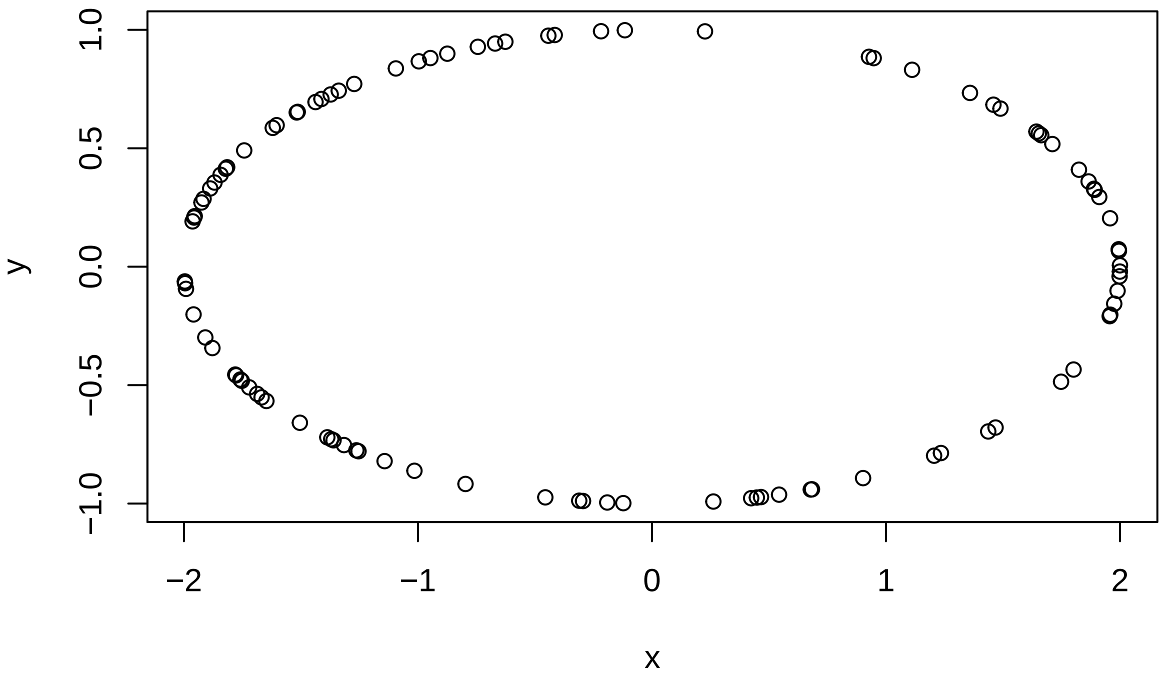
Coastal Kriging R code

The code below demonstrates the application of simple kriging and coastal kriging on the simulated and real data of the GuLF STUDY.

We simulate points from a Gaussian process along an ellipse so that the parameter t would be the arc-length between the points.

```r
num.sim=100
set.seed(111)
l<-runif(num.sim, 0, pi*2)
l<-sort(l)
x<-2*cos(l)
y<-sin(l)
#evaluate elliptic integral using series expansion to get arc-length
e<-sqrt(1-1/4)
t<-l*2*(1-(0.5^2*e^2)-(0.5^2*0.75^2*e^4/3)-(0.5^2*0.75^2*(5/6)^2*e^6/5)-
             (0.5^2*0.75^2*(5/6)^2*(7/8)^2*e^8/7))
xy<-cbind(x, y)
plot(x, y)
```



The following is a function that takes in successive x and y coordinates and returns the corresponding line segments mesurements along the coast.

```r
line.seg<-function(x,y){
  t<-rep(0,length(x))
  xy<-cbind(x,y)
  for ( j in 2:length(x)){
    t[j]<-sqrt((xy[j,1]-xy[j-1,1])^2+(xy[j,2]-xy[j-1,2])^2)
  }
  t<-cumsum(t)
  return(list(t=t))
}
```

We apply it to our simulated data, so now we have $t2$ as line segment length to be used in equation (3).

```
t2<-line.seg(x,y)$t
```

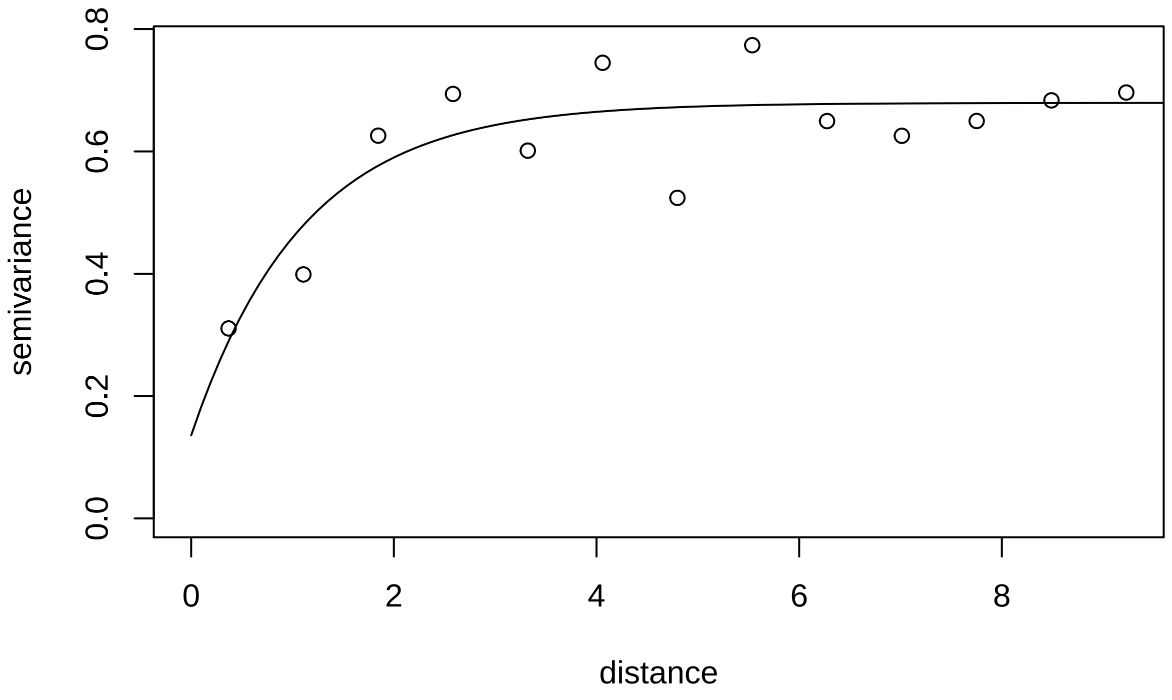
We simulate from a Gaussian process with exponential covariance that depends on t.

```
#variance of the response variable
distance<-abs(outer(t ,t,"-"))
tau.p<-0.1
sigma.p<-1
vv<-sigma.p*exp(-1*distance)
beta.p<-0
#simulation of the response variable
z<-NULL
x0<-rep(1,num.sim)
set.seed(0000)
z<-rmvnorm(1, beta.p%*%t(cbind(x0)),vv+tau.p*diag(num.sim) )
```

```
#choose the training and hold out samples
test<-seq(1,num.sim,4)
train<-(seq(1,num.sim,1)[-test])
n1<-length(train)
n2<-length(test)
```

We use the variogram to get fixed values for ϕ and α in the simplified models and to get starting values for ϕ, σ^2 and τ^2 in the full hierarchical models.

```
curve<-data.frame((z[train]))
vario <- variog(coords = cbind(t[train], rep(0,n1)), data = curve)
initial.values <-expand.grid(seq(0.01,1,l=5), seq(0.01,1,l=5))
fit <- variofit(vario, ini.cov.pars = initial.values,  cov.model = "exponential")
sigma<-fit$cov.pars[1]
phi<-fit$cov.pars[2]
tau<-fit$nugget
alpha <- tau/sigma
plot(vario)
lines(fit)
```



```
#Variogram to get initial values for the simple kriging
curve<-data.frame((z[train]))
varios <- variog(coords = cbind(x[train], y[train]), data = curve)
fits <- variofit(varios, ini.cov.pars = initial.values,  cov.model = "exponential")
sigmas<-fits$cov.pars[1]
phis<-fits$cov.pars[2]
taus<-fits$nugget
alphas <- taus/fits$cov.pars[1]
```

The following function uses the function spLM from the package spBayes to get the posterior samples in a full hierarchical model.

```
priors1 <- list("beta.flat",
                 "phi.Unif"=c(0.8, 30), "sigma.sq.IG"=c(2,2),
                 "tau.sq.IG"=c(2, 2))
starting1 <- list("phi"=phi, "sigma.sq"=sigma, "tau.sq"=tau)
tuning1 <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
Geostat.full<-function(x,y,data,n.samples){
  sp <-spLM(data~1, coords=cbind(x, y), tuning=tuning1, starting=starting1, n.samples=5000,
                     priors=priors1,cov.model="exponential")
  burn.in <- 0.5*n.samples
  m <- spRecover(sp , start=burn.in, verbose=FALSE)
  beta0<-(m$p.beta.recover.samples[,1])
  sigma<-(m$p.theta.recover.samples[,1])
  tau<-(m$p.theta.recover.samples[,2])
  phi<-(m$p.theta.recover.samples[,3])
  spatial<-as.matrix(m$p.w.recover.samples)
  return(list(m=m,beta0=beta0, sigma=sigma, tau=tau,phi=phi,spatial=spatial))
}
```

We apply the Geostat.full function to our simulated data set using the approximations in (3) and (4).

```r
sim1a<-Geostat.full(x=t[train],y=rep(0,n1),data=z[train],
                     n.samples=5000)
beta0sim1a<-sim1a$beta0
sigmasim1a<-sim1a$sigma
tausim1a<-sim1a$tau
phisim1a<-sim1a$phi
spsim1a<-sim1a$spatial
msim1a<-sim1a$m
round(summary(msim1a$p.theta.recover.samples)$quantiles,3)
round(summary(msim1a$p.beta.recover.samples)$quantiles,3)

sim1b<-Geostat.full(x=t2[train],y=rep(0,n1),data=z[train],
                     n.samples=5000)
beta0sim1b<-sim1b$beta0
sigmasim1b<-sim1b$sigma
tausim1b<-sim1b$tau
phisim1b<-sim1b$phi
spsim1b<-sim1b$spatial
msim1b<-sim1b$m
round(summary(msim1b$p.theta.recover.samples)$quantiles,3)
round(summary(msim1b$p.beta.recover.samples)$quantiles,3)
```

The following function uses the function bayesGeostatExact from the package spBayes to get the posterior samples in a simplified hierarchical model.

```r
Geostat.exact<-function(x,y,data,n.samples,beta.prior.mean,beta.prior.precision,
                        sigma.sq.prior.shape, sigma.sq.prior.rate,phi,alpha ){
  exact <- bayesGeostatExact(data~1, n.samples=n.samples,
                             beta.prior.mean=beta.prior.mean,
                             beta.prior.precision=beta.prior.precision,
                             coords=cbind(x, y), phi=phi, alpha=alpha,
                             sigma.sq.prior.shape=sigma.sq.prior.shape,
                             sigma.sq.prior.rate=sigma.sq.prior.rate)
  beta0<-(exact$p.samples[,1])
  sigma<-(exact$p.samples[,2])
  tau<-(exact$p.samples[,3])
  spatial<-as.matrix(exact$sp.effects)
  return(list(beta0=beta0, sigma=sigma, tau=tau,spatial=spatial,exact))
}
```

We apply the Geostat.exact function to coastal kriging using approximations in (3) and (4) and to Euclidean distance kriging for comparison.

```r
sim2a<-Geostat.exact(t[train], rep(0,n1),z[train],5000,0,0,2,2,phi,alpha)
beta0sim2a<-sim2a$beta0
sigmasim2a<-sim2a$sigma
tausim2a<-sim2a$tau
spsim2a<-sim2a$spatial
print(summary(sigmasim2a))

sim2b<-Geostat.exact(t2[train], rep(0,n1),z[train],5000,0,0,2,2,phi,alpha)
beta0sim2b<-sim2b$beta0
sigmasim2b<-sim2b$sigma
tausim2b<-sim2b$tau
spsim2b<-sim2b$spatial
```

```
simsk<-Geostat.exact(x[train], y[train],z[train],5000,0,0,2,2,phis,alphas)
beta0simsk<-simsk$beta0
sigmasimsk<-simsk$sigma
tausimsk<-simsk$tau
spsimsk<-simsk$spatial
print(summary(tausimsk))
```

The following function can be used by specifying the training data, test data, posterior samples of the parameters τ^2, σ^2, ϕ and β, training data coordinates, test data coordinates, distance matrix and specifying whether it's a full hierarchical or simplified model and if it's a full model we specify the object m resulting from spRecover. The function gives the predicted values with 95% C.I. and goodness of fit measures MSPE and DIC.

```
n.samples=5000
goodnessoffit<-function(datatrain, datatest,sigma,phi,tau,beta0,xtrain, xtest,ytrain,ytest,dist,simple,
  v<-vv<-muhat<-yhat<-vector("list")
  pred<-upper<-lower<-NULL
  pdf<-NULL
  n1<-length(xtrain)
  n2<-length(xtest)
  beta0.mean<-mean(beta0)
  sigma.mean<-mean(sigma)
  tau.mean<-mean(tau)
  for(i in 1:(0.5*n.samples)){
    #unconditional variance
    v[[i]]<-sigma[i]*exp(-phi[i]*dist)+tau[i]*diag(n2+n1)
    #conditional variance
    vv[[i]]<- v[[i]][(n1+1):(n2+n1), (n1+1):(n2+n1)]-(v[[i]][(n1+1):(n2+n1), 1:n1])%*%
      chol2inv(chol(v[[i]][1:n1,1:n1]))%*%(t(v[[i]][(n1+1):(n2+n1), 1:n1]))
    #conditional mean
    muhat[[i]]<-t(beta0[i]%*%t(rep(1, (n2))))+(v[[i]][(n1+1):(n2+n1), 1:n1])%*%
      chol2inv(chol(v[[i]][1:n1,1:n1]))%*%t((t(datatrain)-beta0[i]*t(x0[1:n1])))
    set.seed(i)
    # sample from conditional normal on the training sample and 95% intervals
    yhat[[i]]<-rmvnorm(1,muhat[[i]],round(vv[[i]],4))
    #density to be used in DIC
    pdf[i]<-dmvnorm(datatest,muhat[[i]],round(vv[[i]],4))
  }
  if(simple=="TRUE"){
    yhat.mat<-matrix(unlist(yhat), nrow=(n.samples*0.5), ncol=n2, byrow=T)
      pred<-apply(yhat.mat,2,mean)
      upper<-pred+1.96*(sqrt(apply(yhat.mat,2,var)))
      lower<-pred-1.96*(sqrt(apply(yhat.mat,2,var)))
  }else{
  m.pred <- spPredict(m, pred.covars=matrix(rep(1,n2),n2,1),
                      pred.coords=cbind(xtest, ytest),
                      start=0.5*n.samples)
  pred <- as.numeric(apply(m.pred$p.y.predictive.samples, 1, mean))
  upper<-pred+1.96*sqrt(apply(m.pred$p.y.predictive.samples, 1, var))
  lower<-pred-1.96*sqrt(apply(m.pred$p.y.predictive.samples, 1, var))
  }
  v.mean<-sigma.mean*exp(-phi[i]*dist)+tau.mean*diag(n1+n2)
  vv.mean<- v.mean[(n1+1):(n2+n1), (n1+1):(n2+n1)]-((v.mean[(n1+1):(n2+n1), 1:n1])%*%
             chol2inv(chol(v.mean[1:n1,1:n1])))%*%(t(v.mean[(n1+1):(n2+n1), 1:n1]))
  muhat.mean<-t(beta0.mean%*%t(rep(1, (n2))))+((v.mean[(n1+1):(n2+n1), 1:n1])%*%
              chol2inv(chol(v.mean[1:n1,1:n1])))%*%t((t(datatrain)-beta0.mean%*%t(x0[1:n1])))
  #DIC calculation
  D.bar <- -2*log(mean(pdf))
  D.hat <- -2*log(dmvnorm(datatest,muhat.mean,vv.mean))
  pD <- D.bar - D.hat
  DIC <- pD+D.bar
  #MSPE calculation
  mse<-sum((exp(datatest)-exp(pred))^2)/n2
  return(list(pred=pred,upper=upper, lower=lower,DIC=DIC,MSE=mse))
}
```

We apply the previuosly described function goodnessoffit to our 4 coastal kriging models and simple kriging model.

```
#First calculate distance matrices to be used for models using arc-length, models using line segments a
tpa<-c(t[train],t[test])
dista<-abs(outer(tpa,tpa,"-"))
tpb<-c(t2[train],t2[test])
distb<-abs(outer(tpb,tpb,"-"))
dist<-as.matrix(dist(data.frame(cbind(x[c(train,test)],y[c(train,test)])),upper=TRUE, diag=TRUE,method=

gofsim1a<-goodnessoffit(datatrain=z[train], datatest=z[test], sigma=sigmasim1a, phi=phisim1a, tau=tausi
                         beta0=beta0sim1a, xtrain=t[train], xtest=t[test], ytrain=rep(0,n1),
                        ytest=rep(0,n2), dist=dista, simple="FALSE", m=msim1a)
DICsim1a<-gofsim1a$DIC
msesim1a<-gofsim1a$MSE
predsim1a<-gofsim1a$pred
uppersim1a<-gofsim1a$upper
lowersim1a<-gofsim1a$lower

gofsim1b<-goodnessoffit(datatrain=z[train], datatest=z[test], sigma=sigmasim1b, phi=phisim1b, tau=tausi
                         beta0=beta0sim1b, xtrain=t2[train], xtest=t2[test], ytrain=rep(0,n1),
                        ytest=rep(0,n2), dist=distb, simple="FALSE", m=msim1b)
DICsim1b<-gofsim1b$DIC
msesim1b<-gofsim1b$MSE
predsim1b<-gofsim1b$pred
uppersim1b<-gofsim1b$upper
lowersim1b<-gofsim1b$lower

gofsim2a<-goodnessoffit(datatrain=z[train], datatest=z[test], sigma=sigmasim2a, phi=rep(phi,5000), tau=
                         beta0=beta0sim2a, xtrain=t[train], xtest=t[test], ytrain=rep(0,n1),
                        ytest=rep(0,n2), dist=dista, simple="TRUE", m=NULL)
DICsim2a<-gofsim2a$DIC
msesim2a<-gofsim2a$MSE
predsim2a<-gofsim2a$pred
uppersim2a<-gofsim2a$upper
lowersim2a<-gofsim2a$lower

gofsim2b<-goodnessoffit(datatrain=z[train], datatest=z[test], sigma=sigmasim2b, phi=rep(phi,5000), tau=
                         beta0=beta0sim2b, xtrain=t2[train], xtest=t2[test], ytrain=rep(0,n1),
                        ytest=rep(0,n2), dist=distb, simple="TRUE", m=NULL)
DICsim2b<-gofsim2b$DIC
msesim2b<-gofsim2b$MSE
```

```
predsim2b<-gofsim2b$pred
uppersim2b<-gofsim2b$upper
lowersim2b<-gofsim2b$lower

gofsimssk<-goodnessoffit(datatrain=z[train], datatest=z[test], sigma=sigmasimsk, phi=rep(phis,5000), tau
                            beta0=beta0simsk, xtrain=x[train], xtest=x[test], ytrain=y[train],
                           ytest=y[test], dist=dist, simple="TRUE", m=NULL)
DICsimsk<-gofsimssk$DIC
msesimsk<-gofsimssk$MSE
predsimsk<-gofsimssk$pred
uppersimsk<-gofsimssk$upper
lowersimsk<-gofsimssk$lower
```

Then we can plot the true values versus the predicted values and 95% C.I

```
plotCI(z[test], predsim1a,ui=uppersim1a, li=lowersim1a,xlab="True", ylab="Predicted",
       main="Model 1a",
       cex.main=0.9,cex=0.9)
abline(0,1)
```

```
plotCI(z[test], predsim1b,ui=uppersim1b, li=lowersim1b,xlab="True", ylab="Predicted",
       main="Model 1b",
       cex.main=0.9,cex=0.9)
abline(0,1)
```

```
plotCI(z[test], predsim2a,ui=uppersim2a, li=lowersim2a,xlab="True", ylab="Predicted",
       main="Model 2a",
       cex.main=0.9,cex=0.9)
abline(0,1)
```

```
plotCI(z[test], predsim2b,ui=uppersim2b, li=lowersim2b,xlab="True", ylab="Predicted",
       main="Model 2b",
       cex.main=0.9,cex=0.9)
abline(0,1)
```

```
plotCI(z[test], predsimsk,ui=uppersimsk, li=lowersimsk,xlab="True", ylab="Predicted",
       main="Simple Kriging",
       cex.main=0.9,cex=0.9)
abline(0,1)
```

We define the following function that calculates the Kullback-Liberur criterion.

```
kullbackliberur<-function(truev, sigma, phi, tau, beta, distance){
  kl<-NULL
  vm<-vector("list")
  for(j in 1:(0.5*n.samples)){
  vm[[j]]<-sigma[j]*exp(-phi[j]*distance)+tau[j]*diag(num.sim)
  kl[j]<-tr(solve(vm[[j]])%*%(v0))+t(rep(beta[j],num.sim))%*%
    solve(vm[[j]])%*%(rep(beta[j],num.sim))-num.sim+determinant(vm[[j]],log=TRUE)$modulus-determinant(v
  }
  meankl<-0.5*mean(kl)
  return(meankl)
}

#true variance is
v0<-vv[c(train,test),c(train,test)]+tau*diag(num.sim)
```

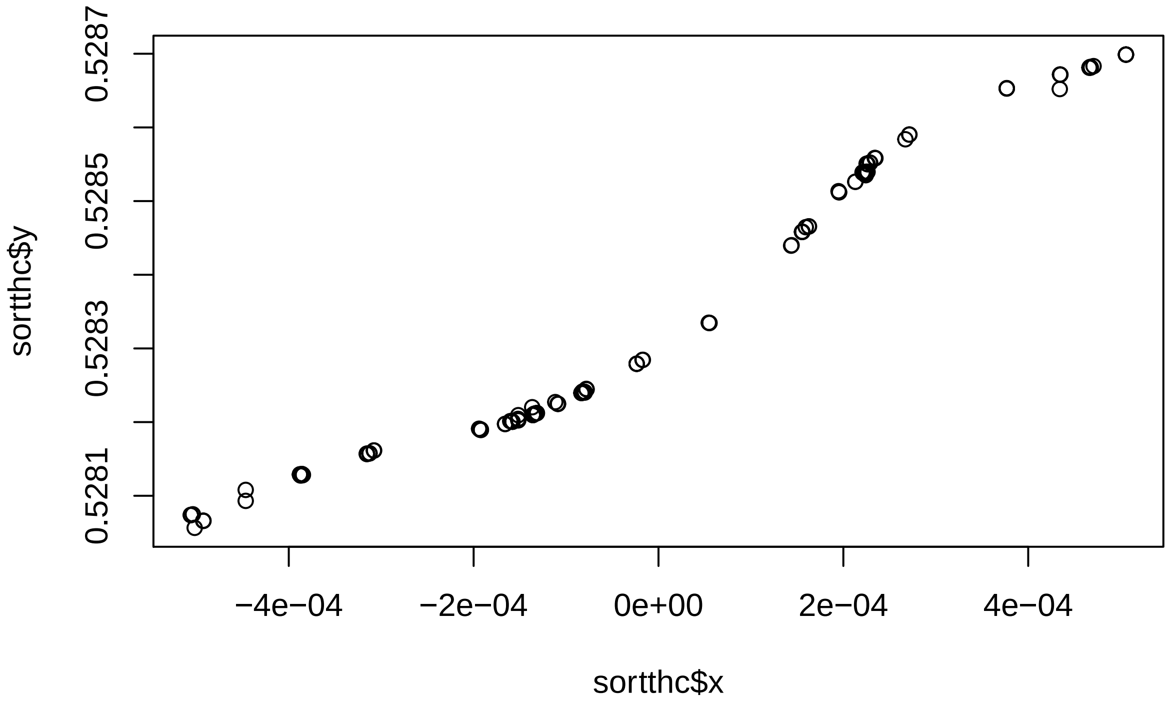
```
kl1a<-kullbackliberur(v0, sigmasim1a, phisim1a, tausim1a, beta0sim1a,dista)
kl1b<-kullbackliberur(v0, sigmasim1b, phisim1b, tausim1b, beta0sim1b,distb)
kl2a<-kullbackliberur(v0, sigmasim2a, rep(phi,5000), tausim2a, beta0sim2a,dista )
kl2b<-kullbackliberur(v0, sigmasim2b, rep(phi,5000), tausim2b, beta0sim2b,distb )
klsk<-kullbackliberur(v0, sigmasimsk, rep(phis,5000), tausimsk, beta0simsk,dist )
```

Now the real data of the GuLF STUDY.

```
data<-read.csv("/Users/n_a_abdallah/Desktop/spatial/Project1/data 2.csv")
THC<-subset(data, data$Reported_Analyte=="Total Hydrocarbons")
sortthc<-THC[order(THC$device_longitude),]
```

Sinusoidal projection is used as follows as a map projection.

```
mapping<-mapproject(sortthc$device_longitude, sortthc$device_latitude, proj= "sinusoidal", parameters=NU
sortthc$x<-mapping$x
sortthc$y<-mapping$y
plot(sortthc$x,sortthc$y)
```



We log transform total hydrocarbon enhaled by workers to achieve normality.

```
sortthc$log<-log(sortthc$Result2)
```

We apply the line.seg function that takes in x and y coordinates and returns the corresponding line segments mesurements along the coast.

```
#Radius of earth
radius<-6371
tcalc<-line.seg(sortthc$x,sortthc$y)
sortthc$tthc<-tcalc$t
```

We remove duplicated locations that cause numerical instability.

```
longthc<-sortthc$device_longitude
latthc<-sortthc$device_latitude
```

```
sortthc$thc<-NULL
for ( j in 2:length(sortthc$log)){
  sortthc$thc[j-1]<-sqrt((longthc[j]-longthc[j-1])^2+(longthc[j]-longthc[j-1])^2)
}
thc<-c(0,sortthc$thc)
sortthc2<-sortthc
sortthc2<-sortthc2[!duplicated(sortthc2$thc),]
```

We choose 12 observations as our hold out data set and 48 as our training data set.

```
testthc<-seq(1,60,5)
trainthc=c(seq(1,length(sortthc2$log),1)[-testthc])
n1=length(trainthc)
n2<-length(testthc)
```

We apply the Geostat.full function to our data set to get results of model 1b and simple kriging.

```
x0<-rep(1,length(sortthc2$log))
priors1 <- list("beta.flat",
                "phi.Unif"=c(0.8, 30), "sigma.sq.IG"=c(2,2),
                "tau.sq.IG"=c(2, 2))
starting1 <- list("phi"=14, "sigma.sq"=2, "tau.sq"=2)
tuning1 <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)

model1b<-Geostat.full(x=radius*sortthc2$tthc[trainthc],y=rep(0,n1),data=sortthc2$log[trainthc],
                      n.samples=5000)
beta01b<-model1b$beta0
sigma1b<-model1b$sigma
tau1b<-model1b$tau
phi1b<-model1b$phi
sp1b<-model1b$spatial
m1b<-model1b$m
#Euclidean distance
modelfsk<-Geostat.full(x=radius*sortthc2$x[trainthc],y=radius*sortthc2$y[trainthc],data=sortthc2$log[tra
beta0fsk<-modelfsk$beta0
sigmafsk<-modelfsk$sigma
taufsk<-modelfsk$tau
phifsk<-modelfsk$phi
spsk<-modelfsk$spatial
mfsk<-modelfsk$m
```

We apply the goodnessoffit function to our two models for comparison.

```
distthc<-radius*abs(outer(sortthc2$tthc[c(trainthc,testthc)],sortthc2$tthc[c(trainthc,testthc)],"-"))
gof1b<-goodnessoffit(datatrain=sortthc2$log[trainthc], datatest=sortthc2$log[testthc],sigma=sigma1b,phi
DIC1b<-gof1b$DIC
pred1b<-gof1b$pred
mse1b<-gof1b$MSE

distsk<-as.matrix(dist(data.frame(cbind(sortthc2$x[c(trainthc,testthc)],sortthc2$y[c(trainthc,testthc)]
gofsk<-goodnessoffit(datatrain=sortthc2$log[trainthc], datatest=sortthc2$log[testthc],sigma=sigmafsk,ph
DICsk<-gofsk$DIC
predsk<-gofsk$pred
msesk<-gofsk$MSE
```

Now we interpolate new set of coordinates and get the predicted values using model 1b to plot on Google map

```r
#first interpolate new points
newx<-approx((sortthc$device_longitude),(sortthc$device_latitude),n=100)
mappingnew<-mapproject(newx$x, newx$y, proj= "sinusoidal", parameters=NULL, orientation=NULL)

#Calculation of the parameter corresponding to line segment approximations to the coast
newtthc<-rep(0,100)
newtthc<-line.seg(mappingnew$x,mappingnew$y)
for ( j in 2:100){
  newtthc[j-1]<-sqrt((mappingnew$x[j]-mappingnew$x[j-1])^2+(mappingnew$y[j]-mappingnew$y[j-1])^2)
}
newtthc<-c(0,newtthc)
newtthc2<-cumsum(newtthc)
m.predthc <- spPredict(m1b, pred.covars=matrix(rep(1,100),100,1),
                       pred.coords=cbind(radius*newtthc2, rep(0,(100))),
                       start=0.5*n.samples)
y.hatthc.new <- as.numeric(apply(m.predthc$p.y.predictive.samples, 1, mean))
```

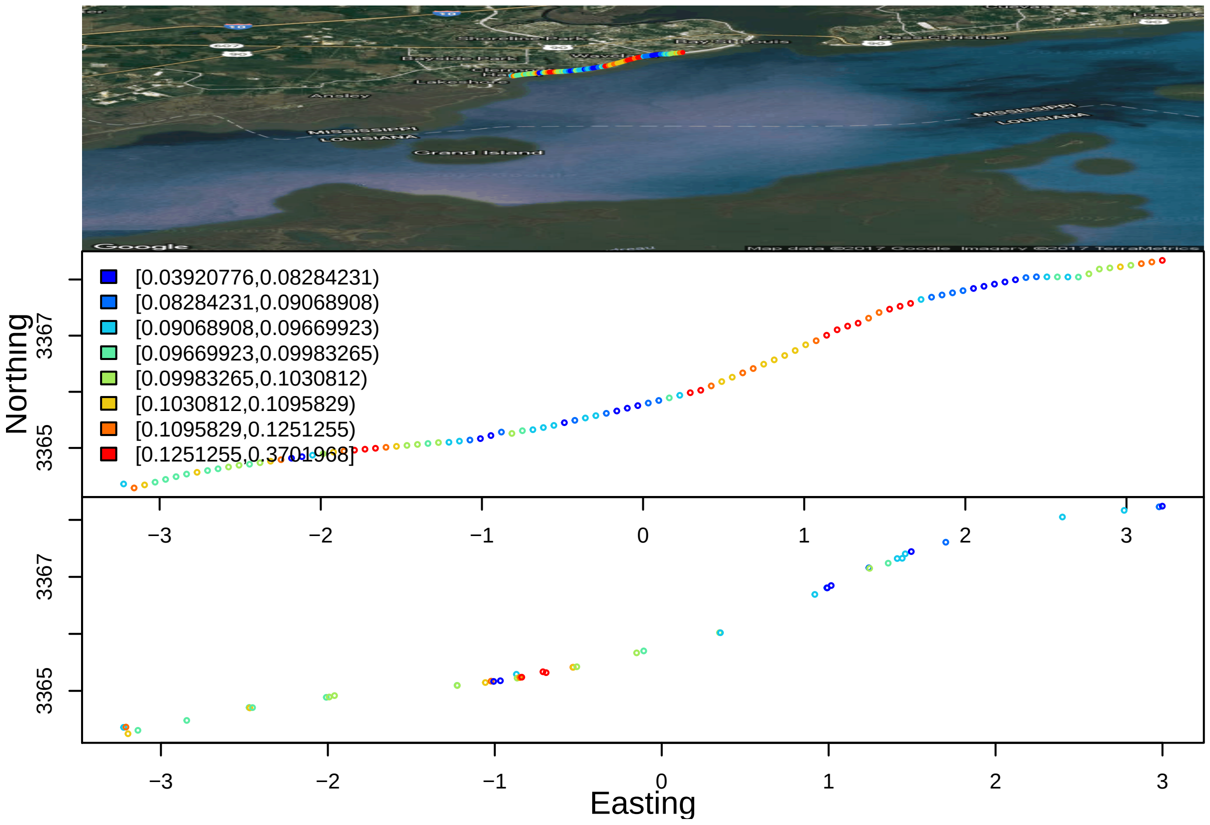
Now we plot Google map with predictions at new location and also a plot of spatial residuals from our observed training sample.

```r
col.br=colorRampPalette(c("blue", "cyan", "yellow", "red"))
col.pal <- col.br(5)
quantthc<-classIntervals(exp(y.hatthc.new) , style="quantile")
quant.colthc<-findColours(quantthc, col.pal)

par(mfrow=c(3,1), oma=c(3,3,2,2))
MyMapthc <- GetMap.bbox(lonR = c(-89.38,-89.35), latR = c(30,30.35),
                        size=c(640,640), maptype = "hybrid")
#We can also obtain the posterior means and standard deviations of the spatial residuals and plot on th

w.hat.mu1 <- apply(sp1b,1,mean)
w.hat.sd1 <- apply(sp1b,1,sd)
PlotOnStaticMap(MyMapthc)
convert_pointsthcnew <- LatLon2XY.centered(MyMapthc, newx$y, newx$x)

points(convert_pointsthcnew$newX, convert_pointsthcnew$newY, col = quant.colthc, pch=19,cex=0.3)
plot(radius*mappingnew$x,radius*mappingnew$y, col=quant.colthc, cex=0.5,ylab="Northing", xlab="Easting")
mtext(text="Northing",side=2,line=2)
     legend("topleft",fill=attr(quant.colthc,"palette"),legend=
         names(attr(quant.colthc,"table")),bg="white", cex=1, bty="n")
plot(radius*sortthc2$x[trainthc],radius*sortthc2$y[trainthc], col=quant.colthc, cex=0.5,ylab="Northing"
     mtext(text="Easting",side=1,line=2)
```



Grand Island
Bay Saint Louis
Pass Christian
Waveland
Ansley
Lakeshore
MISSISSIPPI
LOUISIANA
Google
Map data ©2017 Google Imagery ©2017 TerraMetrics
[0.03920776,0.08284231)
[0.08284231,0.09068908)
[0.09068908,0.09669923)
[0.09669923,0.09983265)
[0.09983265,0.1030812)
[0.1030812,0.1095829)
[0.1095829,0.1251255)
[0.1251255,0.3701968]
Northing
3365
3367
−3
−2
−1
0
1
2
3
Easting