  [3] "Grasslands"                         "Croplands"
##  [5] "Urban and built-up"                 "Evergreen needleleaf forest"
##  [7] "Water bodies"                       "Deciduous needleleaf forest"
##  [9] "Deciduous broadleaf forest"         "Open shrublands"
## [11] "Permanent Wetlands"

if(length(swissFit$parameters)) {
    knitr::kable(swissFit$parameters$summary[,c(1,3,5)], digits=3)
    knitr::kable(swissFitR$parameters$summary[,c(1,3,5)], digits=3)
}
```

	mean	0.025quant	0.975quant
(Intercept)	3.156	2.645	3.648
elev	-0.001	-0.001	0.000
landRelMixed forests	-0.186	-0.466	0.092
landRelGrasslands	-0.059	-0.479	0.360
landRelCroplands	-0.388	-0.735	-0.047
landRelUrban and built-up	-0.685	-1.265	-0.101
landRelEvergreen needleleaf forest	-0.598	-1.192	-0.027
landRelWater bodies	-0.997	-1.747	-0.246
landRelDeciduous needleleaf forest	-0.594	-1.315	0.123
landRelDeciduous broadleaf forest	0.330	-0.354	1.040
landRelOpen shrublands	-0.134	-1.235	0.973
landRelPermanent Wetlands	-2.636	-3.652	-1.625
range/1000	109.925	44.811	255.443
sdNugget	0.355	0.233	0.489
sd	0.803	0.479	1.331

covariates are in data, interactions

```
newdat = swissRain
newdat$elev = extract(swissAltitude, swissRain, ID=FALSE)

swissFit =  glgm(
    formula = lograin~ elev : land,
    data=newdat,
    grid=squareRaster(swissRain,Ncell),
    covariates=list(land=swissLandType),
    family="gaussian", buffer=0,
    prior=list(sd=c(0.2, 0.5), range=c(100000,0.5)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
)
if(length(swissFit$parameters)) {
  knitr::kable(swissFit$parameters$summary[,c(1,3,5)], digits=3)
}
```



(a) map (b) range

Figure 6: interactions

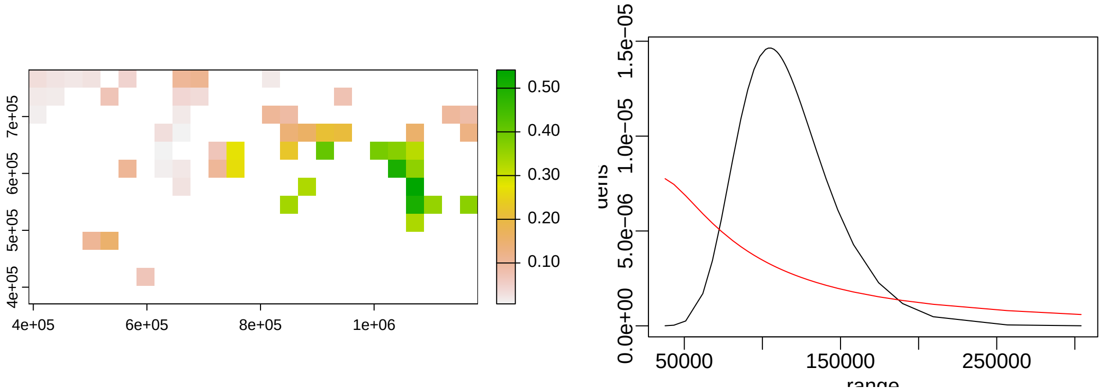
	mean	0.025quant	0.975quant
(Intercept)	2.936	2.446	3.416
elev:landCropland/natural vegetation mosaic	0.000	-0.001	0.000
elev:landMixed forests	-0.001	-0.001	0.000
elev:landGrasslands	0.000	-0.001	0.000
elev:landCroplands	-0.001	-0.002	0.000
elev:landUrban and built-up	-0.001	-0.002	0.000
elev:landEvergreen needleleaf forest	-0.001	-0.001	-0.001
elev:landWater bodies	-0.002	-0.004	0.000
elev:landDeciduous needleleaf forest	-0.001	-0.001	0.000
elev:landDeciduous broadleaf forest	0.000	-0.001	0.001
elev:landOpen shrublands	-0.001	-0.001	0.000
elev:landPermanent Wetlands	-0.010	-0.013	-0.006
range/1000	106.197	44.229	240.419
sdNugget	0.348	0.233	0.473
sd	0.770	0.459	1.270

```
if(length(swissFit$parameters)) {
   plot(swissFit$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')
}
```

categorical tests

```
data('loaloa')
```

```
  loaloa = unwrap(loaloa)
  ltLoa = unwrap(ltLoa)
  elevationLoa = unwrap(elevationLoa)
  eviLoa = unwrap(eviLoa)

  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 = classify(ltLoa, rcl)
levels(ltLoaR) = levels(ltLoa)

elevationLoa = elevationLoa - 750
elevLow = min(elevationLoa, 0)
elevHigh = max(elevationLoa, 0)

eviLoa2 = (eviLoa - 1e7)/1e6

covList = list(elLow = elevLow, elHigh = elevHigh,
    land = ltLoaR, evi=eviLoa2)

 loaFit = glgm(
    y ~ 1 + 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 = 100*1000),
    control.inla = list(strategy='gaussian')
    )

if(length(loaFit$parameters)) {
  knitr::kable(loaFit$par$summary[,c(1,3,5)], digits=3)
}
```



(a) predict

(b) range

Figure 7: categorical

	mean	0.025quant	0.975quant
(Intercept)	-5.271	-7.156	-3.391
landWoody Savannas	-0.157	-0.659	0.338
landCropland/Natural Vegetation Mosaics	0.136	-0.290	0.562
evi	0.115	0.068	0.163
elHigh	-0.003	-0.005	-0.002
elLow	0.003	0.001	0.004
range/1000	116.869	67.995	189.760
sd	0.694	0.426	1.057
sd villageID	0.652	0.528	0.745

```
if(length(loaFit$parameters)) {
  plot(loaFit$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')
}
```

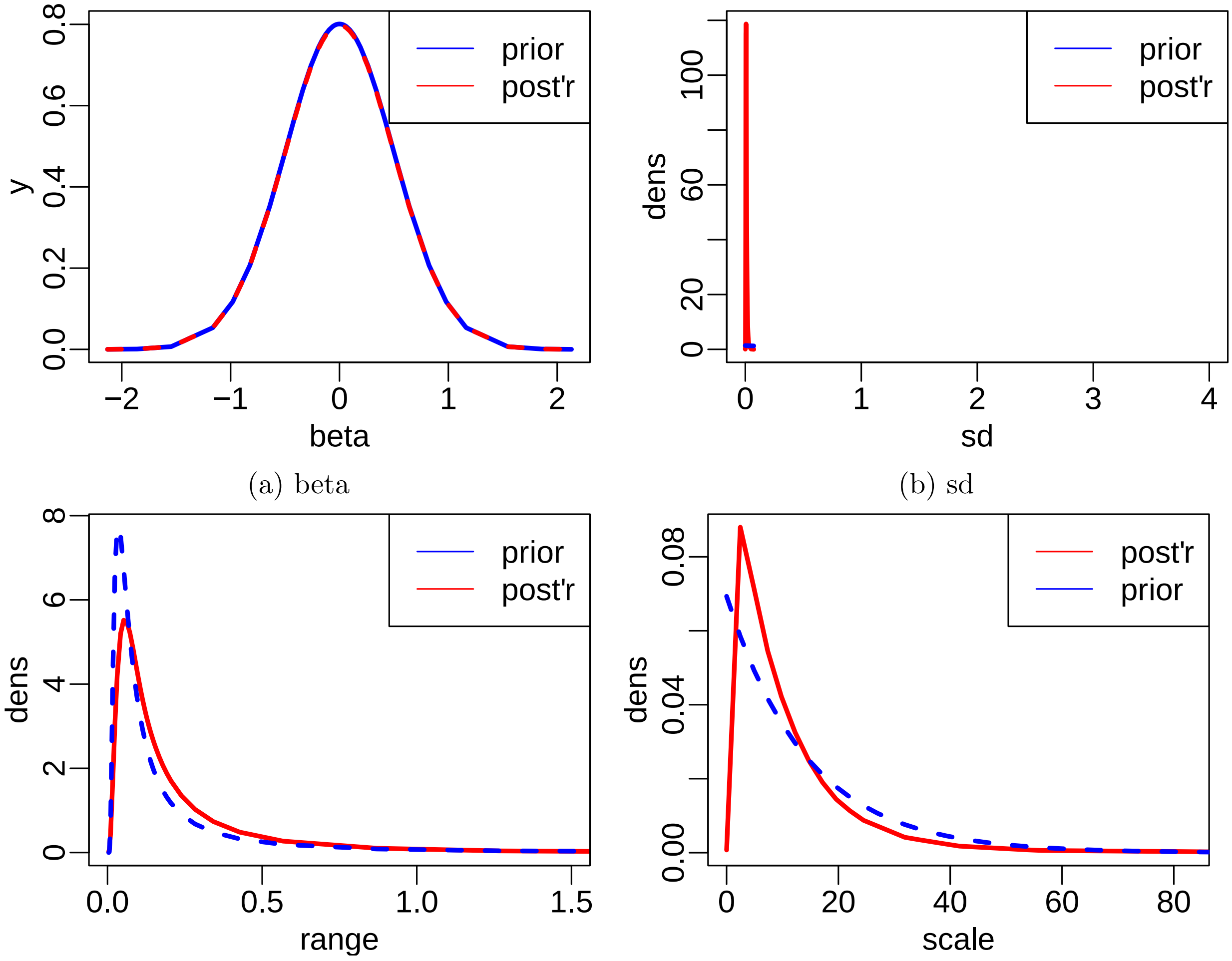
prior for observation standard deviation

```
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')
)
```

no data checks

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

```
data2 = vect(cbind(c(1,0), c(0,1)),
    atts=data.frame(y=c(0,0), offset=c(-50,-50), x=c(-1,1)),
    crs = '+proj=merc')


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')
  )

if(length(res$parameters)) {
# beta
  plot(res$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')
```



(a) beta

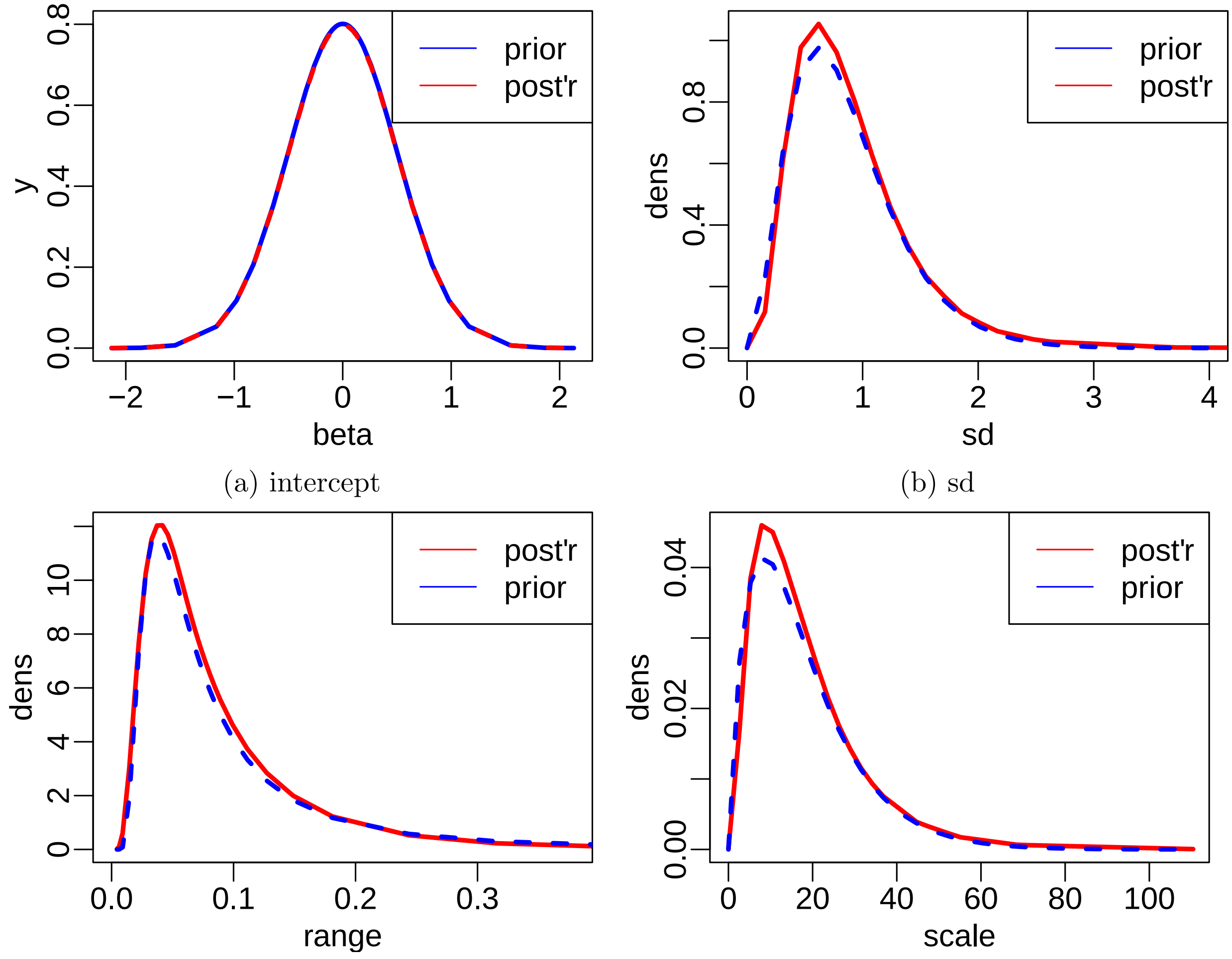
(b) sd

(c) range

(d) scale

Figure 8: no data, pc priors

```
    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"))
}


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

if(length(res$parameters)) {
# beta
  plot(res$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"))
}
```



(a) intercept

(b) sd

(c) range

(d) scale

Figure 9: no data quantile priors

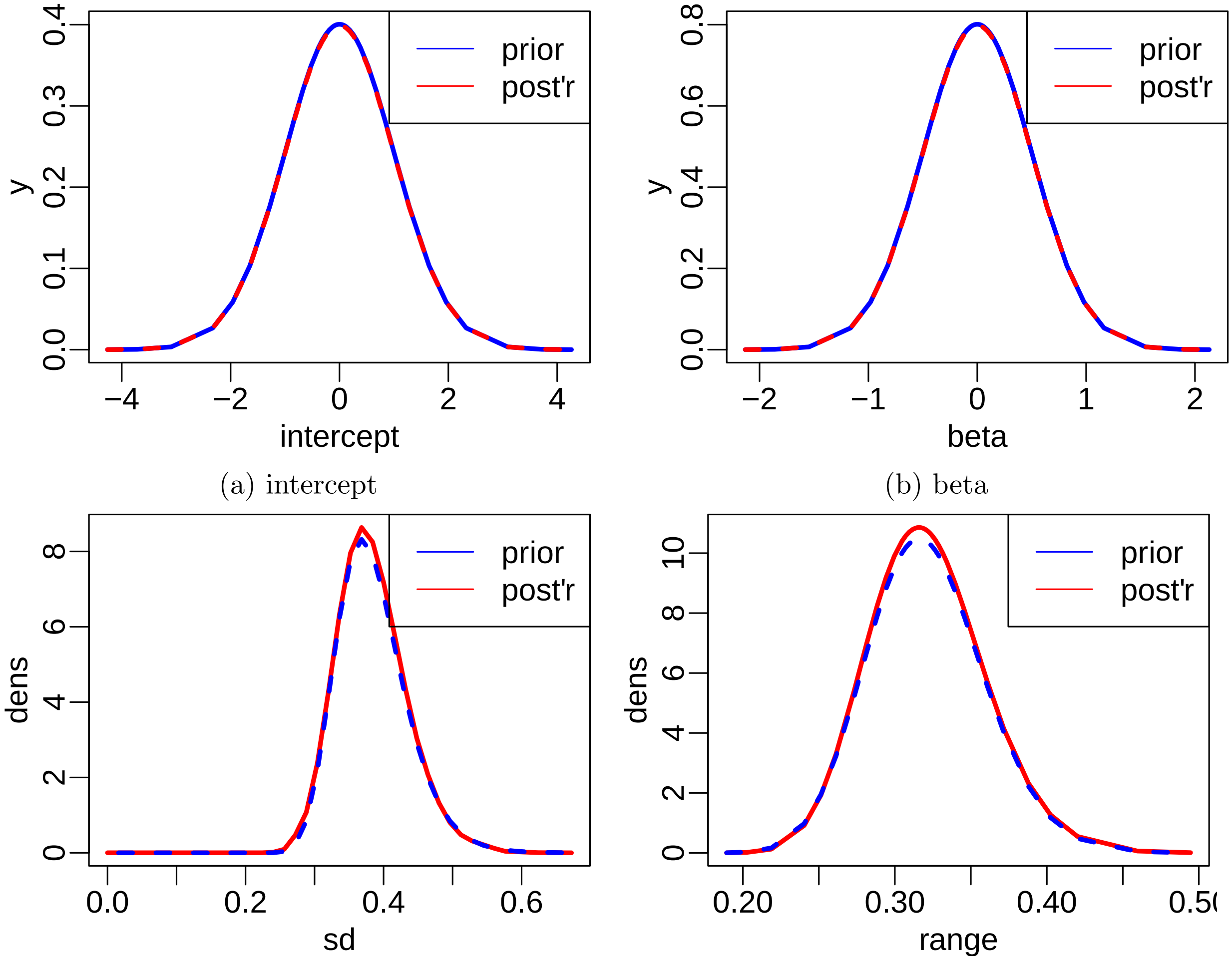
No data, legacy priors

```
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)
```

```
  )

if(length(res$parameters)) {
# intercept
  plot(res$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$posterior[,'x'],
      res$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=10*1000)

swissFit = glgm(
    formula = rain ~ 1,
    data=swissRain,
    grid=theGrid,
    family="gaussian",
    spaceFormula = ~ f(space, model='matern2d',
      nrow = nrow(theGrid), ncol = ncol(theGrid),
```



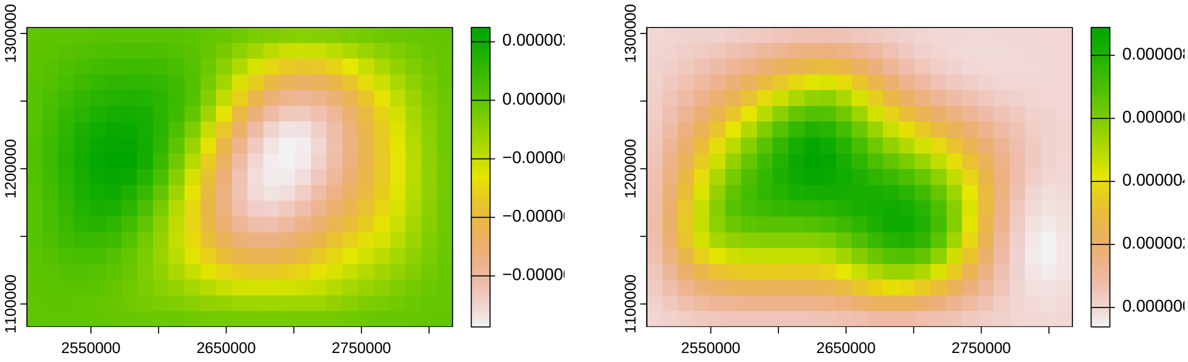
(a) intercept

(b) beta

(c) sd

(d) range

Figure 10: No data, legacy priors



(a) one

(b) two

Figure 11: spatial formula provided

```
      nu = 1, replicate = group),
      control.inla = list(strategy='gaussian'),
)

if(length(swissFit$parameters)) {
  swissFit$rasterTwo = setValues(
    rast(swissFit$raster, nlyrs=2),
    as.matrix(swissFit$inla$summary.random$space[
      ncell(theGrid)+values(swissFit$raster[['space']]),
      c('mean','0.5quant')]))
  plot(swissFit$raster[['random.mean']])

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