swissrain data code

ruoyong

2023-12-14

R Markdown

Get MLE's of parameters

```
library('geostatsp')
data("swissRain")
swissRain = terra::unwrap(swissRain)
swissAltitude = terra::unwrap(swissAltitude)
swissRain$elevation = terra::extract(swissAltitude, swissRain, ID=FALSE)

swissRes =  lgm( formula=rain~ elevation,
                 data=swissRain, grid=20,
                 covariates=swissAltitude,
                 reml = FALSE,
                 fixBoxcox=FALSE, fixShape=FALSE, fixNugget = FALSE,
                 aniso=TRUE )
swissRes2 =  lgm( formula=rain~ elevation,
                  data=swissRain, grid=20,
                  covariates=swissAltitude,
                  reml = FALSE,
                  shape=0.5,
                  fixBoxcox=FALSE, fixShape=TRUE, fixNugget = FALSE,
                  aniso=TRUE )
swissRes3 =  lgm( formula=rain~ elevation,
                  data=swissRain,
                  grid=20,
                  covariates=swissAltitude,
                  reml = FALSE,
                  shape=0.9,
                  fixBoxcox=FALSE, fixShape=TRUE, fixNugget = FALSE,
                  aniso=TRUE )
swissRes4 =  lgm( formula=rain~ elevation,
                  data=swissRain, grid=20,
                  covariates=swissAltitude,
                  reml = FALSE,
                  shape=10,
                  fixBoxcox=FALSE, fixShape=TRUE, fixNugget = FALSE,
                  aniso=TRUE )
swissRes5 =  lgm( formula=rain~ elevation,
                  data=swissRain, grid=20,
                  covariates=swissAltitude,
                  reml = FALSE,
```

```
                  shape=20,
                  fixBoxcox=FALSE, fixShape=TRUE, fixNugget = FALSE,
                  aniso=TRUE )
swissRes6 =  lgm( formula=rain~ elevation,
                  data=swissRain, grid=20,
                  covariates=swissAltitude,
                  reml = FALSE,
                  shape=100,
                  fixBoxcox=FALSE, fixShape=TRUE, fixNugget = FALSE,
                  aniso=TRUE )
swissRes$summary[,c('estimate','ci0.05', 'ci0.95')]
```

```
##                          estimate        ci0.05       ci0.95
## (Intercept)         5.007959e+00  3.3239652105  6.691951829
## elevation           2.303839e-04 -0.0003602182  0.000820986
## sdNugget            9.919893e-01  0.6151378026  1.599711232
## sdSpatial           2.680360e+00  1.5686609623  4.579912084
## range/1000          3.861873e+01 18.5441420554 80.424676428
## shape               1.826358e+00 -1.2546086206  4.907324764
## anisoRatio          8.091897e+00  4.7253194736 13.857008966
## anisoAngleRadians   6.518085e-01  0.5785331808  0.725083813
## anisoAngleDegrees   3.734588e+01 33.1475095693 41.544242295
## boxcox              4.962398e-01  0.3375438647  0.654935794
```

```
swissDataWrap = terra::wrap(swissRes$data)
```

Set the representative parameters

```
library('gpuR')
setContext(grep("gpu", listContexts()$device_type)[1])
library(gpuLik)
alpha1=c(0.00001, 0.01, 0.1, 0.2, 0.25, 0.3, 0.5, 0.8, 0.9, 0.95, 0.99, 0.999)
alpha2=c(0.00001, 0.01, 0.1, 0.2, 0.25, 0.5, 0.8, 0.9, 0.95, 0.99, 0.999)
model_list <- list(swissRes, swissRes2,swissRes3,swissRes4,swissRes5,swissRes6)
A <- configParams(Model = model_list, alpha=alpha1, alphasecond = alpha2,
                        data = terra::unwrap(swissDataWrap))
paramsUse <- A$representativeParamaters
b <- A$boxcox
```

```
intercept<- sort(c(swissRes$summary['(Intercept)','estimate'], seq(0, 12, len=199)))
cov1  <- sort(c(swissRes$summary['elevation','estimate'], seq(-7, 15, len=199)*1e-04))
Betas <- cbind(intercept, cov1)
sdSpatial <- c(seq(1.2, 5, len=199), swissRes$summary['sdSpatial','estimate'])
```

Get the profie loglikelihoods and estimates for all model parameters on GPU in one step,

change verbose argument if you would like to see backend code information

```
result<-gpuLik::likfitLgmGpu(model =swissRes,
                        params=paramsUse,
                        data = terra::unwrap(swissDataWrap),
                        paramToEstimate=c('range','combinedRange','shape','nugget',
```

```
                                        'aniso1', 'aniso2',"anisoRatio",
                                        "anisoAngleRadians",'boxcox'),
                        boxcox = seq(b[1],b[9],len=33),
                        Betas = Betas,
                        sdSpatial = sdSpatial,
                        cilevel=0.9,
                        type = "double",
                        convexHullForBetas = FALSE,
                        NparamPerIter=256,
                        Nglobal=c(64,64),
                        Nlocal=c(8,8),
                        NlocalCache=2000,
                        verbose=c(0,0))
```

Show the results

```
# total parameter configurations for \omega
nrow(paramsUse)
```

```
## [1] 15318
```

```
result$summary
```

```
##                       estimate     lower90ci    upper90ci
## (Intercept)       5.007959e+00     2.7669574 7.797762e+00
## elevation         2.303839e-04    -0.0004360 9.060000e-04
## sdSpatial         2.680360e+00     1.6293434 4.739678e+00
## range             3.861873e+04 21811.4034449 1.181803e+05
## combinedRange     1.357603e+04  6366.3971588 4.863710e+04
## shape             1.826358e+00     0.3881117 1.042552e+02
## nugget            1.369707e-01     0.0000000 2.771401e-01
## aniso1            7.030799e-01     0.3042457 1.283947e+00
## aniso2            2.568575e+00     1.5311561 3.554275e+00
## anisoRatio        8.091897e+00     3.7282632 1.490593e+01
## anisoAngleRadians 6.518085e-01     0.5363850 7.313044e-01
## boxcox            4.962398e-01     0.3415693 6.579213e-01
```

```
result$reml
```

```
## [1] FALSE
```

```
result$Infindex
```

```
##  [1]  46  58 100 118 127 129 138 288 300 309 324 369 377 395 611 613 619 622 664
## [20] 667 691 694 700 703 730
```

```
swissRes$optim$logL
```

```
## m2logL.ml   logL.ml
##   639.908  -319.954
```

```
max(result$LogLik)
```

```
## [1] -319.954
```

Alternatively, we can get the estimates step by step

Step 1, get the profie loglikelihoods for correlation and BoxCox parameters

```
result_1 <- gpuLik::getProfLogL(data = terra::unwrap(swissDataWrap),
                                formula=rain~ elevation,
                                coordinates=terra::crds(swissRain),
                                params=paramsUse,
                                boxcox = seq(b[1],b[9],len=33),
                                type = "double",
                                NparamPerIter=256,
                                gpuElementsOnly = FALSE,
                                reml=FALSE,
                                Nglobal=c(64,64),
                                Nlocal=c(16,8),
                                NlocalCache=2000,
                                verbose=c(0,0))
```

Show some results

```
# indices of parameter sets that leads to NaN or infinity in Loglikelihood
result_1$Infindex
```

```
##  [1]  46  58 100 118 127 129 138 288 300 309 324 369 377 395 611 613 619 622 664
## [20] 667 691 694 700 703 730
```

```
# total configurations for \lambda
result_1$Ndata
```

```
## [1] 33
```

```
# number of observations
result_1$Nobs
```

```
## [1] 100
```

```
# number of covariates in the model
result_1$Ncov
```

```
## [1] 2
```

```
# predictors in the model
result_1$predictors
```

```
## [1] "(Intercept)" "elevation"
```

Get colorful profile likelihoods plots (setup)

```
LogLikcpu = result_1$LogLik  # cpu matrix
XVYXVX = result_1$XVYXVX  # cpu matrix
ssqResidual = result_1$ssqResidual  # cpu matrix
paramToEstimate = c('range','combinedRange',"anisoRatio",
                    'shape','nugget', 'sdNugget',
                    "anisoAngleRadians", 'aniso1',
                    'aniso2','boxcox')
cilevel=0.9  # decimal
paramsRenew = result_1$paramsRenew
```

```
params = result_1$paramsRenew
boxcox = result_1$boxcox
Ndata = result_1$Ndata
Nobs = result_1$Nobs
Ncov = result_1$Ncov
reml = FALSE
predictors = result_1$predictors
chisqValue <- qchisq(cilevel,  df = 1)/2
Table <- matrix(NA, nrow=length(paramToEstimate) + Ncov + 1, ncol=3)
rownames(Table) <-  c(predictors, "sdSpatial", paramToEstimate)
colnames(Table) <-  c("estimate", "lci", "uci")
index <- which(LogLikcpu == max(LogLikcpu, na.rm = TRUE), arr.ind = TRUE)
##################sigma hat#########################
if(reml==FALSE)  {
  Table["sdSpatial",1] <- sqrt(ssqResidual[index[1],index[2]]/Nobs)
}else{
  Table["sdSpatial",1] <- sqrt(ssqResidual[index[1],index[2]]/(Nobs - Ncov))
}
maximum <- max(LogLikcpu)
breaks = maximum - qchisq(cilevel,  df = 1)/2
############### profile for covariance parameters ####################

aniso1 <-  unname(sqrt(paramsRenew[,'anisoRatio']-1) * cos(2*(paramsRenew[,'anisoAngleRadians'])))
aniso2 <-  unname(sqrt(paramsRenew[,'anisoRatio']-1) * sin(2*(paramsRenew[,'anisoAngleRadians'])))
aniso <- cbind(aniso1, aniso2)
sumLogRange <- log(paramsRenew[,'range']^2/paramsRenew[,'anisoRatio'])
combinedRange <- sqrt(paramsRenew[,'range']^2/paramsRenew[,'anisoRatio'])
paramsRenew <- cbind(paramsRenew, sumLogRange, combinedRange, aniso, sqrt(paramsRenew[,"nugget"]) * Tabl
colnames(paramsRenew)[ncol(paramsRenew)] <- 'sdNugget'
#################################################
Spars = c("range",'sumLogRange',"combinedRange","nugget",'sdNugget',"shape",
          'aniso1','aniso2','anisoRatio','anisoAngleRadians',"alpha")
result = data.table::as.data.table(cbind(LogLikcpu, paramsRenew[,Spars]))
profileLogLik <- result[, .(profile=max(.SD)), by=Spars]

colAlpha = mapmisc::colourScale(profileLogLik$alpha, style='unique', breaks = 12, col=rainbow, opacity =
colAlpha$plot[which(is.na(colAlpha$plot))] = '#000000FF'

profileLogLik[,'profile'] <- profileLogLik[,'profile'] - breaks
profileLogLik <- profileLogLik[profile > maximum- breaks-10]  #maximum- breaks
profileLogLik <- as.data.frame(profileLogLik)
```

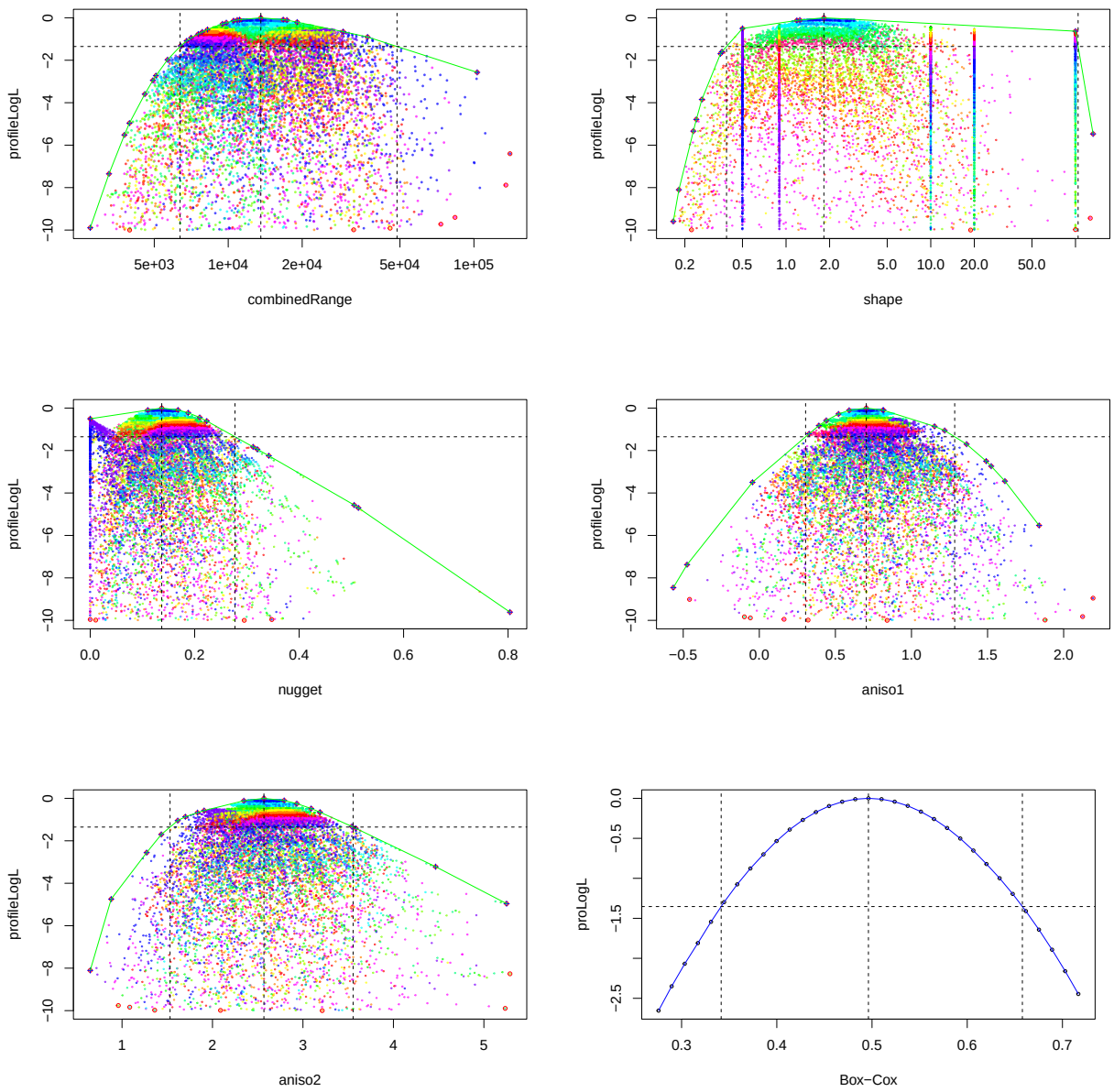
Get the colorful profile plots for covariance and BoxCox parameters

```
plot(profileLogLik$combinedRange, profileLogLik$profile-chisqValue, log='x',cex=.4, xlab="combinedRange
profileLogLik$sumLogRange <- 2*log(profileLogLik$combinedRange)
newdata <- profileLogLik[,c('sumLogRange','profile')]
colnames(newdata)[1]<-"x1"
datC2 = geometry::convhulln(newdata)
allPoints = unique(as.vector(datC2))
toTest = newdata[allPoints,]
toTest[,'profile'] = toTest[,'profile'] + 0.1
inHull = geometry::inhulln(datC2, as.matrix(toTest))
```

```
toUse = newdata[allPoints,][!inHull,]
toTest = newdata[allPoints,]
toUse <- toUse[order(toUse$x1),]
toUse <- head(toUse, - 1)

interp1 = mgcv::gam(profile ~ s(x1, k=nrow(toUse),  m=1, fx=TRUE), data=toUse)
profsumLogRange = data.frame(x1=seq(min(toUse$x1), max(toUse$x1), len=1001))
profsumLogRange$z = predict(interp1, profsumLogRange)

points(exp(0.5*toTest[,1]), toTest[,2]-chisqValue, col='red', cex=0.6)
points(exp(0.5*toUse[,1]), toUse[,2]-chisqValue, col='blue', cex=0.6, pch=3)
lines(exp(0.5*profsumLogRange$x1), profsumLogRange$z-chisqValue, col = 'green')
abline(h =-chisqValue, lty = 2, col='black')
lower = min(newdata$x1)
upper = max(newdata$x1)
f1 <- approxfun(toUse[,1], toUse[,2])
MLE <- sqrt(paramsRenew[index[1],'range']^2/paramsRenew[index[1],'anisoRatio'])
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(MLE,exp(0.5*ci)), lty = 2, col='black')



## shape
plot(profileLogLik$shape, profileLogLik$profile-chisqValue, cex=.2, xlab="shape", ylab="profileLogL",
profileLogLik$logshape <- log(profileLogLik$shape)
newdata <- profileLogLik[,c('logshape','profile')]
colnames(newdata)[1]<-"x1"

datC2 = geometry::convhulln(newdata)
allPoints = unique(as.vector(datC2))
toTest = newdata[allPoints,]
toTest[,'profile'] = toTest[,'profile'] + 0.1
inHull = geometry::inhulln(datC2, as.matrix(toTest))
toUse = newdata[allPoints,][!inHull,]
toTest = newdata[allPoints,]
toUse <- toUse[order(toUse$x1),]
points(exp(toTest[,1]),toTest[,2]-chisqValue, col='red', cex=0.6)
points(exp(toUse[,1]), toUse[,2]-chisqValue, col='blue', cex=0.6, pch=3)
interp1 = mgcv::gam(profile ~ s(x1, k=nrow(toUse),  m=1, fx=TRUE), data=toUse)
profShapeLog = data.frame(x1=seq(min(toUse$x1), max(toUse$x1), len=1001))
profShapeLog$z = predict(interp1, profShapeLog)

lines(exp(profShapeLog$x1), profShapeLog$z-chisqValue, col = 'green')
abline(h =-chisqValue, lty = 2, col='black')
f1 <- approxfun(toUse[,1], toUse[,2])

lower = min(toUse$x1)
upper = max(toUse$x1)
MLE <- paramsRenew[index[1],'shape']
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(MLE,exp(ci)), lty = 2, col='black')
```

```
## nugget
plot(profileLogLik$nugget, profileLogLik$profile-chisqValue, cex=.2, xlab="nugget", ylab="profileLogL",
profileLogLik2 <- profileLogLik[,c('nugget','profile')]
colnames(profileLogLik2) <- c("x1", 'profile')
datC2 = geometry::convhulln(profileLogLik2)
allPoints = unique(as.vector(datC2))
toTest = profileLogLik2[allPoints,]
toTest[,'profile'] = toTest[,'profile'] + 0.1
inHull = geometry::inhulln(datC2, as.matrix(toTest))
toUse = profileLogLik2[allPoints,][!inHull,]
toTest = profileLogLik2[allPoints,]
toUse <- toUse[order(toUse$x1),]
points(toTest[,1], toTest[,2]-chisqValue, col='red', cex=0.6)
points(toUse[,1], toUse[,2]-chisqValue, col='blue', cex=0.6, pch=3)

interp1 = mgcv::gam(profile ~ s(x1, k=nrow(toUse), m=1, fx=TRUE), data=toUse)
profNugget = data.frame(x1=seq(min(toUse$x1), max(toUse$x1), len=1001))
profNugget$z = predict(interp1, profNugget)
lines(profNugget$x1, profNugget$z-chisqValue, col = 'green')
abline(h =-chisqValue, lty = 2, col='black')

lower = min(profileLogLik2$x1)
upper = max(profileLogLik2$x1)
f1 <- approxfun(toUse[,1], toUse[,2])
MLE <- paramsRenew[index[1],'nugget']
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(MLE,ci), lty = 2,  col='black')


## aniso1
plot(profileLogLik$aniso1, profileLogLik$profile-chisqValue, cex=.2, xlab="aniso1", ylab="profileLogL",
profileLogLik2 <- profileLogLik[,c('aniso1','profile')]
colnames(profileLogLik2) <- c("x1", 'profile')
datC2 = geometry::convhulln(profileLogLik2)
allPoints = unique(as.vector(datC2))
toTest = profileLogLik2[allPoints,]
toTest[,'profile'] = toTest[,'profile'] + 0.1
inHull = geometry::inhulln(datC2, as.matrix(toTest))
toUse = profileLogLik2[allPoints,][!inHull,]
toTest = profileLogLik2[allPoints,]
toUse <- toUse[order(toUse$x1),]
toUse <- head(toUse, - 1)
points(toTest[,1], toTest[,2]-chisqValue, col='red', cex=0.6)
points(toUse[,1], toUse[,2]-chisqValue, col='blue', cex=0.6, pch=3)

interp1 = mgcv::gam(profile ~ s(x1, k=nrow(toUse), m=1, fx=TRUE), data=toUse)
profaniso1 = data.frame(x1=seq(min(toUse$x1), max(toUse$x1), len=1001))
profaniso1$z = predict(interp1, profaniso1)
lines(profaniso1$x1, profaniso1$z-chisqValue, col = 'green')

abline(h =-chisqValue, lty = 2, col='black')
lower = min(profaniso1$x1)
upper = max(profaniso1$x1)
```

```
f1 <- approxfun(toUse[,1], toUse[,2])
MLE <- sqrt(paramsRenew[index[1],'anisoRatio']-1) * cos(2*(paramsRenew[index[1],'anisoAngleRadians']))
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(MLE,ci), lty = 2, col='black')


## aniso2
plot(profileLogLik$aniso2, profileLogLik$profile-chisqValue, cex=.2, xlab="aniso2", ylab="profileLogL",
profileLogLik3 <- profileLogLik[,c('aniso2','profile')]
colnames(profileLogLik3) <- c("x1", 'profile')
datC2 = geometry::convhulln(profileLogLik3)
allPoints = unique(as.vector(datC2))
toTest = profileLogLik3[allPoints,]
toTest[,'profile'] = toTest[,'profile'] + 0.1
inHull = geometry::inhulln(datC2, as.matrix(toTest))
toUse = profileLogLik3[allPoints,][!inHull,]
toTest = profileLogLik3[allPoints,]
toUse <- toUse[order(toUse$x1),]
toUse <- head(toUse, - 1)
points(toTest[,1], toTest[,2]-chisqValue, col='red', cex=0.6)
points(toUse[,1], toUse[,2]-chisqValue, col='blue', cex=0.6, pch=3)

interp1 = mgcv::gam(profile ~ s(x1, k=nrow(toUse), m=1, fx=TRUE), data=toUse)
profaniso2 = data.frame(x1=seq(min(toUse$x1), max(toUse$x1), len=1001))
profaniso2$z = predict(interp1, profaniso2)
lines(profaniso2$x1, profaniso2$z-chisqValue, col = 'green')
abline(h =-chisqValue, lty = 2, col='black')
lower = min(profileLogLik3$x1)
upper = max(profileLogLik3$x1)

f1 <- approxfun(toUse[,1], toUse[,2])
MLE <- sqrt(paramsRenew[index[1],'anisoRatio']-1) * sin(2*(paramsRenew[index[1],'anisoAngleRadians']))
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(MLE,ci), lty = 2, col='black')


## boxcox
likForboxcox = cbind(boxcox, apply(LogLikcpu, 2,  max) )
f1 <- approxfun(likForboxcox[,1], likForboxcox[,2]-breaks)
plot(likForboxcox[,1], likForboxcox[,2]-breaks-chisqValue, ylab= "proLogL", xlab='Box-Cox', cex=0.5)
likForboxcox <- likForboxcox[order(likForboxcox[,1]),]
lines(likForboxcox[,1], likForboxcox[,2]-breaks-chisqValue, col='blue')
abline(h =-chisqValue, lty = 2)

lower = min(boxcox)
upper = max(boxcox)
ci<-rootSolve::uniroot.all(f1, lower = lower, upper = upper)
abline(v =c(boxcox[index[2]],ci), lty = 2)
```



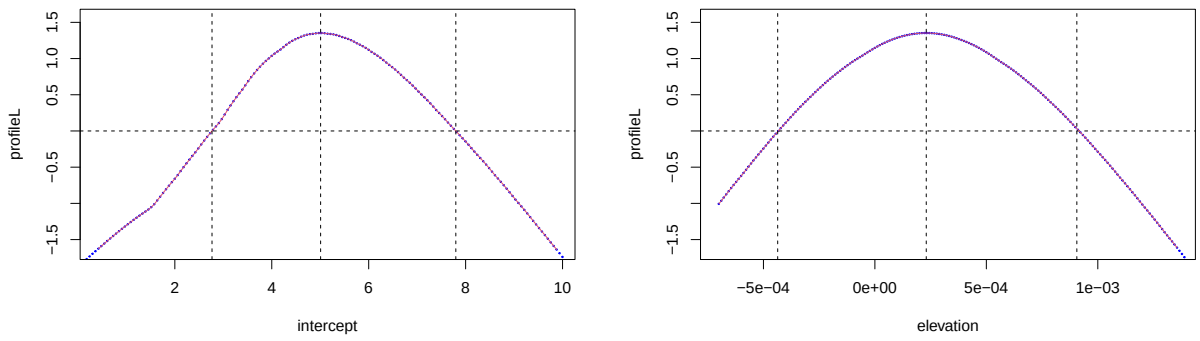
Step 2, get estimates for correlation parameters only

```
result_2<-gpuLik::prof1dCov(LogLik = result_1$LogLik,  # cpu matrix
                  XVYXVX = result_1$XVYXVX,  # cpu matrix
                  ssqResidual = result_1$ssqResidual,  # cpu matrix
                  paramToEstimate = c('range','combinedRange','shape',
                                      'nugget','aniso1', 'aniso2','boxcox'),
                  cilevel=0.9,
                  params = result_1$paramsRenew,
                  boxcox = result_1$boxcox,
                  Ndata = result_1$Ndata,
                  Nobs = result_1$Nobs,
```

```
                    Ncov = result_1$Ncov,
                    reml = FALSE,
                    predictors = result_1$predictors,  # character string
                    verbose=FALSE)

# table of estimates
result_2$summary
# index for the MLE parameter set
result_2$mleIndex
```

Step 3, get estimates and profile plots for betas'

```
intercept<- sort(c(swissRes$summary['(Intercept)','estimate'], seq(0, 12, len=199)))
elevation  <- sort(c(swissRes$summary['elevation','estimate'], seq(-7, 15, len=199)*1e-04))
Betas <- cbind(intercept, elevation)

output<-gpuLik::Prof1dBetas(Betas=Betas,
                            cilevel=0.9,
                            Nobs = result_1$Nobs,
                            Ndata = result_1$Ndata,
                            Nparam = result_1$Nparam,
                            Ncov = result_1$Ncov,
                            detVar = result_1$detVar,
                            result_1$detReml,
                            result_1$ssqY,
                            result_1$XVYXVX,
                            result_1$jacobian,
                            convexHull = FALSE)
```




```
output$estimates
```

```
##                     MLE lower90ci upper90ci  maximum
## intercept 5.0079585199  2.766957  7.797762 -319.954
## elevation 0.0002303839 -0.000436  0.000906 -319.954
```

```
output$breaks
```

```
## [1] -321.3068
```

Step 4, get stimates and profile plots for variance parameter

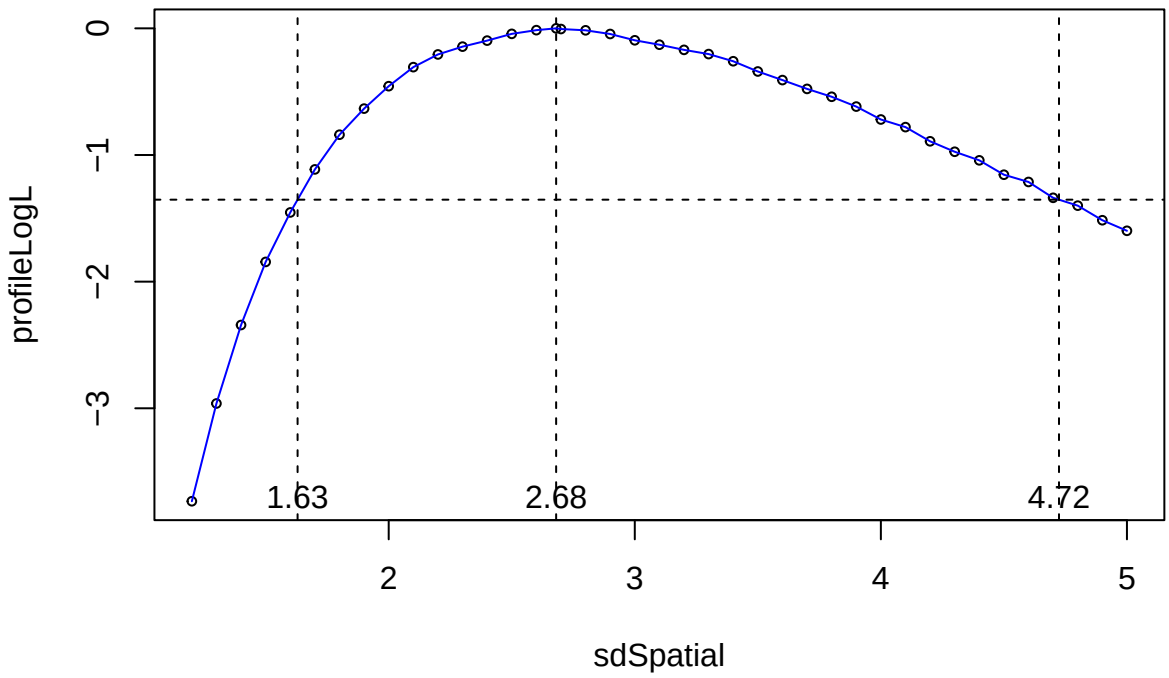
```
sdSpatial <- sort(c(seq(1.2, 5, len=39), swissRes$summary['sdSpatial','estimate']))
provarresult <- gpuLik::profVariance(sdSpatial,
             cilevel=0.9,
             result_1$Nobs,
             result_1$Ndata,
             result_1$Nparam,
             result_1$Ncov,
             result_1$detVar,
             result_1$detReml,
             result_1$ssqResidual,
             result_1$jacobian)
```

```
provarresult$estimates
```

```
##          MLE lower90ci upper90ci  maximum
## [1,] 2.68036  1.629685  4.723883 -319.954
```

```
breaks <- provarresult$breaks

temp <- qchisq(cilevel,  df = 1)/2
LogLik <- provarresult$LogLik
```

```
plot(sdSpatial,LogLik-breaks-temp, cex=0.6, ylab='profileLogL')
lines(sdSpatial,LogLik-breaks-temp, col='blue')
abline(h=-temp, lty = 2)

abline(v=provarresult$estimates[c(1,2,3)], lty = 2)
text(provarresult$estimates[1], -3.7, round(provarresult$estimates[1],digits = 2))
text(provarresult$estimates[2], -3.7, round(provarresult$estimates[2],digits = 2))
text(provarresult$estimates[3], -3.7, round(provarresult$estimates[3],digits = 2))
```



Show some 2-dimensional profile plots (setup)

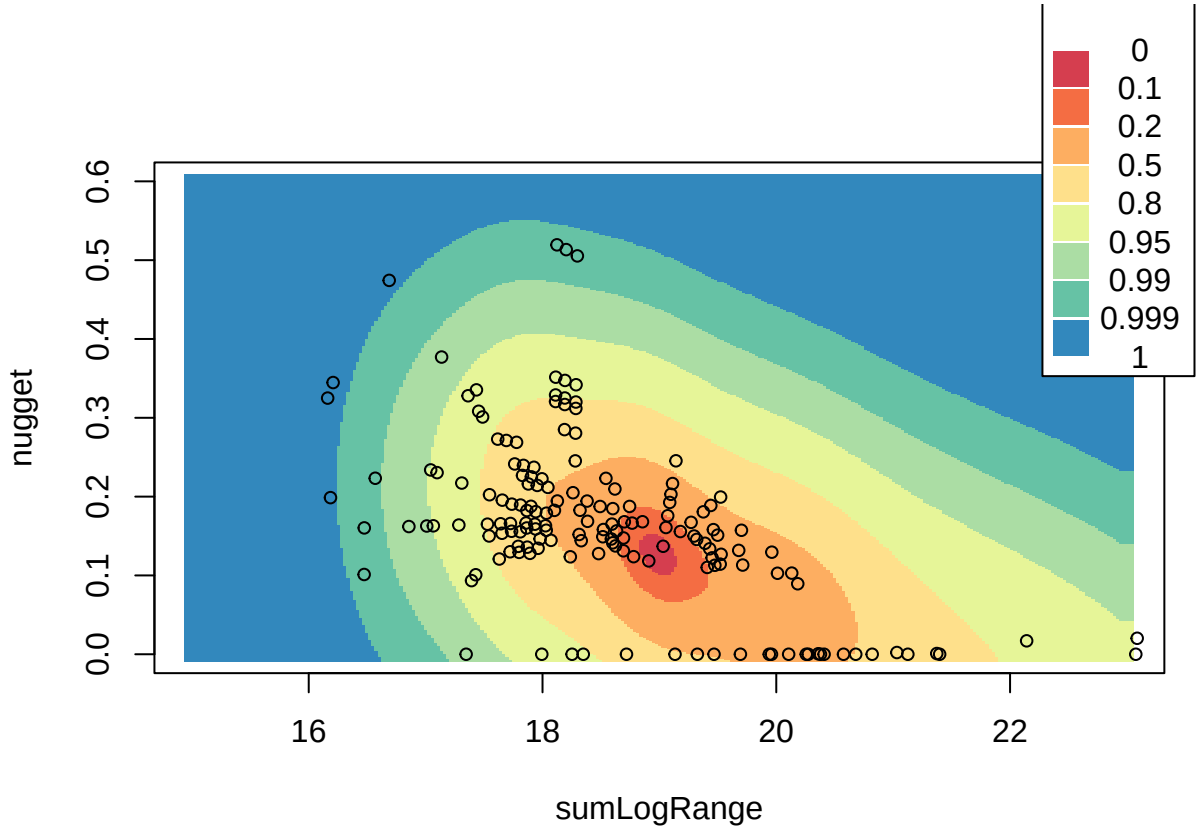
```
Spars = c("range",'sumLogRange',"combinedRange","nugget",'sdNugget',"shape",'aniso1','aniso2','anisoRati
result = data.table::as.data.table(cbind(LogLikcpu, paramsRenew[,Spars]))
profileLogLik <- result[, .(profile=max(.SD)), by=Spars]
head(profileLogLik)
```

```
##       range sumLogRange combinedRange    nugget  sdNugget      shape    aniso1
## 1: 38618.73    19.03212     13576.032 0.1369707 0.9919893   1.826358 0.7030799
## 2: 56619.83    20.10588     23224.002 0.0000000 0.0000000   0.500000 0.8801964
## 3: 53229.66    19.76531     19587.637 0.1019224 0.8557125   0.900000 0.7196071
## 4: 27746.92    18.27493      9297.163 0.1629248 1.0818990  10.000000 0.7119055
## 5: 26715.42    18.18058      8868.744 0.1655695 1.0906448  20.000000 0.7170093
## 6: 25918.05    18.10231      8528.368 0.1676042 1.0973257 100.000000 0.7226641
##      aniso2 anisoRatio anisoAngleRadians alpha   profile
## 1: 2.568575   8.091897         0.6518085    NA -319.9540
## 2: 2.041822   5.943784         0.5818920    NA -320.4571
## 3: 2.422198   7.384877         0.6410064    NA -320.2834
## 4: 2.720318   8.906941         0.6574186    NA -320.3786
## 5: 2.749530   9.074020         0.6578512    NA -320.4893
## 6: 2.777323   9.235765         0.6581197    NA -320.5854
```

```
profileLogLik[,'profile'] <- profileLogLik[,'profile'] -maximum
profileLogLik <- profileLogLik[profile > -10]
profileLogLik$logshape <- log(profileLogLik$shape)
profileLogLik <- as.data.frame(profileLogLik)
```

2-dimensional profile plot of sumLogRange vs nugget

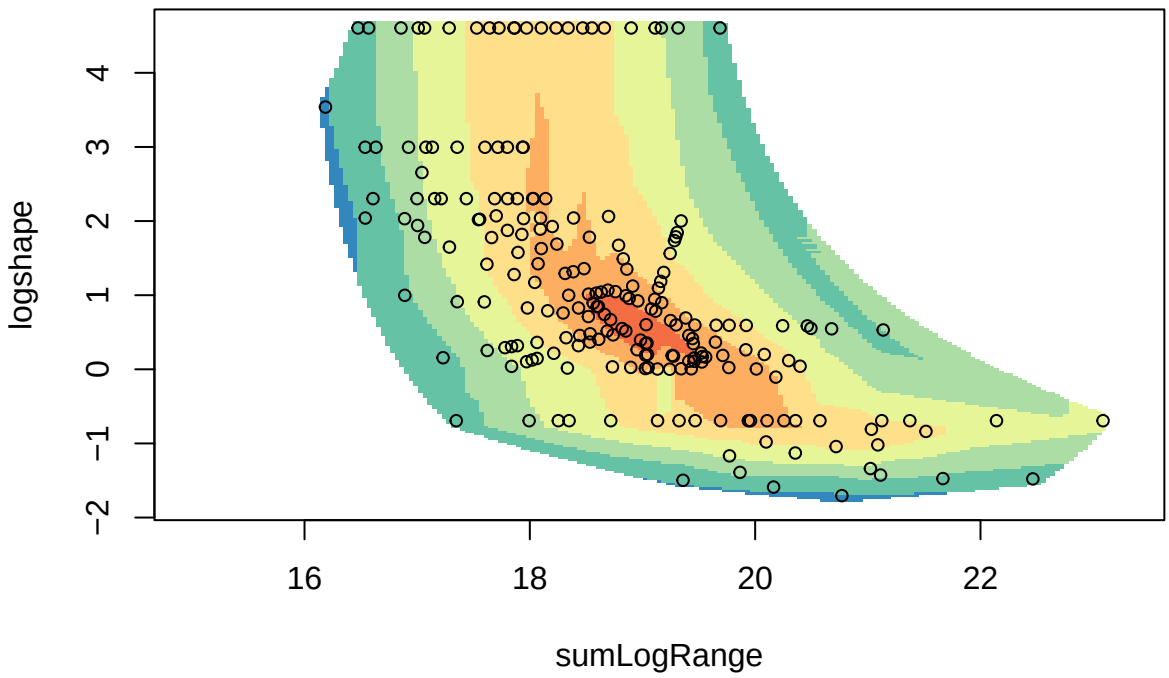
```
profileLogL <- profileLogLik[,c('sumLogRange','nugget','profile')]
datC2 = geometry::convhulln(profileLogL)
allPoints2 = unique(as.vector(datC2))
toTest2 = profileLogL[allPoints2,]
toTest2[,'profile'] = toTest2[,'profile'] - 0.1
inHull2 = geometry::inhulln(datC2, as.matrix(toTest2))
toUse2 = profileLogL[allPoints2,][inHull2,]
toTest2 = profileLogL[allPoints2,]

Sprob = c(0, 0.1, 0.2, 0.5, 0.8, 0.95, 0.99, 0.999, 1)
Sbreaks =qchisq(Sprob, df=2)/2
Sbreaks = pmin(Sbreaks, 1000)
Sbreaks[1] = -10
SbreaksC = rev(- Sbreaks)
colDat2 = mapmisc::colourScale(profileLogL[,'profile'], style='fixed',
                               breaks=SbreaksC,
                               col='Spectral', rev=TRUE)

interp2 = mgcv::gam(profile ~ s(sumLogRange, nugget, k=16, m=3,fx=TRUE), data=toUse2)
prof2list = list(sumLogRange=seq(15, 23, len=301),
                 nugget=seq(0, 0.6, len=301))
prof2 = do.call(expand.grid, prof2list)
prof2$z = predict(interp2, prof2)

col2 = mapmisc::colourScale(as.vector(prof2[,'z']), breaks=SbreaksC, col=colDat2$col, style='fixed')
colPoints = mapmisc::colourScale(toUse2[,'profile'], breaks=SbreaksC, col=col2$col, style='fixed')


plot(prof2[,c('sumLogRange','nugget')], col=col2$plot, pch=15)
points(toUse2[,c('sumLogRange','nugget')], col='black', cex=0.8)
mapmisc::legendBreaks('topright', breaks = rev(Sprob), col=colDat2$col)
```



2-dimensional profile plot of sumLogRange vs shape

```
profileLogL <- profileLogLik[ , c('sumLogRange','logshape', 'shape',  'profile')]

transFun = function(xx) xx^(-0.5)
invTransFun = function(xx) xx^(-2)

Sprob = c(0, 0.1, 0.2, 0.5, 0.8, 0.95, 0.99, 0.999, 1)
Sbreaks =qchisq(Sprob, df=2)/2
Sbreaks = pmin(Sbreaks, 1000)
Sbreaks[1] = -10
SbreaksC = rev(-Sbreaks)
colDat2 = mapmisc::colourScale(profileLogL[,'profile'], style='fixed',
                               breaks=SbreaksC,
                               col='Spectral', rev=TRUE)

Sshape = c(0.1, 1/4, 0.5, 1, 2, 5, 10, 100)

profForChull = cbind(
  sumLogRange = profileLogL$sumLogRange,
  shapeTrans = transFun(profileLogL$shape),
  profile=profileLogL$profile)


theHull = geometry::convhulln(profForChull)
```

```
toTest = profForChull[as.vector(theHull), ]
toTest[,'profile'] = toTest[,'profile'] - 0.1
inHull = geometry::inhulln(theHull, as.matrix(toTest))
toUse = toTest[inHull,]
toUse2 =  toUse[!duplicated(toUse[,c('sumLogRange','shapeTrans')]), ]


if(TRUE) {
  profForChullSub = profForChull[profileLogL$shape > 1 & profileLogL$shape < 8, ]
  theHullSub = geometry::convhulln(profForChullSub)
  toTestSub = profForChullSub[as.vector(theHullSub), ]
  toTestSub[,'profile'] = toTestSub[,'profile'] - 0.1
  inHullSub = geometry::inhulln(theHullSub, as.matrix(toTestSub))
  toUseSub = toTestSub[inHullSub,]
  toUse2 = rbind(toUse, toUseSub)
  toUse2 = toUse2[!duplicated(toUse2[,c('sumLogRange','shapeTrans')]), ]
}

theGridList = list(sumLogRange = seq(15, 23.3, len=201),
                   logshape = seq(-1.78, 4.6, len=201))
theGrid = do.call(expand.grid, theGridList)

theInterp = akima::interp(
  toUse2[,'sumLogRange'], toUse2[,'shapeTrans'], toUse2[,'profile'],
  theGridList$sumLogRange,
  transFun(exp(theGridList$logshape)),
  extrap=FALSE, linear=TRUE)


col2 = mapmisc::colourScale(as.vector(theInterp$z), breaks=SbreaksC, col=colDat2$col, style='fixed')
colPoints = mapmisc::colourScale(toUse2[,'profile'], breaks=SbreaksC, col=col2$col, style='fixed')

plot(theGrid[,'sumLogRange'],theGrid[,'logshape'], col=col2$plot, pch=15, ylab='logshape', xlab='sumLogR
points(toUse2[,'sumLogRange'], log(invTransFun(toUse2[, 'shapeTrans'])), cex=0.8)
```


logshape
4
3
2
1
0
−1
−2
16
18
20
22
sumLogRange