

Various GLGM examples

Patrick Brown

February 26, 2026

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

The data

```
library("geostatssp")
## Loading required package: Matrix
## Loading required package: terra
## terra 1.8.93
##
## 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 = geostatsp::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) | 2.287   | 1.597      | 2.935      |
| CHE alt     | 0.000   | 0.000      | 0.000      |
| range/1000  | 182.703 | 58.298     | 527.749    |
| sdNugget    | 0.313   | 0.182      | 0.504      |
| sd          | 1.538   | 0.692      | 3.546      |



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 = geostatsp::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),

```

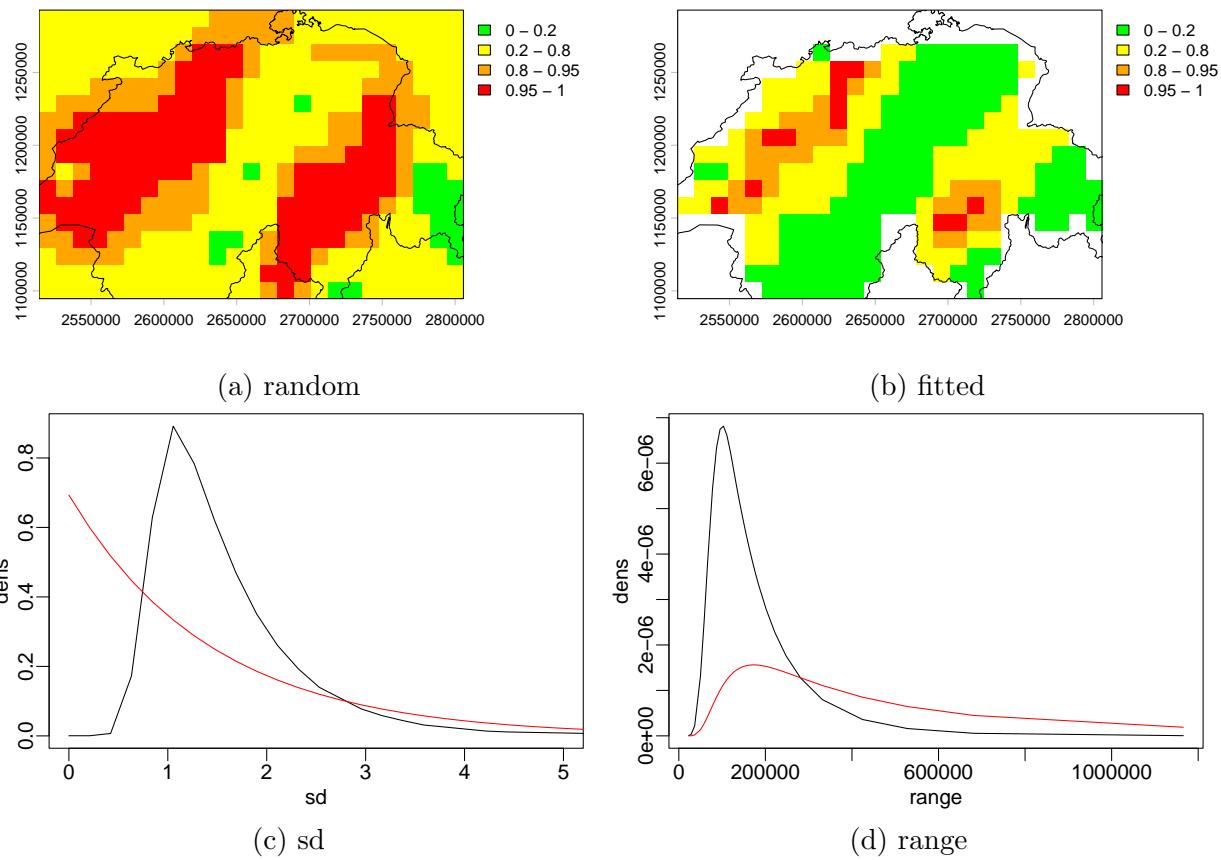


Figure 1: Swiss rain as in help file

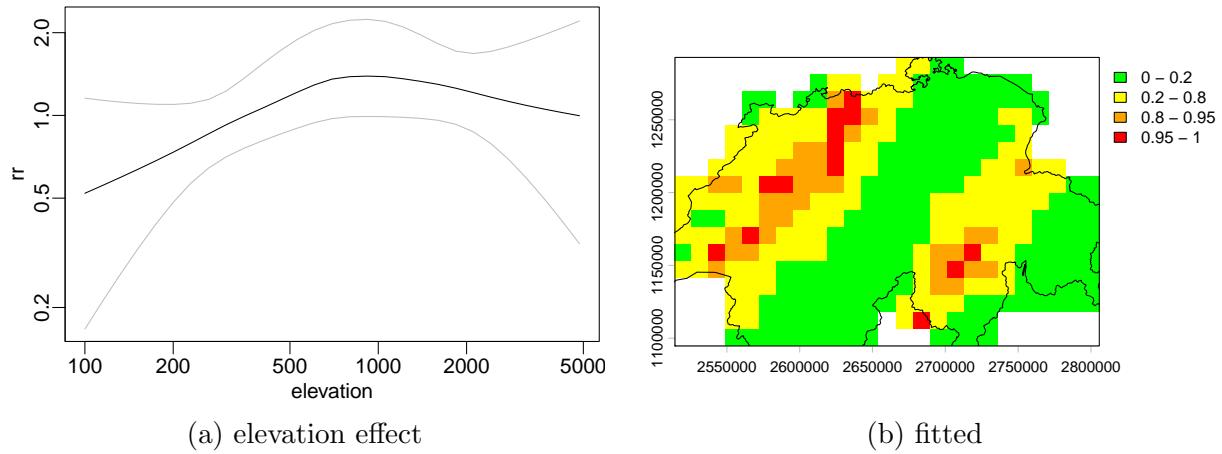


Figure 2: Swiss rain elevation rw2

```

range=c(50000,500000),
sd0bs = 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 = geostatsp::glgm("lograin", swissRain, Ncell,
  covariates=swissAltitude, family="gaussian", buffer=20000,
  priorCI=list(sd=c(0.2, 2), range=c(50000,500000), sd0bs = 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) | 2.4619 | 1.6556 | 2.4936 | 3.1013 | 12.4152 |
| CHE alt | -0.0001 | -0.0004 | -0.0001 | 0.0002 | 1.0125 |
| range/1000 | 109.2736 | 41.8909 | 92.5548 | 280.4207 | NA |
| sdNugget | 0.3229 | 0.1946 | 0.3043 | 0.5178 | NA |
| sd | 0.9631 | 0.5797 | 0.9011 | 1.6007 | NA |

intercept only, add a covariate just to confuse glgm.

```

swissFit = geostatsp::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)

```

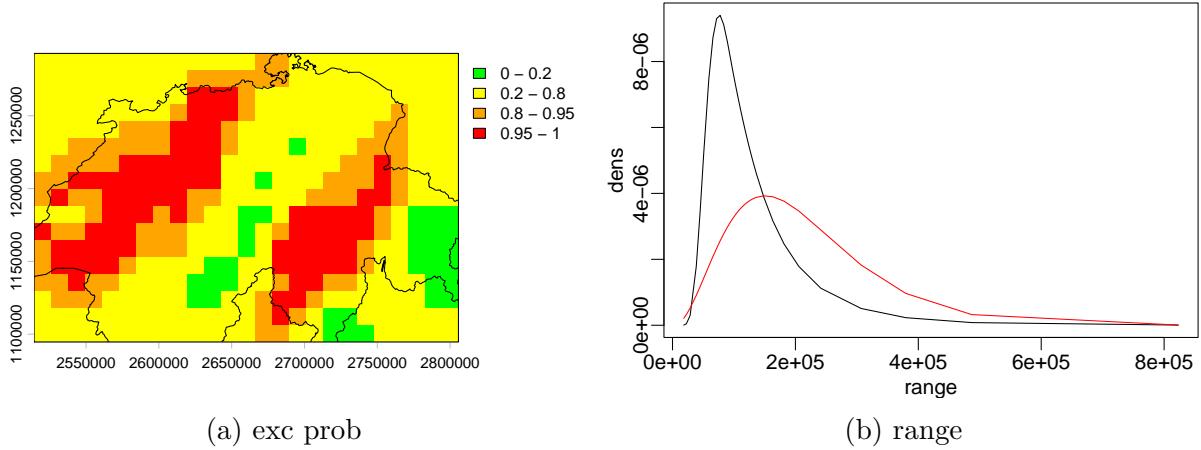


Figure 3: Swiss intercept only

```

swissLandType = unwrap(swissLandType)
swissFit = geostatsp::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 = geostatsp::glgm(lograin ~ elev,
  swissRain, Ncell,
  covariates=list(elev=swissAltitude),
  family="gaussian", buffer=20000,
  priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),

```

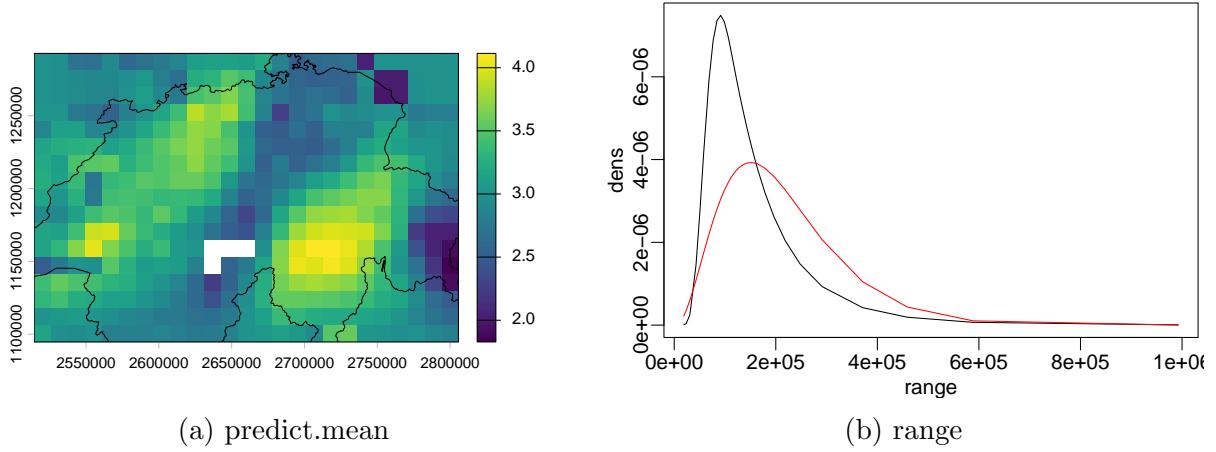


Figure 4: covaraites in data

```

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) 2.446812e+00  1.6362238825 3.093347e+00
## elev        -8.827999e-05 -0.0004181989 2.423870e-04
## range/1000   1.290901e+02  43.1238098601 3.659219e+02
## sdNugget     3.429462e-01  0.2148412410 5.213780e-01
## sd           1.030688e+00  0.5736352734 1.902397e+00

categorical covariates

swissFit = geostatsp::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)

```

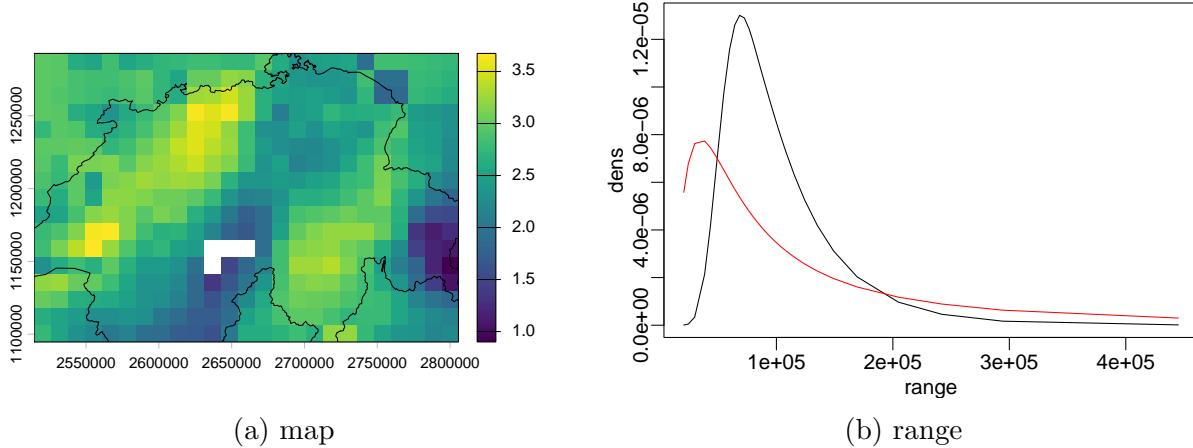


Figure 5: categorical covariates

```

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 = geostatsp::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) | 26.668 | 3.199 | 20.368 | 26.671 | 32.947 |
| elev | -0.004 | 0.003 | -0.011 | -0.004 | 0.002 |
| landMixed forests | -3.950 | 3.234 | -10.293 | -3.954 | 2.419 |
| landGrasslands | -3.190 | 4.933 | -12.871 | -3.195 | 6.519 |
| landCroplands | -9.280 | 4.016 | -17.160 | -9.285 | -1.371 |
| landUrban and built-up | -7.632 | 5.420 | -18.268 | -7.638 | 3.038 |
| landEvergreen needleleaf forest | -11.908 | 6.221 | -24.111 | -11.916 | 0.344 |
| landWater bodies | -15.161 | 7.963 | -30.772 | -15.175 | 0.527 |
| landDeciduous needleleaf forest | -8.342 | 7.981 | -24.000 | -8.351 | 7.370 |
| landDeciduous broadleaf forest | 8.633 | 7.953 | -7.002 | 8.635 | 24.257 |
| landOpen shrublands | -11.262 | 11.012 | -32.850 | -11.279 | 10.424 |
| landPermanent Wetlands | -21.046 | 10.787 | -42.160 | -21.074 | 0.229 |
| range/1000 | 217.073 | 329.940 | 19.786 | 121.593 | 1020.159 |
| sdNugget | 11.577 | -3.036 | 9.582 | 11.139 | 13.011 |
| sd | 0.009 | 0.000 | 0.003 | 0.008 | 0.025 |

covariates in data, factors

```

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

swissFit = geostatsp::glgm(
  formula = lograin~ elev + landOrig,
  data=newdat,
  covariates=list(elev = swissAltitude),
  grid=geostatsp::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 = geostatsp::glgm(
  formula = lograin~ elev + landRel,
  data=newdat,
  grid=geostatsp::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] "Mixed forests"                     "Water bodies"
## [3] "Evergreen needleleaf forest"       "Deciduous needleleaf forest"
## [5] "Deciduous broadleaf forest"        "Open shrublands"
## [7] "Grasslands"                       "Permanent Wetlands"
## [9] "Croplands"                        "Urban and built-up"
## [11] "Cropland/natural vegetation mosaic"

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) | 2.988 | 2.466 | 3.497 |
| elev | -0.001 | -0.001 | 0.000 |
| landRelWater bodies | -0.770 | -1.514 | -0.026 |
| landRelEvergreen needleleaf forest | -0.437 | -0.998 | 0.099 |
| landRelDeciduous needleleaf forest | -0.370 | -1.077 | 0.334 |
| landRelDeciduous broadleaf forest | 0.537 | -0.157 | 1.251 |
| landRelOpen shrublands | 0.095 | -0.980 | 1.176 |
| landRelGrasslands | 0.132 | -0.240 | 0.503 |
| landRelPermanent Wetlands | -2.453 | -3.448 | -1.464 |
| landRelCroplands | -0.180 | -0.527 | 0.162 |
| landRelUrban and built-up | -0.505 | -1.082 | 0.076 |
| landRelCropland/natural vegetation mosaic | 0.175 | -0.100 | 0.452 |
| range/1000 | 114.053 | 43.764 | 273.162 |
| sdNugget | 0.352 | 0.231 | 0.486 |
| sd | 0.804 | 0.462 | 1.386 |

covariates are in data, interactions

```

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

swissFit = geostatsp::glgm(
  formula = lograin~ elev : land,
  data=newdat,
  grid=geostatsp::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)
}

```

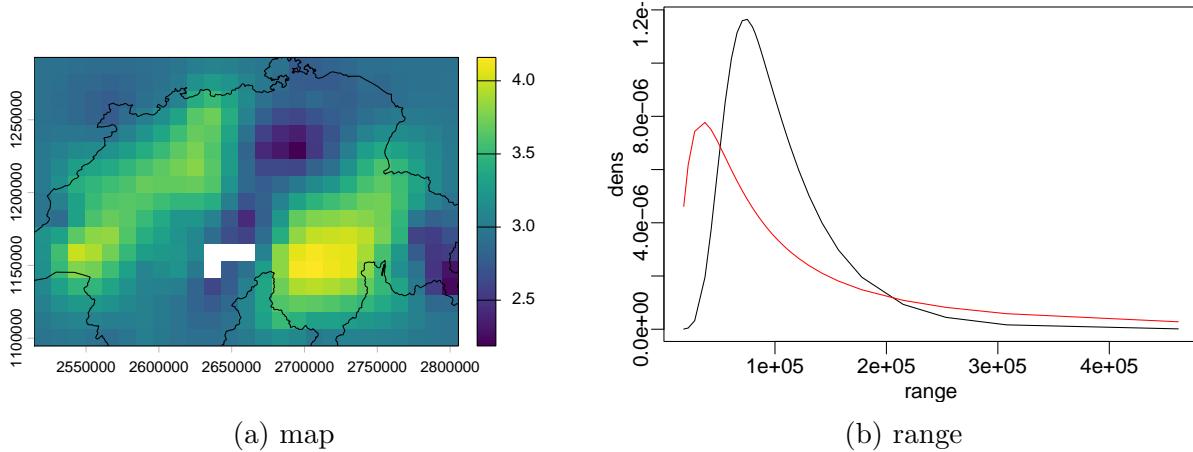


Figure 6: interactions

| | mean | 0.025quant | 0.975quant |
|---|---------|------------|------------|
| (Intercept) | 2.938 | 2.446 | 3.418 |
| 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 | 107.889 | 42.654 | 253.538 |
| sdNugget | 0.347 | 0.231 | 0.477 |
| sd | 0.772 | 0.450 | 1.314 |

```

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(
  # wetlands 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 mosaids
  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 = geostatsp::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)
}

```

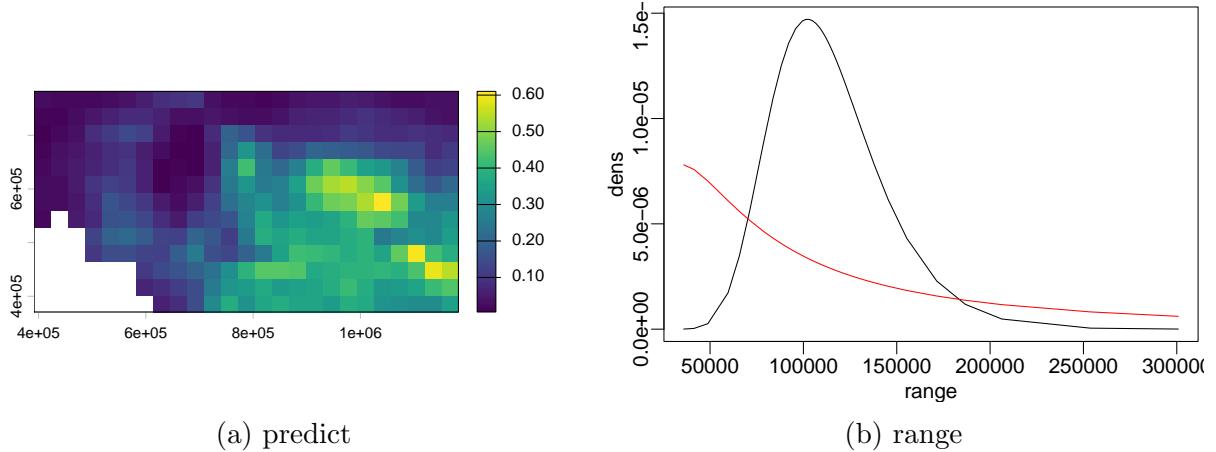


Figure 7: categorical

| | mean | 0.025quant | 0.975quant |
|---|---------|------------|------------|
| (Intercept) | -5.297 | -7.309 | -3.294 |
| landWoody Savannas | -0.077 | -0.575 | 0.416 |
| landCropland/Natural Vegetation Mosaics | 0.193 | -0.227 | 0.614 |
| evi | 0.117 | 0.067 | 0.168 |
| elHigh | -0.004 | -0.005 | -0.002 |
| elLow | 0.003 | 0.001 | 0.004 |
| range/1000 | 114.249 | 65.687 | 186.724 |
| sd | 0.681 | 0.428 | 1.014 |
| sd villageID | 0.627 | 0.506 | 0.722 |

```

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 = geostatsp::glgm( formula="lograin", data=swissRain, grid=Ncell,
  covariates=swissAltitude, family="gaussian", buffer=20000,
  prior=list(sd=0.5, range=200000, sd0bs=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 = geostatsp::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')
```

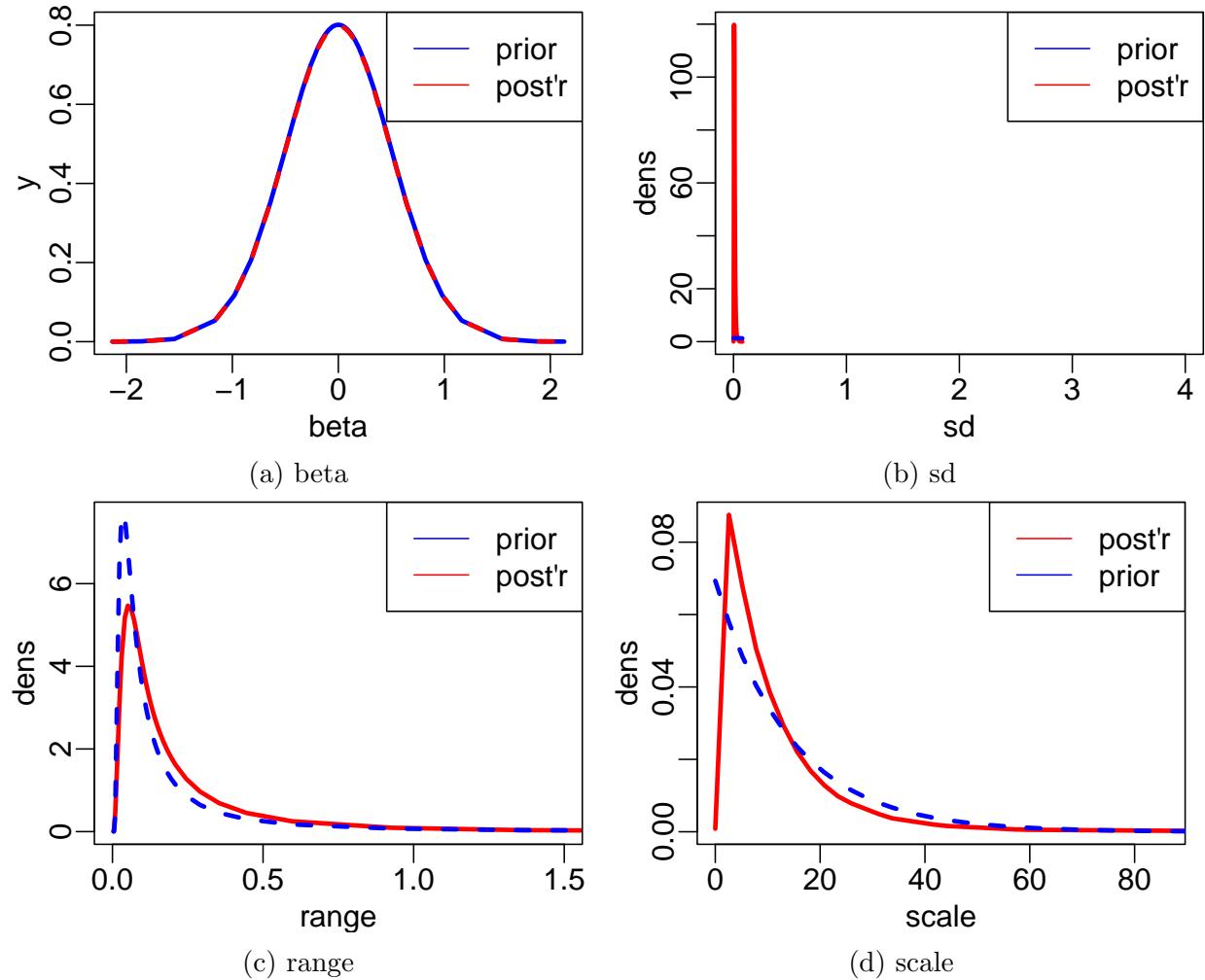


Figure 8: no data, pc priors

```

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 = geostatsp::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']),
    ylim = c(0, 1),
    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"))
}

```

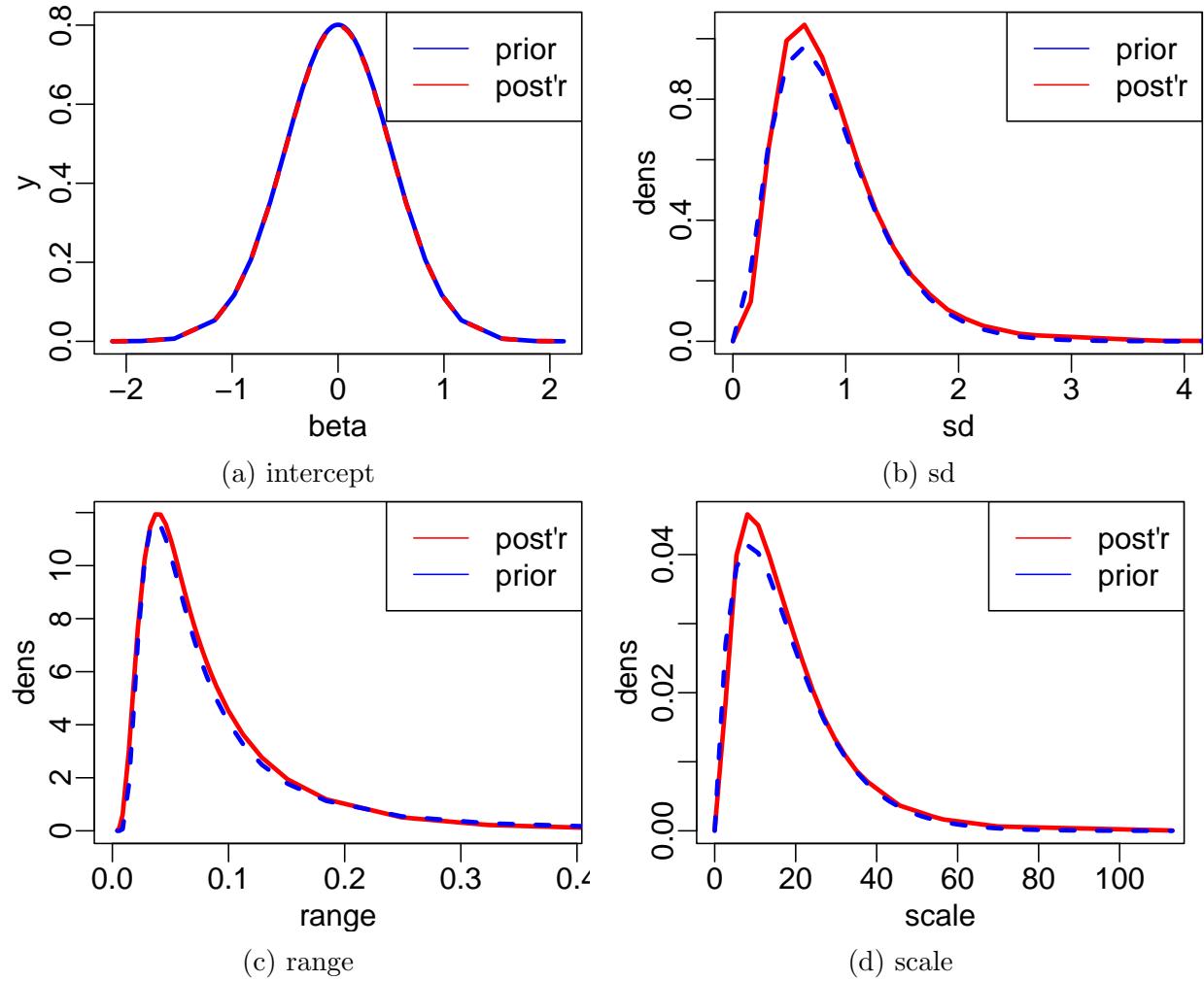


Figure 9: no data quantile priors

No data, legacy priors

```
resLegacy = res = geostatspp::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 = geostatsp::squareRaster(swissRain, Ncell, buffer=10*1000)

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

```

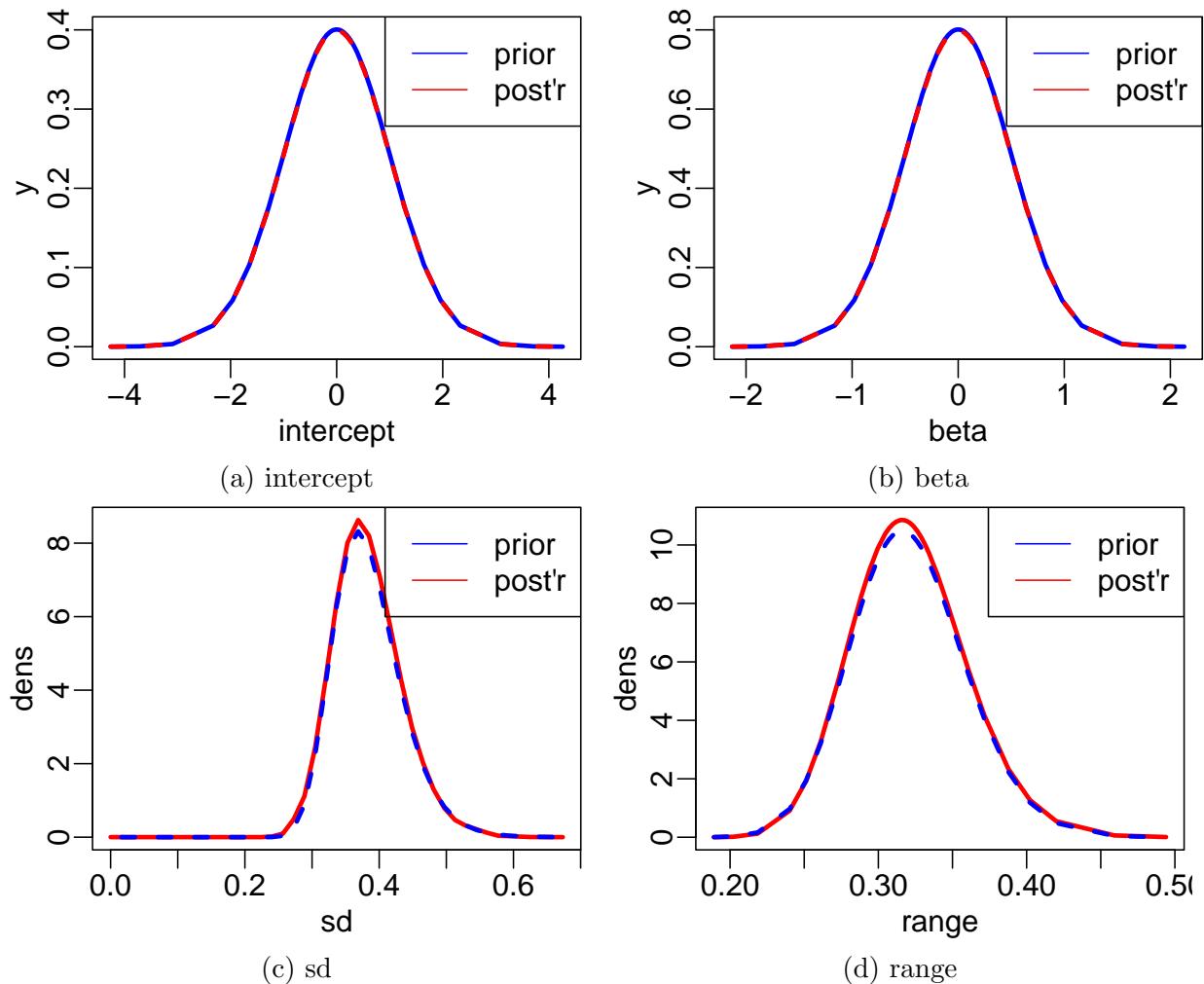


Figure 10: No data, legacy priors

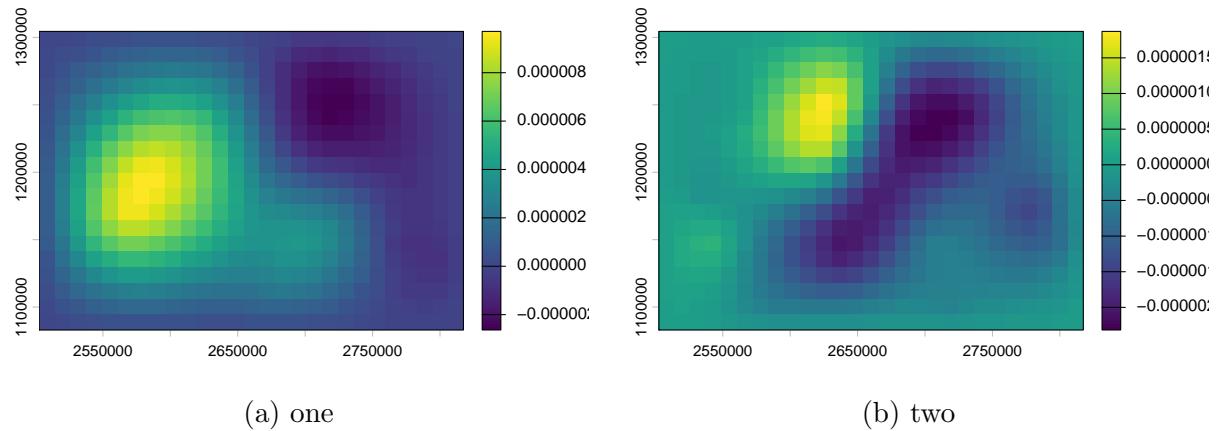


Figure 11: spatial formula provided

```

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']])
}

```