Various GLGM examples

Patrick Brown

September 21, 2023

```
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)
```

```
if(requireNamespace("INLA", quietly=TRUE) ) {
  INLA::inla.setOption(num.threads=2)
  # not all versions of INLA support blas.num.threads
  try(INLA::inla.setOption(blas.num.threads=2), silent=TRUE)
} else {
  print("INLA not installed")
}
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
```

```
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 = 25*fact)

## [1] 25
```

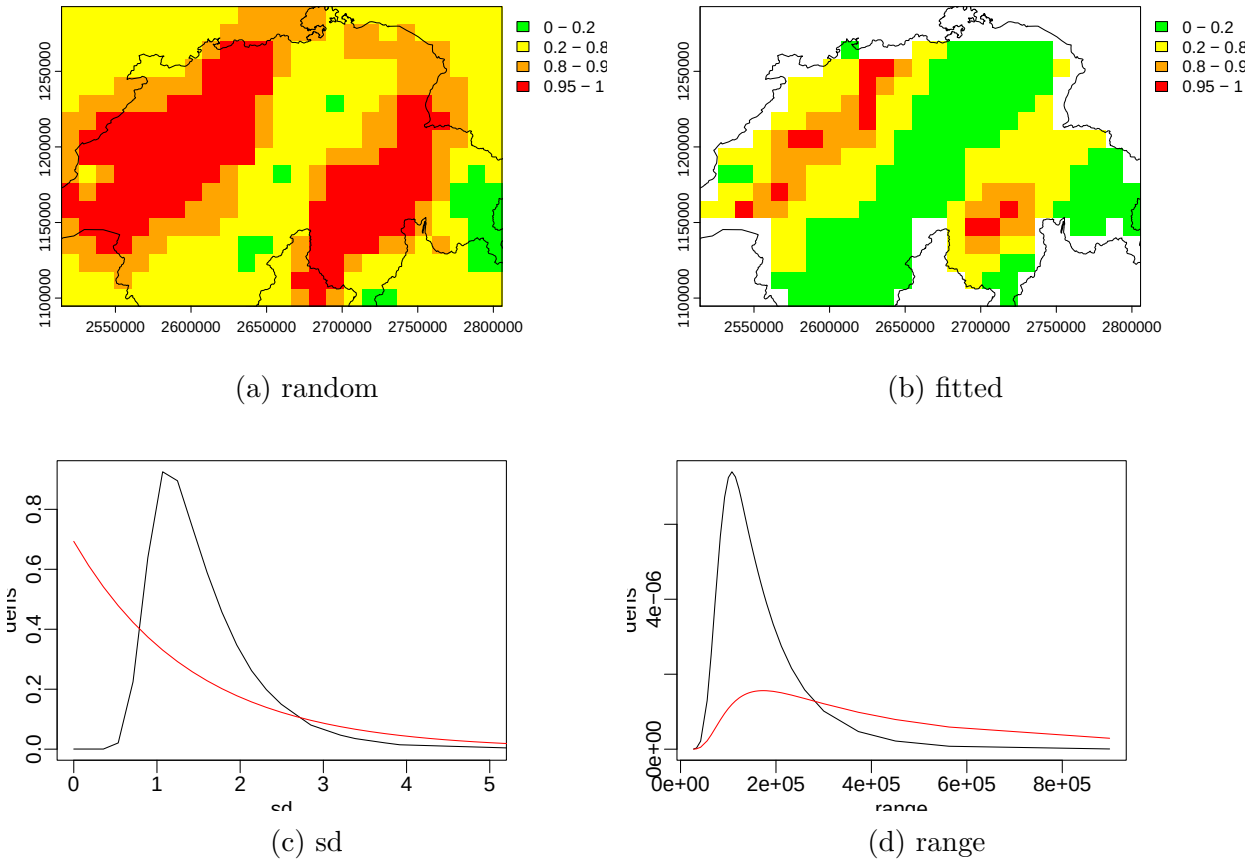
standard formula

```
if(requireNamespace('INLA', quietly=TRUE)) {

  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'),
    verbose=TRUE
  )
  if(length(swissFit$parameters)) {
    knitr::kable(swissFit$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')],
    lty=1, col=c('black','red'), type='l',
    xlab='range', ylab='dens')
  } else {
    print("INLA was not run, probably INLA isn't configured correctly")
  }
}

## saving INLA objects as /var/folders/1s/zkmc02qn4k18r6jdtbb459hc0000gn/T//RtmpzXbNWs
## inla done
```

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'

if(requireNamespace('INLA', quietly=TRUE)) {

  swissFitNp = glgm(
```



(a) random

(b) fitted

(c) sd

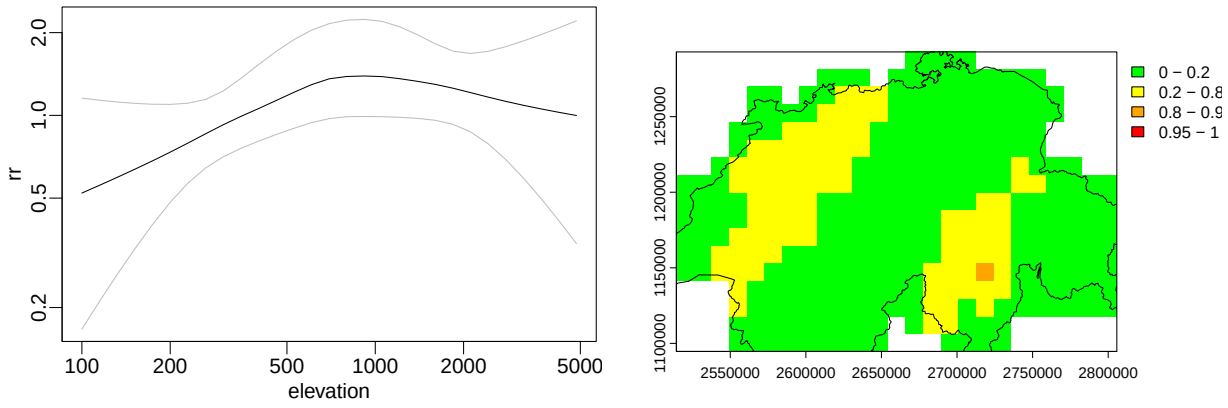
(d) range

Figure 1: Swiss rain as in help file

```
    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')
  )
  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)
  } else {
    print("INLA was not run, probably INLA isn't configured correctly")
  }

}
```

intercept only, named response variable. legacy priors

```
if(requireNamespace('INLA', quietly=TRUE)) {
  swissFit =  glgm("lograin", swissRain, Ncell,
    covariates=swissAltitude, family="gaussian", buffer=20000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000), sdObs = 2),
```



(a) elevation effect (b) fitted

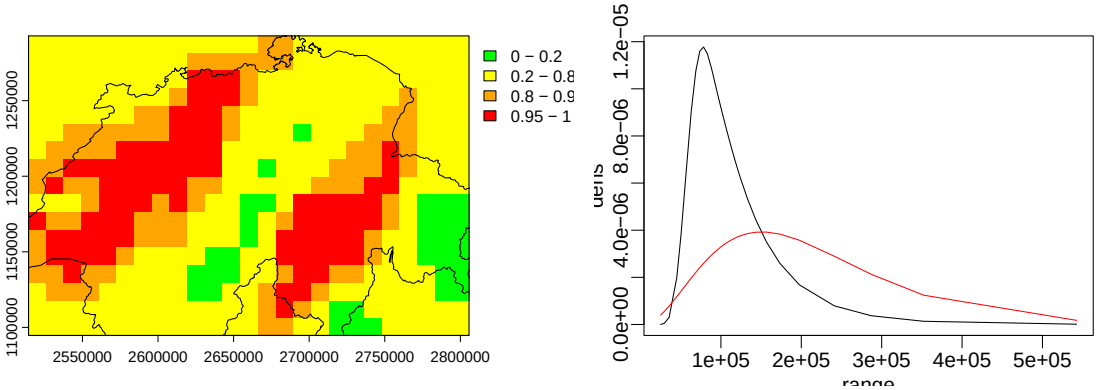
Figure 2: Swiss rain elevation rw2

```
    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	12.6074
CHE alt	-0.0001	-0.0004	-0.0001	0.0002	1.0127
range/1000	99.4402	43.7268	87.1929	231.8193	NA
sdNugget	0.3215	0.1989	0.3022	0.5028	NA
sd	0.9250	0.5925	0.8679	1.4267	NA

intercept only, add a covariate just to confuse glgm.

```
if(requireNamespace('INLA', quietly=TRUE)) {

  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)
```



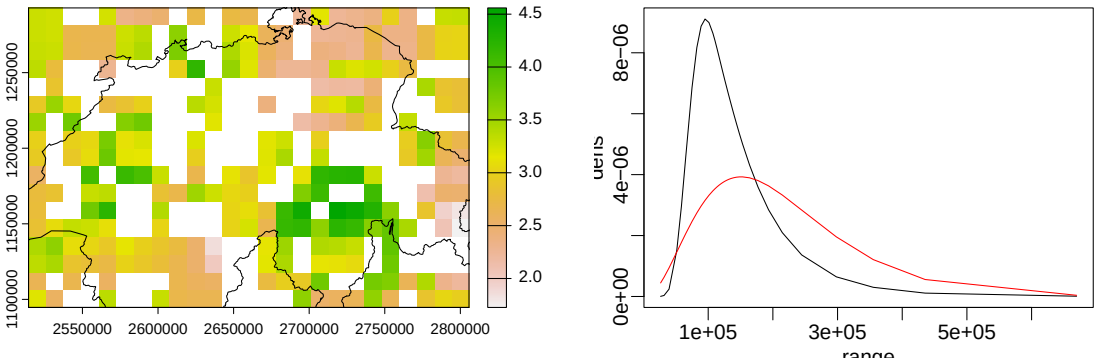
(a) exc prob (b) range

Figure 3: Swiss intercept only

```
  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')
  } else {
    print("INLA was not run, probably INLA isn't configured correctly")
  }

}
```

covariates are in data

```
newdat = swissRain
newdat$elev = extract(swissAltitude, swissRain, ID=FALSE)
swissLandType = unwrap(swissLandType)
if(requireNamespace('INLA', quietly=TRUE)) {
  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'),
```



(a) predict.mean (b) range

Figure 4: covaraites in data

```
    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')
  } else {
    print("INLA was not run, probably INLA isn't configured correctly")
  }

}
```

formula, named list elements

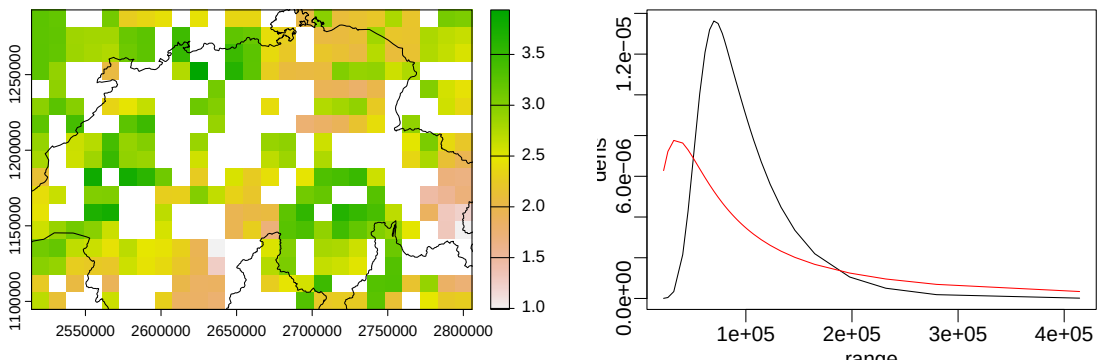
```
if(requireNamespace('INLA', quietly=TRUE)) {

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

```
##                       mean    0.025quant   0.975quant
## (Intercept)  2.456452e+00  1.6858526229 3.081684e+00
## elev        -9.736983e-05 -0.0003999107 2.049729e-04
## range/1000   1.165096e+02 48.7670921126 2.802847e+02
## sdNugget     3.482906e-01  0.2262342486 5.092476e-01
## sd           1.023815e+00  0.6179160696 1.669414e+00
```

categorical covariates

```
if(requireNamespace('INLA', quietly=TRUE)) {
  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)


  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')
    }
}
```



(a) map (b) range

Figure 5: categorical covariates

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

```
temp = values(swissAltitude)
temp[seq(10000,12000)] = NA
values(swissAltitude) = temp
if(requireNamespace('INLA', quietly=TRUE)) {

  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)	27.183	3.246	20.791	27.188	33.551
elev	-0.005	0.003	-0.011	-0.005	0.002
landMixed forests	-4.250	3.253	-10.630	-4.255	2.157
landGrasslands	-3.330	4.895	-12.938	-3.335	6.306
landCroplands	-9.537	4.209	-17.793	-9.542	-1.247
landUrban and built-up	-8.066	5.474	-18.807	-8.073	2.713
landEvergreen needleleaf forest	-11.998	6.265	-24.287	-12.007	0.340
landWater bodies	-15.823	8.040	-31.582	-15.837	0.021
landDeciduous needleleaf forest	-8.985	8.004	-24.686	-8.995	6.774
landDeciduous broadleaf forest	8.308	8.000	-7.419	8.309	24.025
landOpen shrublands	-11.591	11.024	-33.200	-11.608	10.120
landPermanent Wetlands	-21.619	10.869	-42.890	-21.649	-0.178
range/1000	198.862	281.445	19.386	116.237	898.883
sdNugget	11.661	-3.066	9.643	11.213	13.141
sd	0.009	0.000	0.003	0.008	0.022

covariates in data, factors

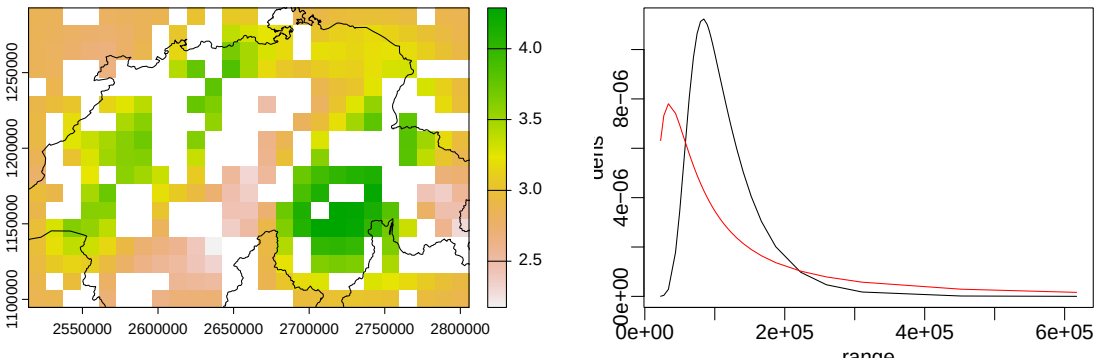
```
newdat = swissRain
newdat$landOrig = extract(swissLandType, swissRain, ID=FALSE)
newdat$landRel = relevel(newdat$landOrig, 'Mixed forests')



if(requireNamespace('INLA', quietly=TRUE)) {

  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)))),
    verbose= TRUE
  )

  if(length(swissFit$parameters)) {
    levels(newdat$landOrig)
    levels(newdat$landRel)
    levels(swissFit$inla$.args$data$landOrig)
    levels(swissFitR$inla$.args$data$landRel)
    knitr::kable(swissFit$parameters$summary, digits=3)
    knitr::kable(swissFitR$parameters$summary, digits=3)
  }
}

## saving INLA objects as /var/folders/1s/zkmc02qn4k18r6jdtbb459hc0000gn/T//RtmpzXbNWs
## inla done
```

	mean	sd	0.025quant	0.5quant	0.975quant	mode
(Intercept)	3.156	0.255	2.645	3.159	3.648	3.159
elev	-0.001	0.000	-0.001	-0.001	0.000	-0.001
landRelMixed forests	-0.186	0.142	-0.466	-0.186	0.092	-0.186
landRelGrasslands	-0.059	0.213	-0.479	-0.059	0.360	-0.059
landRelCroplands	-0.388	0.175	-0.735	-0.386	-0.047	-0.386
landRelUrban and built-up	-0.685	0.296	-1.265	-0.686	-0.101	-0.686
landRelEvergreen needleleaf forest	-0.598	0.296	-1.192	-0.594	-0.027	-0.594
landRelWater bodies	-0.997	0.381	-1.747	-0.998	-0.247	-0.998
landRelDeciduous needleleaf forest	-0.594	0.365	-1.315	-0.593	0.123	-0.593
landRelDeciduous broadleaf forest	0.330	0.354	-0.354	0.325	1.040	0.325
landRelOpen shrublands	-0.134	0.561	-1.235	-0.135	0.973	-0.135
landRelPermanent Wetlands	-2.636	0.515	-3.652	-2.635	-1.625	-2.635
range/1000	109.921	55.406	44.833	96.876	255.204	75.407
sdNugget	0.355	0.003	0.233	0.341	0.489	0.368
sd	0.803	0.045	0.479	0.751	1.331	0.847

covariates are in data, interactions

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

## Warning:  [‘[[<-‘] only using the first column

if(requireNamespace('INLA', quietly=TRUE)) {

  swissFit =  glgm(
```



(a) map (b) range

Figure 6: interactions

```
    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, 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')
  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)

if(requireNamespace('INLA', quietly=TRUE)  & fact > 1) {


  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)) {
```

```
  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')
}
}
```

```
if(requireNamespace('INLA', quietly=TRUE)  & 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 = vect(cbind(c(1,0), c(0,1)),
    atts=data.frame(y=c(0,0), offset=c(-50,-50), x=c(-1,1)))

if(requireNamespace('INLA', quietly=TRUE)  & 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')
  )

  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')
    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(requireNamespace('INLA', quietly=TRUE)  & fact > 1) {


  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"))
  }
}
```

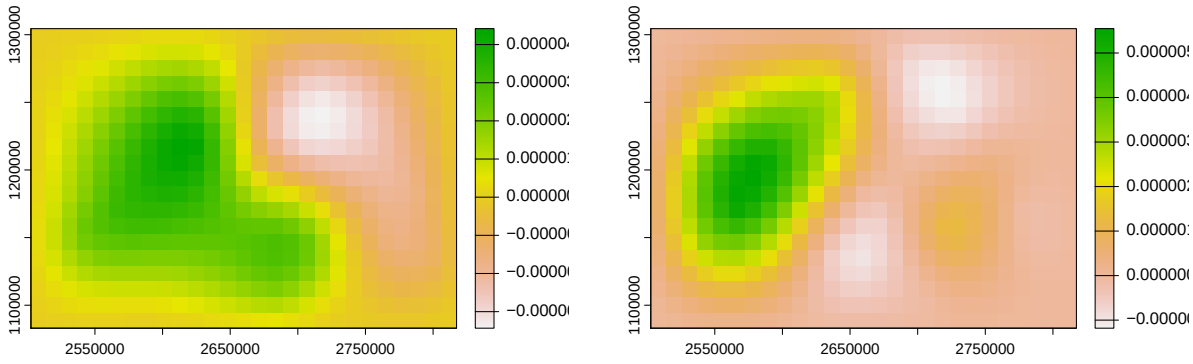
No data, legacy priors

```
if(requireNamespace('INLA', quietly=TRUE)  & 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)
  )

   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)

if(requireNamespace('INLA', quietly=TRUE) ) {
  swissFit = glgm(
    formula = rain ~ 1,
    data=swissRain,
    grid=theGrid,
    family="gaussian",
    spaceFormula = ~ f(space, model='matern2d',
      nrow = nrow(theGrid), ncol = ncol(theGrid),
      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']])
  }
}
```



(a) one (b) two

Figure 7: spatial formula provided