Various GLGM examples

Patrick Brown

March 10, 2021

```
library('mapmisc')

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

library("geostatsp")

## Loading required package:  Matrix

data('swissRain')



print(requireNamespace('INLA', quietly=TRUE))

## [1] FALSE

swissRain$lograin = log(swissRain$rain)

swissAltitudeCrop = raster::mask(swissAltitude,swissBorder)
```

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

```
fact

## [1] 1

(Ncell = 25*fact)

## [1] 25
```

standard formula

```r
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")
  }
}
```

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

  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')
  )
  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),
    control.inla=list(strategy='gaussian')
  )
  if(length(swissFit$parameters))
    knitr::kable(swissFit$parameters$summary[,c(1, 3:5, 8)], digits=4)
}
```

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)

  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$elev = extract(swissAltitude, swissRain)
if(requireNamespace('INLA', quietly=TRUE)) {
  swissFit =  glgm(lograin~ elev + land,
    newdat, Ncell,
    covariates=list(land=swissLandType),
    family="gaussian", buffer=40000,
    priorCI=list(sd=c(0.2, 2), range=c(50000,500000)),
    control.inla = list(strategy='gaussian'),
    control.family=list(hyper=list(prec=list(prior="loggamma",
          param=c(.1, .1))))
  )
  if(length(swissFit$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)]
}
```

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

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

}
```

covariates are in data, interactions

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

  swissFit =  glgm(
    formula = lograin~ elev : land,
    data=newdat,
    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))))
  )
  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')
  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(requireNamespace('INLA', quietly=TRUE)  & 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')
    )

    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 = SpatialPointsDataFrame(cbind(c(1,0), c(0,1)),
    data=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(!is.null(swissFit$inla$summary.random$space)) {
  swissFit$rasterTwo = setValues(
    raster::brick(swissFit$raster, nl=2),
    as.matrix(swissFit$inla$summary.random$space[
      ncell(theGrid)+values(swissFit$raster[['space']]),
      c('mean','0.5quant')]))
  plot(swissFit$raster[['random.mean']])

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