Arquivos de Comandos

```
rm(list=ls())
load("C:\\Users\\user\\Desktop\\geo.analise\\Análise\\Setor_01\\Setor1.dados.RDa
ta")
require(Rcmdr)
require(maptools)     ## funções para importação/exportação e manipulação de
mapas e dados geográficos
gpclibPermit()        ## função para habilitar licençaa de uso
require(sp)           ## (classes) para representação de dados espaciais no R
require(geoR)         ##Principal pacote utilizado nas análises
require(coda)         ##Para realizar a análise de convergencias
require(xtable)

head(Setor1.dados,5)
attach(Setor1.dados)

fator=as.factor(c(Estação))
Meses=as.factor(Mês)
Precipitação<- tapply(Total,fator,median);
lat<- tapply(Latitude,fator,mean);
long<- tapply(Longitude,fator,mean);

Setor1<-data.frame(long,lat,Precipitação)
rr=cbind(Setor1[1:67,], Setor1[68:134,], Setor1[135:201,], Setor1[202:268,])
dados = xtable(rr, caption =
"Serie histórica de precipitação entre 1994 a 2011 no Estado da Paraíba", label
= "dados", digits = 2)
print(tab)


dG <- as.geodata(Setor1, coords.col=c(1,2), data.col=3)
#Verificando se necessita de transformação

pdf('boxplot.pdf')
boxcox(dG, lambda = seq(-2.2, 3, 1/1055))
dev.off()
#...Necessita de transformação, o número 1 não esta "dentro da região."
box=boxcox(dG, lambda = seq(-2.2, 3, 1/1055),plotit=F)
x = box$x; y = box$y;
lambda=cbind(y,x)[order(y,x),][length(y),2]#Obtendo o valor de x correspondente
ao ponto máximo da função boxcox

class(dG)
points(dG)
summary(dG)
d.max=summary(dG)$distances.summary[2]/2
pdf('dG.pdf')
plot(dG, lam=lambda, low=T)
dev.off()
points(dG, pt.div="quint", cex.min=1, cex.max=1)
#Variograma
v <- variog(dG, lam=lambda, max.dist=d.max)
plot(v)

vC <- variog(dG, lam=lambda, max.dist=d.max, option="cloud")
plot(vC)
```

```
vS <- variog(dG, lam=lambda, max.dist=d.max, option="smooth")
plot(vS)

## escolhendo funcao de variograma
plot(v)
ef <- eyefit(v)
ef

## lendo dados tipo shapefile
ctba <- readShapePoly("Limite_Estadual.shp")
#Amarzenando as bordas do Estado em um objeto
bordas<-ctba@polygons[[1]]@Polygons[[2]]@coords
#### inserindo as bordas nos dados #####
dG$borders<-bordas
## visualizando os dados  dentro do grid de predição
points(dG, lam=lambda)

### modelos candidatos ###

plot(dG, lam=lambda, low=T)
plot(dG, lam=lambda, low=T,trend=~coords[,1])
plot(dG, lam=lambda, low=T,trend=~coords[,2])
plot(dG, lam=lambda, low=T, trend=~coords[,1]+I(coords[,1]^2))
#..... ..................
#### VARIOGRAMAS ####       #Falta colocar com diferentes modelos
k.v1 <-variog(dG, lam=lambda, max.dist=d.max)
k.v2 <-variog(dG, lam=lambda, max.dist=d.max,trend=~coords[,1])
k.v3 <-variog(dG, lam=lambda, max.dist=d.max,trend=~coords[,2])
k.v4 <-variog(dG, lam=lambda, max.dist=d.max,trend=~coords[,1]+I(coords[,1]^2))

plot(k.v4)

#Envolope Simulado
env = variog.mc.env(dG, obj.var=k.v4, nsim=1000)
pdf('envelope.pdf')
plot(k.v4, envelope=env, xlab="Distancia", ylab="Semivariancia", axes=T,
frame=T, col="blue", pch=19)
dev.off()
par(mfrow=c(2,2))
plot(k.v1)
plot(k.v2)
plot(k.v3)
plot(k.v4)
```

Escolhendo o modelo por meio da máxima verossimilhança

```
# escolhendo o modelo através da máxima verossimilhança #
ml1.gaussian <- likfit(dG, cov.model="gaussian", lam=lambda, ini=c(11.6,1.15),
nug=1.45)
ml2.gaussian <- likfit(dG, cov.model="gaussian", lam=lambda, trend=~coords[,1] ,
ini=c(11.6,1.15), nug=1.45);
ml3.gaussian <- likfit(dG, cov.model="gaussian", lam=lambda, trend=~coords[,2] ,
ini=c(11.6,1.15), nug=1.45);
ml4.gaussian <- likfit(dG, cov.model="gaussian", lam=lambda,
trend=~coords[,1]+I(coords[,1]^2), ini=c(5,2), nug=2)

grau.ml2.ml1.gaussian <- ml2.gaussian$npars - ml1.gaussian$npars;
grau.ml2.ml1.gaussian # Graus de liberdade de qui-quadrado
val.tab.guassian <- qchisq(0.95,grau.ml2.ml1.gaussian);val.tab.guassian
Log.ml2.ml1.gaussian <- 2*(logLik(ml2.gaussian) - logLik(ml1.gaussian));
Log.ml2.ml1.gaussian

grau.ml3.ml1.gaussian <- ml3.gaussian$npars - ml1.gaussian$npars;
grau.ml3.ml1.gaussian    # Grau de liberdade
val.tab1.gaussian <- qchisq(0.95,grau.ml3.ml1.gaussian); val.tab1.gaussian
Log.ml3.ml1.gaussian <- 2*(logLik(ml3.gaussian) - logLik(ml1.gaussian));
Log.ml3.ml1.gaussian

grau.ml4.ml1.gaussian <- ml4.gaussian$npars - ml1.gaussian$npars;
grau.ml4.ml1.gaussian    # Grau de liberdade
val.tab2.gaussian <- qchisq(0.95,grau.ml4.ml1.gaussian); val.tab2.gaussian
Log.ml4.ml1.gaussian <- 2*(logLik(ml4.gaussian) - logLik(ml1.gaussian));
Log.ml4.ml1.gaussian

#Modelo Exponential
ml1.exponential <- likfit(dG, cov.model="exponential", lam=lambda,
ini=c(11.6,1.15), nug=1.45)
ml2.exponential <- likfit(dG, cov.model="exponential", lam=lambda,
trend=~coords[,1] , ini=c(11.6,1.15), nug=1.45);
ml3.exponential <- likfit(dG, cov.model="exponential", lam=lambda,
trend=~coords[,2] , ini=c(11.6,1.15), nug=1.45);
ml4.exponential <- likfit(dG, cov.model="exponential", lam=lambda,
trend=~coords[,1]+I(coords[,1]^2), ini=c(5,2), nug=2);

grau.ml2.ml1.exponential <- ml2.exponential$npars - ml1.exponential$npars;
grau.ml2.ml1.exponential            # Graus de liberdade de qui-quadrado
val.tab.exponential <- qchisq(0.95,grau.ml2.ml1.exponential);
val.tab.exponential
Log.ml2.ml1.exponential <- 2*(logLik(ml2.exponential) -
logLik(ml1.exponential)); Log.ml2.ml1.exponential
```

```
grau.ml3.ml1.exponential <- ml3.exponential$npars - ml1.exponential$npars;
grau.ml3.ml1.exponential    # Grau de liberdade
val.tab1.exponential <- qchisq(0.95,grau.ml3.ml1.exponential);
val.tab1.exponential
Log.ml3.ml1.exponential <- 2*(logLik(ml3.exponential) -
logLik(ml1.exponential)); Log.ml3.ml1.exponential

grau.ml4.ml1.exponential <- ml4.exponential$npars - ml1.exponential$npars;
grau.ml4.ml1.exponential    # Grau de liberdade
val.tab2.exponential <- qchisq(0.95,grau.ml4.ml1.exponential);
val.tab2.exponential
Log.ml4.ml1.exponential <- 2*(logLik(ml4.exponential) -
logLik(ml1.exponential)); Log.ml4.ml1.exponential


#Modelo matern com kappa = 1
ml1.matern.k2 <- likfit(dG, cov.model="matern", lam=lambda, kappa = 1,
ini=c(11.6,1.15), nug=1.45)
ml2.matern.k2 <- likfit(dG, cov.model="matern", lam=lambda, trend=~coords[,1] ,
ini=c(11.6,1.15), nug=1.45, kappa = 1)
ml3.matern.k2 <- likfit(dG, cov.model="matern", lam=lambda, trend=~coords[,2] ,
ini=c(11.6,1.15), nug=1.45, kappa = 1);
ml4.matern.k2 <- likfit(dG, cov.model="matern", lam=lambda,
trend=~coords[,1]+I(coords[,1]^2),
ini=c(5,2), nug=1.45, kappa = 1)

grau.ml2.ml1.matern.k2 <- ml2.matern.k2$npars - ml1.matern.k2$npars;
grau.ml2.ml1.matern.k2
val.tab.matern.k2 <- qchisq(0.95,grau.ml2.ml1.matern.k2); val.tab.matern.k2
Log.ml2.ml1.matern.k2 <- 2*(logLik(ml2.matern.k2) - logLik(ml1.matern.k2));
Log.ml2.ml1.matern.k2

grau.ml3.ml1.matern.k2 <- ml3.matern.k2$npars - ml1.matern.k2$npars;
grau.ml3.ml1.matern.k2    # Grau de liberdade
val.tab1.matern.k2 <- qchisq(0.95,grau.ml3.ml1.matern.k2); val.tab1.matern.k2
Log.ml3.ml1.matern.k2 <- 2*(logLik(ml3.matern.k2) - logLik(ml1.matern.k2));
Log.ml3.ml1.matern.k2

grau.ml4.ml1.matern.k2 <- ml4.matern.k2$npars - ml1.matern.k2$npars;
grau.ml4.ml1.matern.k2
val.tab2.matern.k2 <- qchisq(0.95,grau.ml4.ml1.matern.k2); val.tab2.matern.k2
Log.ml4.ml1.matern.k2 <- 2*(logLik(ml4.matern.k2) - logLik(ml1.matern.k2));
Log.ml4.ml1.matern.k2

#Modelo power.exponential
ml1.powered.exponential <- likfit(dG, cov.model="powered.exponential",
lam=lambda, ini=c(11.6,1.15), nug=1.45)
ml2.powered.exponential <- likfit(dG, cov.model="powered.exponential",
lam=lambda,
trend=~coords[,1] , ini=c(11.6,1.15), nug=1.45); ml2.powered.exponential
ml3.powered.exponential <- likfit(dG, cov.model="powered.exponential",
lam=lambda, trend=~coords[,2] , ini=c(11.6,1.15), nug=1.45);
ml3.powered.exponential
ml4.powered.exponential <- likfit(dG, cov.model="powered.exponential",
lam=lambda, trend=~coords[,1]+I(coords[,1]^2), ini=c(5,2), nug=2);
ml4.powered.exponential

grau.ml2.ml1.powered.exponential <- ml2.powered.exponential$npars -
ml1.powered.exponential$npars;
```

```
val.tab.powered.exponential <- qchisq(0.95,grau.ml2.ml1.powered.exponential);
val.tab.powered.exponential
Log.ml2.ml1.powered.exponential <- 2*(logLik(ml2.powered.exponential) -
logLik(ml1.powered.exponential)); Log.ml2.ml1.powered.exponential

grau.ml3.ml1.powered.exponential <- ml3.powered.exponential$npars -
ml1.powered.exponential$npars; grau.ml3.ml1.powered.exponential    # Grau de
liberdade
val.tab1.powered.exponential <- qchisq(0.95,grau.ml3.ml1.powered.exponential);
val.tab1.powered.exponential
Log.ml3.ml1.powered.exponential <- 2*(logLik(ml3.powered.exponential) -
logLik(ml1.powered.exponential)); Log.ml3.ml1.powered.exponential

grau.ml4.ml1.powered.exponential <- ml4.powered.exponential$npars -
ml1.powered.exponential$npars; grau.ml4.ml1.powered.exponential    # Grau de
liberdade
val.tab2.powered.exponential <- qchisq(0.95,grau.ml4.ml1.powered.exponential);
val.tab2.powered.exponential
Log.ml4.ml1.powered.exponential <- 2*(logLik(ml4.powered.exponential) -
logLik(ml1.powered.exponential)); Log.ml4.ml1.powered.exponential

#Modelo spherical
ml1.spherical <- likfit(dG, cov.model="spherical", lam=lambda, ini=c(11.6,1.15),
nug=1.45)
ml2.spherical <- likfit(dG, cov.model="spherical", lam=lambda, trend=~coords[,1]
, ini=c(11.6,1.15), nug=1.45); ml2.spherical
ml3.spherical <- likfit(dG, cov.model="spherical", lam=lambda, trend=~coords[,2]
, ini=c(11.6,1.15), nug=1.45); ml3.spherical
ml4.spherical <- likfit(dG, cov.model="spherical", lam=lambda,
trend=~coords[,1]+I(coords[,1]^2), ini=c(5,2), nug=2); ml4.spherical

grau.ml2.ml1.spherical <- ml2.spherical$npars - ml1.spherical$npars;
grau.ml2.ml1.spherical            # Graus de liberdade de qui-quadrado
val.tab.spherical <- qchisq(0.95,grau.ml2.ml1.spherical); val.tab.spherical
Log.ml2.ml1.spherical <- 2*(logLik(ml2.spherical) - logLik(ml1.spherical));
Log.ml2.ml1.spherical

grau.ml3.ml1.spherical <- ml3.spherical$npars - ml1.spherical$npars;
grau.ml3.ml1.spherical # Grau de liberdade
val.tab1.spherical <- qchisq(0.95,grau.ml3.ml1.spherical); val.tab1.spherical
Log.ml3.ml1.spherical <- 2*(logLik(ml3.spherical) - logLik(ml1.spherical));
Log.ml3.ml1.spherical

grau.ml4.ml1.spherical <- ml4.spherical$npars - ml1.spherical$npars;
grau.ml4.ml1.spherical    # Grau de liberdade
val.tab2.spherical <- qchisq(0.95,grau.ml4.ml1.spherical); val.tab2.spherical
Log.ml4.ml1.spherical <- 2*(logLik(ml4.spherical) - logLik(ml1.spherical));
Log.ml4.ml1.spherical

#Modelo wave
ml1.wave <- likfit(dG, cov.model="wave", lam=lambda, ini=c(11.6,1.15), nug=1.45)
ml2.wave <- likfit(dG, cov.model="wave", lam=lambda, trend=~coords[,1] ,
ini=c(11.6,1.15), nug=1.45); ml2.wave
ml3.wave <- likfit(dG, cov.model="wave", lam=lambda, trend=~coords[,2] ,
ini=c(11.6,1.15), nug=1.45); ml3.wave
ml4.wave <- likfit(dG, cov.model="wave", lam=lambda,
trend=~coords[,1]+I(coords[,1]^2), ini=c(5,2), nug=2); ml4.wave

grau.ml2.ml1.wave <- ml2.wave$npars - ml1.wave$npars; grau.ml2.ml1.wave   #
Graus de liberdade de qui-quadrado
val.tab.wave <- qchisq(0.95,grau.ml2.ml1.wave); val.tab.wave
```

```
Log.ml2.ml1.wave <- 2*(logLik(ml2.wave) - logLik(ml1.wave)); Log.ml2.ml1.wave

grau.ml3.ml1.wave <- ml3.wave$npars - ml1.wave$npars; grau.ml3.ml1.wave # Grau
de liberdade
val.tab1.wave <- qchisq(0.95,grau.ml3.ml1.wave); val.tab1.wave
Log.ml3.ml1.wave <- 2*(logLik(ml3.wave) - logLik(ml1.wave)); Log.ml3.ml1.wave

grau.ml4.ml1.wave <- ml4.wave$npars - ml1.wave$npars; grau.ml4.ml1.wave    #
Grau de liberdade
val.tab2.wave <- qchisq(0.95,grau.ml4.ml1.wave); val.tab2.wave
Log.ml4.ml1.wave <- 2*(logLik(ml4.wave) - logLik(ml1.wave)); Log.ml4.ml1.wave

#.............................              ..............................
#Modelos escolhidos
ml4.matern.k2#Kappa=1.0
ml4.gaussian
ml4.exponential
ml4.spherical
ml4.wave
round(data.frame(AIC(ml4.matern.k2),AIC(ml4.gaussian), AIC(ml4.exponential),
                 AIC(ml4.spherical),AIC(ml4.wave)),2)
```

Modelo escolhido: Esférico

```
#Modelo Esférico
Modelo = ml4.spherical
#A seguir o indice de dependencia espacial
IDE = (Modelo$tausq)/(Modelo$tausq+Modelo$sigmasq)

## visualizando o grid onde serão feitas as predições
gr <- pred_grid(dG$borders,by=.035)
points(dG,lam=lambda)
points(gr,cex=0.4,pch=19)
## fazendo a prediÃ§Ã£o espacial (krigagem)kc
kc<-krige.control(obj.model=Modelo)
kr<-krige.conv(dG,loc=gr,borders=dG$borders, krige=kc)
names(kr)
## visualizando os valores preditos
pdf('preditos.pdf', width=7)
my.colors <- colorRampPalette(c("white","blue", "dark Blue"))
image(kr, axes=T, frame=T,xlab = "Longitude", ylab = "Latitude",
col=my.colors(100),
plot.title = title(cex.main = 4))
contour(kr, add=T, filled=F, nlev=18, col="black")
legend.krige(col=my.colors(100),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Precipitação (mm)", font = 16),
val=kr$predict, off=1.5, cex=1, scale.vals = seq(50,200,20), vert=F)
dev.off()

## visualizando os erros padrÃ£o de prediÃ§Ã£o (mapa de incerteza de prediÃ§Ã£o)
pdf('erros.pdf')
image(kr, val=sqrt(kr$krige.var), axes=T, frame=T,xlab = "Longitude", ylab =
"Latitude",
col=my.colors(100))
legend.krige(col=my.colors(100),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Erro padrão", font = 16),
val=sqrt(kr$krige.var), off=1.5, cex=1, scale.vals = seq(10,30,4), vert=F)
```

```
dev.off()

image(kr,col=terrain.colors(21))
points(dG, add=T)
contour(kr, add=T, nlev=21)

persp(kr)
persp(kr, theta=20, phi=35)

##
## exportando prediÃ§Ãµes
##
## exportando somente um vetor dos atributos (formato texto)
write(kr$pred, file="kr1.txt", ncol=1)
file.show("kr1.txt")    ## tecle c para terminar a exibiÃ§Ã£o do arquivo

## exportando somente os atributos como matriz (formato texto)
write(kr$pred, ncol=51, file="kr2.txt")

## exportando atributos e locaÃ§Ãµes de prediÃ§Ã£o
names(kr)


## simulacoes condicionais
args(krige.conv)
args(output.control)

kr.ml4 <- krige.conv(dG, loc=gr,
                     krige=krige.control(obj.m=Modelo),
                     output = output.control(n.pred=100))
names(kr.ml4)
dim(kr.ml4$simulations)

## a krigagem (pos simulacao) correponde a
predSim <- apply(kr.ml4$simulations, 1, mean)
plot(kr.ml4$pred, predSim)
abline(0,1,col="blue",pch=20)
summary(kr.ml4$pred - predSim)


## mapa de prob de estar abaixo de 100
p.90 <- apply(kr.ml4$simulations, 1, function(x) mean(x<100))
pdf('prob_clas.pdf')
my.colors1 <- colorRampPalette(c("blue", "white", "yellow", "orange", "red"))
image(kr.ml4, val=p.90, xlab="Longitude", ylab="Latitude", col=my.colors1(1000))
legend.krige(col=my.colors1(150),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Probabilidade", font = 16),
val=p.90, off=1.5, cex=1, scale.vals = seq(0,9,0.1), vert=F)
dev.off()


## suponha agora que definimos como regiao de alto risco
## as que possuiram prob > 0.7 de estar abaixo de 50
p50.alto <- ifelse(p50 < 0.7, 0, 1)
image(kr.ml4, val=p50.alto)#col=c(2,1)

image(kr.ml4, val=p50.alto,col=my.colors(100))
contour(kr.ml4, val=p50.alto, filled=T, col=my.colors(20),
        plot.title =title(xlab = "Longitude", ylab = "Latitude"),
        key.axes = axis(4
         ,seq(0, 1, by = 0.10)),
```

```
         key.title = title(main="Probab."))


## agora definindo 5 niveis de risco em fc da prob
image(kr.ml4, val=p50, breaks=seq(0, 1, by=0.2),
      col=c("blue","green","yellow","red","black"))


bor <- locator(type="l")
polygon(bor, col=4)


## completar aqui a krigagem dentrop do polygono

plot(dG)
v <-variog(dG, lam=lambda, max.dist=d.max,trend=~coords[,1]+I(coords[,1]^2))
plot(v, ylim=c(0,5))

args(variog)
v0 <- variog(dG, lam=lambda, max.dist=d.max, direction=0)
v45 <- variog(dG, lam=lambda, max.dist=d.max, direction=pi/4)
v90 <- variog(dG, lam=lambda, max.dist=d.max, direction=pi/2)
v135 <- variog(dG, lam=lambda, max.dist=d.max, direction=3*pi/4)

lines(v0)
lines(v45, col=2)
lines(v90, col=3)
lines(v135,col=4)

v4 <- variog4(dG, lam=lambda, max.dist=d.max)
plot(v4)
plot(v4, omn=T)
```

Análise Bayesiana

```
gr.bayes <- pred_grid(dG$borders,by=.030)

## Inf. Bayesiana
## Inf. Bayesiana
MC <- model.control(cov.model = "spherical", trend.d =
~coords[,1]+I(coords[,1]^2),
trend.l = ~coords[,1]+I(coords[,1]), lambda=1)

OC <- output.control(n.posterior=10000)

PCphi <- prior.control(beta="normal", sigmasq.pr="sc.inv.chisq", sigmasq = 2,
df.sigmasq = 4,
                       phi.prior="rec", tausq.rel.prior ="rec",
                       tausq.rel.discrete = seq(1, 3, by=0.2), phi.disc=seq(0,
3, by=0.2))

s100.kb <- krige.bayes(dG, loc=gr.bayes, model=MC, prior=PCphi, out = OC)

names(s100.kb$posterior)
amostra = (s100.kb$posterior$sample)
beta0 = amostra$beta0; beta1 = amostra$beta1; beta2 = amostra$beta2;
sigmasq = amostra$sigmasq; phi = amostra$phi; tausq =
(amostra$tausq.rel)*sigmasq
#Estudo de convergencia
nburn = 1001:10000 #"Esquecendo" as 100 primeiras iterações
amostra.par = s100.kb$posterior$sample
 library(coda)
 parametros = mcmc(cbind(beta0, beta1, beta2, sigmasq, phi, tausq)[nburn,])
 colnames(parametros) = names(amostra.par)
 summary(parametros)

 #salvando em tex
```

```
 aa= as.matrix(summary(parametros))
 estatisticas = data.frame(aa[[1]][,1],aa[[2]][,c(1,3,5)])
 tab = xtable(estatisticas, caption =
"Resumo da distribuição a posteriori dos parâmetros do modelo \ref{modelo}",
label = "estatistica", digits = 2)
print(tab)

#Gerando Gráficos
range = 1001:10000 #"Esquecendo" as 1000 primeiras iterações
#Para os betas
pdf('trace1.pdf')
 par(mfrow = c(3, 1))
 ts.plot(beta0[range], type = "l", main = expression(beta[0]), xlab = "Amostra",
col="blue")
 ts.plot(beta1[range], type = "l", main = expression(beta[1]), xlab = "Amostra",
col="green")
 ts.plot(beta2[range], type = "l", main = expression(beta[2]), xlab = "Amostra",
col="red")
dev.off()
#Para os parametros S(x)
pdf('trace2.pdf')
 par(mfrow = c(3, 1))
 ts.plot(sigmasq[range],type="l", main = expression(sigma^2), xlab = "Amostra",
col="77")
 ts.plot(phi[range], type="l",main = expression(phi),col="hotpink",
xlab="Amostra")
 ts.plot(tausq[range], type = "l", main = expression(tau^2), xlab = "Amostra",
col="orange")
dev.off()

pdf('cor1.pdf')
par(mfrow = c(3, 1))
 acf(beta0[range], main = expression(beta[0]),col="blue")
 acf(beta1[range], main = expression(beta[1]),col="green")
 acf(beta2[range], main = expression(beta[2]),col="red")
dev.off()

pdf('cor2.pdf')
 par(mfrow = c(3, 1))
 acf(sigmasq[range], main = expression(sigma^2),col="77")
 acf(phi[range], main = expression(phi),col="hotpink")
 acf(tausq[range], main = expression(tau^2),col="orange")
dev.off()

## visualizando as posterioris (marginais)
## beta|y
hist(s100.kb$posterior$sample[,1], prob=T)
density(s100.kb$posterior$sample[,1])
lines(density(s100.kb$posterior$sample[,1]))
rug(s100.kb$posterior$sample[,1])
## sigmasq|y
hist(s100.kb.phi$posterior$sample[,2], prob=T, main=expression(sigma^2))
lines(density(s100.kb.phi$posterior$sample[,2]))
rug(s100.kb.phi$posterior$sample[,2])
## phi|y
barplot(table(s100.kb.phi$posterior$sample[,3]))

#Mapa da predição
pdf('pred_bayes.pdf')
my.colors1 <- colorRampPalette(c("white", "blue", "dark Blue"))
```

```
image(s100.kb, xlab="Longitude", ylab="Latitude", val=s100.kb$predictive$mean,
col=my.colors1(100))
legend.krige(col=my.colors1(100),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Predição", font = 16),
val=s100.kb$predictive$mean, off=1.5, cex=1, scale.vals = seq(30,300,(300-
30)/10), vert=F)
dev.off()


#Mapa da variabilidade da predição
pdf('var_bayes.pdf')
my.colors1 <- colorRampPalette(c("white", "red"))
image(s100.kb, xlab="Longitude", ylab="Latitude",
val=s100.kb$predictive$variance, col=my.colors1(100))
legend.krige(col=my.colors1(100),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Variabilidade", font = 16),
val=s100.kb$predictive$variance, off=1.5, cex=1, scale.vals =
seq(0,1700,1700/10), vert=F)
dev.off()


## mapa de um funcional: probabilidade de estar abaixo de 100 mm
## calculando as probabilidades
p100.0 <- pnorm(100.0, s100.kb$predictive$mean,
sqrt(s100.kb$predictive$variance), lower.tail=TRUE)
## e colocando no mapa
pdf('prob_bayes.pdf')
my.colors1 <- colorRampPalette(c("blue", "white", "yellow","orange","red"))
image(s100.kb, xlab="Longitude", ylab="Latitude", val=p100.0,
col=my.colors1(100))
legend.krige(col=my.colors1(100),x.leg=c(-38.7,-35), y.leg=c(-5.8,-5.6),
text(-36.9,-5.87,"Probabilidade", font = 16),
val=p100.0, off=1.5, cex=1, scale.vals = seq(0,1,0.1), vert=F)
dev.off()


names(s100.kb)

names(s100.kb$posterior)

summary(s100.kb$pred$uncertainty)


p50bayes <- apply(kb$pred$simul, 1, function(x) mean(x>50))
image(kb, values = p50bayes)

## comparando as predicoes
image(kc.ml4)
image(kb)

plot(kc.ml, kb$predictive$pred)
```