MATH 8090: Geostatistics

Whitney Huang, Clemson University

11/17-19/2025

Contents

Toy Examples

Create spatial locations

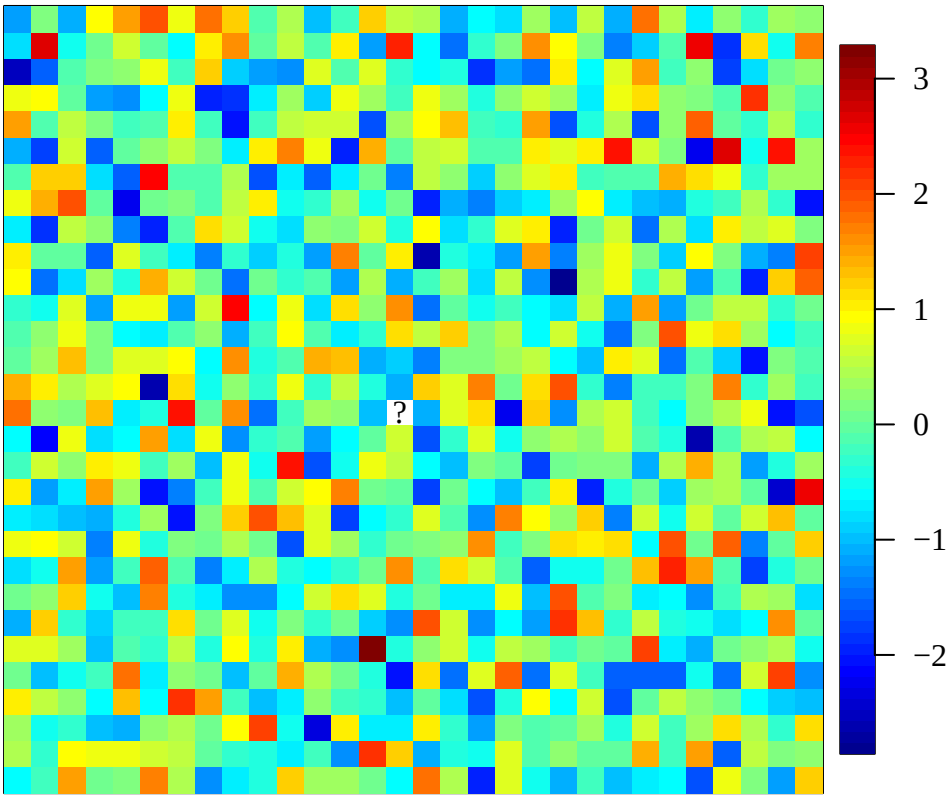
```
N = 30
xg <- yg <- seq(0, 1, length = N)
locs <- expand.grid(xg, yg)
```

Case 1: No spatial pattern

```
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
set.seed(123)
y1 <- array(rnorm(n = N^2), dim = c(N, N))
y1[N / 2, N / 2] <- NA
which(is.na(y1) == 1)
```

```
## [1] 435
```

```
library(fields)
image.plot(xg, yg, y1, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
text(xg[N / 2], yg[N / 2], "?")
```



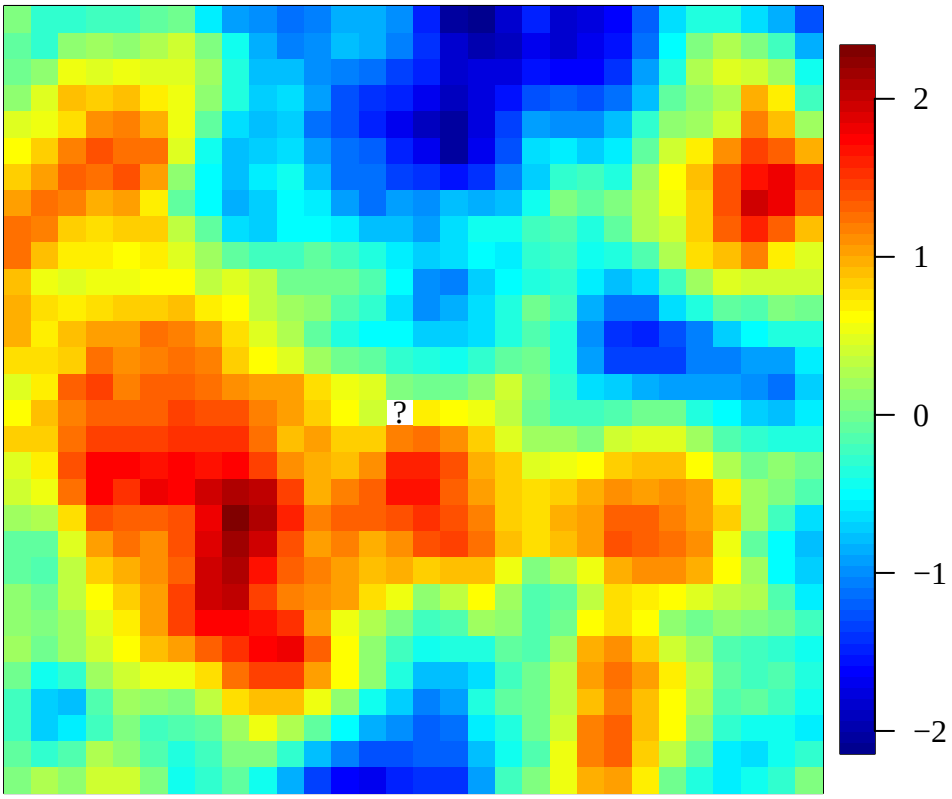
Case 2: A smooth spatial image

```
library(MASS)
cov.Matern <- function(h, pars){
  Matern(h, phi = pars[1], range = pars[2], smoothness = pars[3])
}
dist <- rdist(locs)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
```

```
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2[N / 2,  N / 2] <- NA
which(is.na(y2) == 1)
```

```
## [1] 435
```

```
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
image.plot(xg, yg, y2, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
text(xg[N / 2], yg[N / 2], "?")
```



Using Variogram Cloud and (Binned) Variogram to Examine the Spatial Dependence

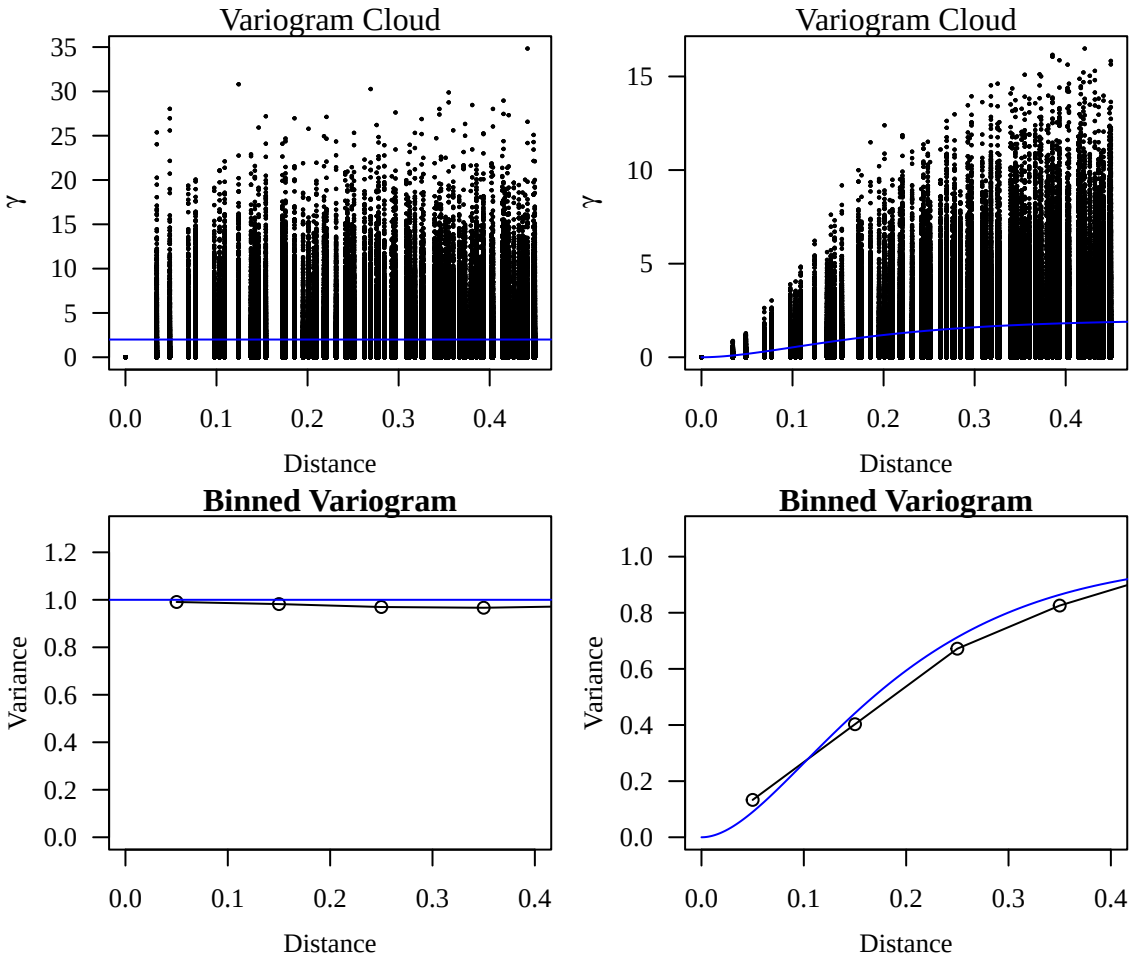
```
gamma1 <- array(dim = c(N^2, N^2))
system.time(for (i in 1:N^2){
  for (j in 1:N^2){
    gamma1[i, j] <- (c(y1)[i] - c(y1)[j])^2
  }
})
```

```
##    user  system elapsed
##   2.693   0.869   3.737
```

```r
system.time(gamma1 <- outer(c(y1), c(y1), FUN = "-")^2)
```

```
##    user  system elapsed
##   0.003   0.001   0.004
```

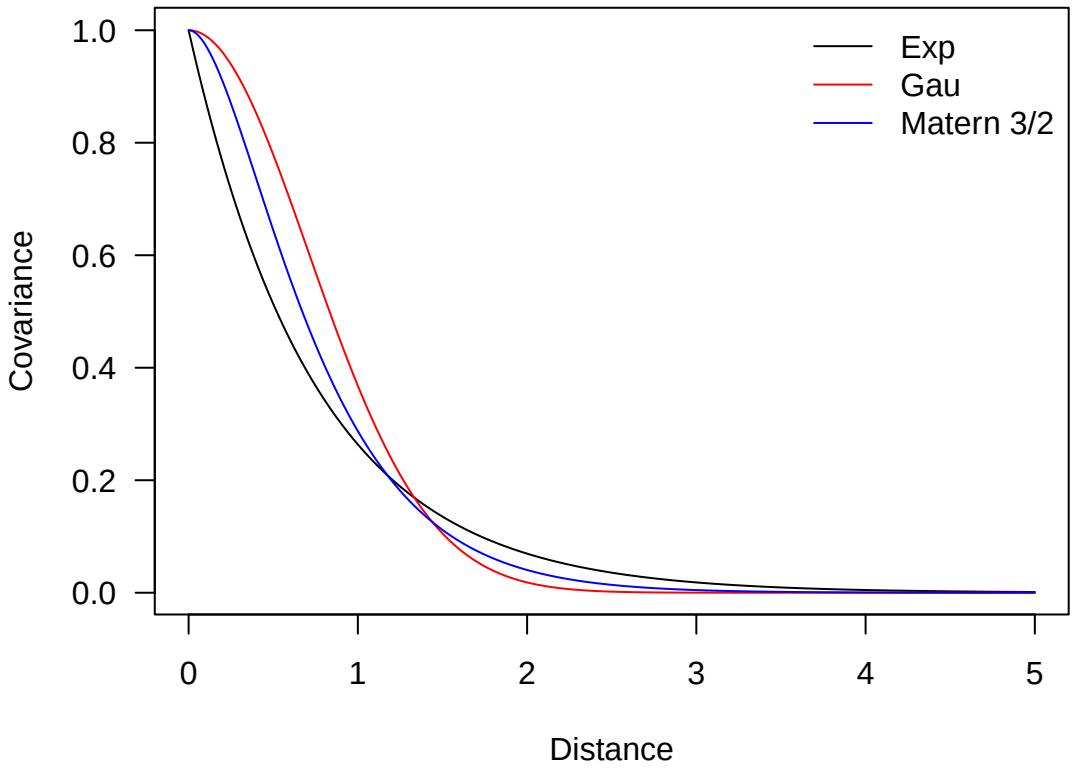
```r
gamma2 <- outer(c(y2), c(y2), FUN = "-")^2

dist_cut <- 0.45
good <- which(dist <= dist_cut)

par(mfrow = c(2, 2), mar = c(3.6, 3.6, 1, 0.6),
    mgp = c(2.4, 1, 0), las = 1, family = "serif")
plot(c(dist)[good], c(gamma1)[good],
     cex = 0.2, xlab = "Distance", ylab = expression(gamma))
abline(h = 2 * 1, col = "blue")
mtext("Variogram Cloud")

plot(c(dist)[good], c(gamma2)[good], cex = 0.2, xlab = "Distance",
     ylab = expression(gamma))
dgrid <- seq(0, sqrt(2), 0.001)
lines(dgrid, 2 * (1 - cov.Matern(dgrid, c(1, 0.1, 1.5))),
      col = "blue")
mtext("Variogram Cloud")

vargram1 <- vgram(locs, c(y1), N = 30)
plot(vargram1, ylim = c(0, 1.3), xlim = c(0, 0.4), main = "Binned Variogram")
abline(h = 1, col = "blue")

vargram2 <- vgram(locs, c(y2), N = 30)
plot(vargram2, ylim = c(0, 1.1), xlim = c(0, 0.4), main = "Binned Variogram")
lines(dgrid, 1 - cov.Matern(dgrid, c(1, 0.1, 1.5)), col = "blue")
```



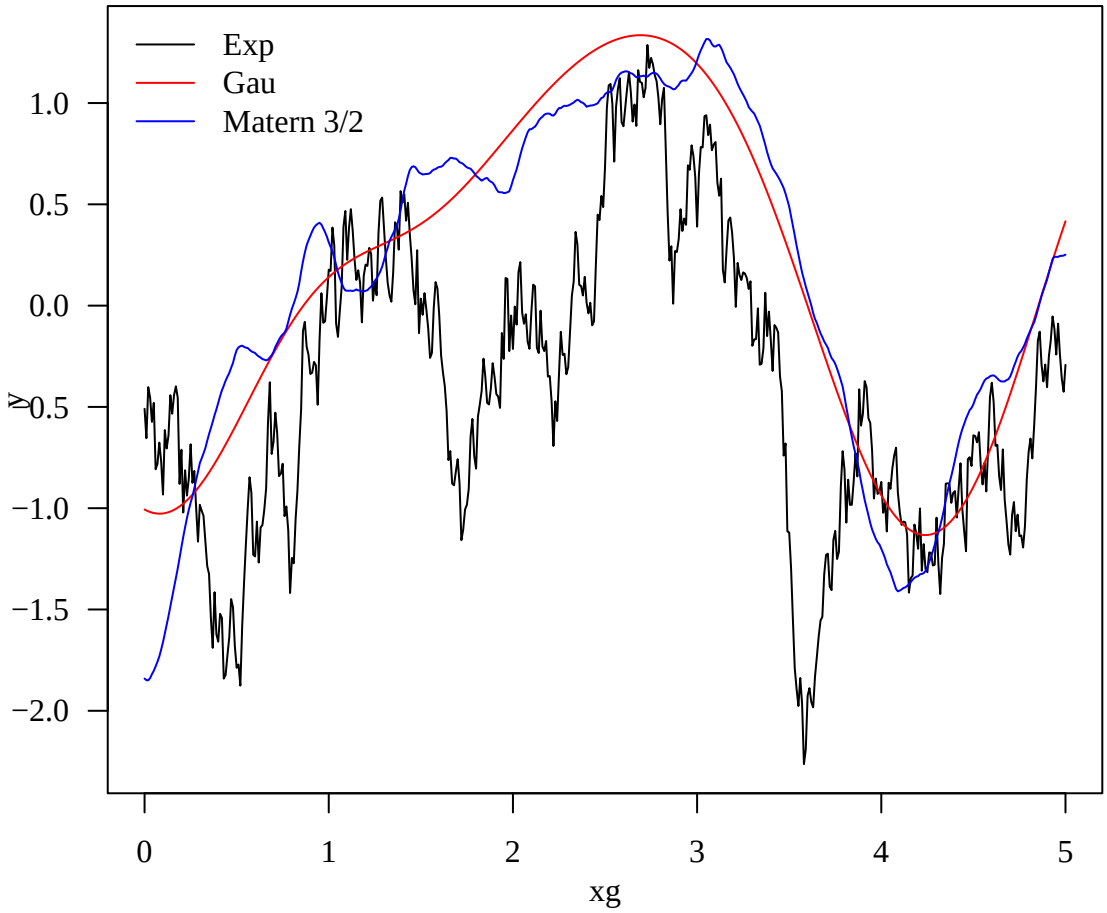
Gaussian Processes: Covariance functions and their realzations

Simulate 1D realzations

```
# Commonly used covariance functions
cov.exp <- function(h, pars) pars[1] * exp(-h / pars[2])
cov.doubleExp <- function(h, pars) pars[1] * exp(-(h / pars[2])^2)

xg <- seq(0, 5, 0.01)
c_exp <- cov.exp(xg, c(1, 0.75))
c_doubleExp <- cov.doubleExp(xg, c(1, 1))
c_Matern <- cov.Matern(xg, c(1, 0.4, 1.5))

plot(xg, c_exp, type = "l", ylab = "Covariance", xlab = "Distance", las = 1)
lines(xg, c_doubleExp, col = "red")
lines(xg, c_Matern, col = "blue")
legend("topright", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```




```
Sigma_exp <- cov.exp(rdist(xg), c(1, 0.75))
Sigma_doubleExp <- cov.doubleExp(rdist(xg), c(1, 1))
Sigma_Matern <- cov.Matern(rdist(xg), c(1, 0.4, 1.5))
library(MASS)
set.seed(123)
sim_exp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_exp)
set.seed(123)
sim_doubleExp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_doubleExp)
set.seed(123)
sim_Matern_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_Matern)

par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
plot(xg, sim_exp_1d, type = "l", ylim = range(sim_exp_1d, sim_doubleExp_1d,
                                              sim_Matern_1d),
     ylab = "y", las = 1)
lines(xg, sim_doubleExp_1d, col = "red")
lines(xg, sim_Matern_1d, col = "blue")
legend("topleft", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```

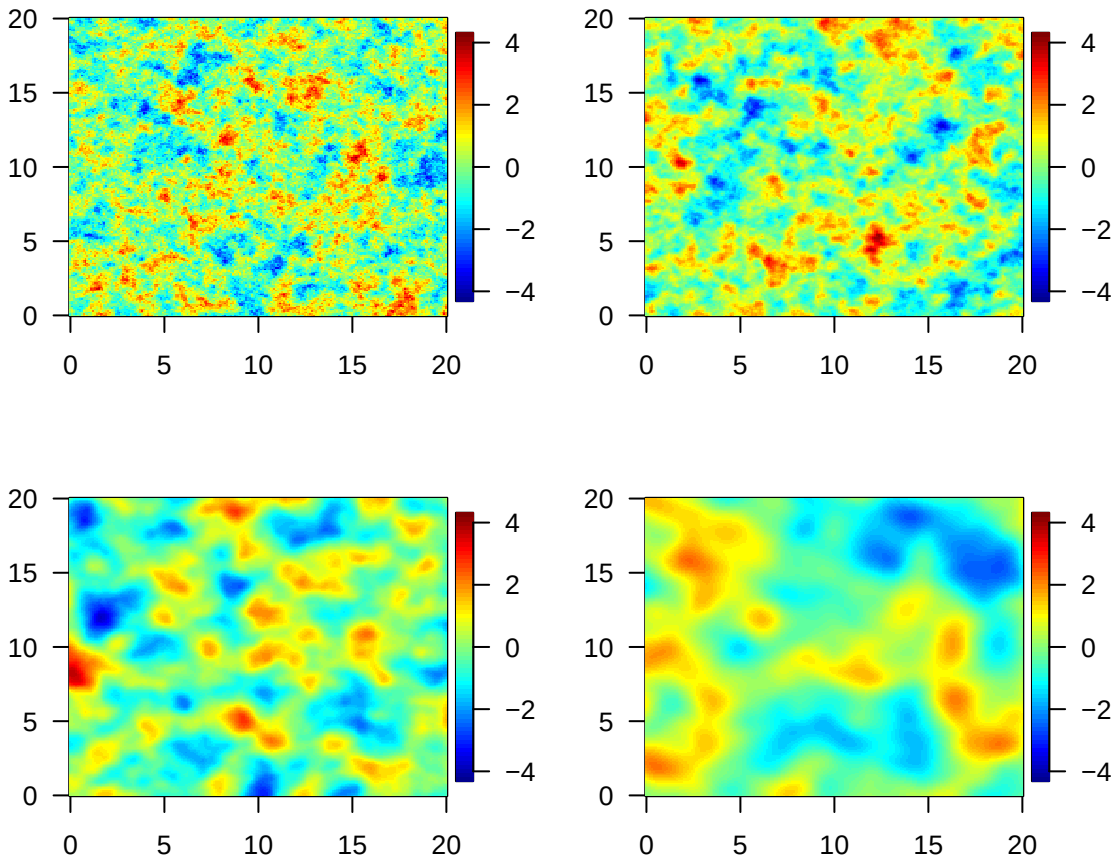


Simulate 2D realzations

```
grid <- list(x = seq(0, 20, len = 200), y = seq(0, 20, len = 200))
nu <- c(0.5, 1, 2.5, 5)
obj <- list()
for (i in 1:4) obj[[i]] <- matern.image.cov(grid = grid, theta = 0.5,
                                            smoothness = nu[i], setup = TRUE)
set.seed(2021)
sim <- lapply(obj, sim.rf)
set.panel(2, 2)
```

```
## plot window will lay out plots in a 2 by 2 matrix
```

```
par(mar = c(2.6, 3.1, 3.1, 0.6), las = 1)
for (i in 1:4){
  image.plot(grid$x, grid$y, sim[[i]], zlim = c(-4.25, 4.25), xlab = "", ylab = "")
}
```



Spatial interpolation

Here we assume $m(s) = 0 \quad \forall s \in \mathcal{S}$

Predicting one location

$$\hat{y}_0 = k^T \Sigma^{-1} \boldsymbol{y}$$
$$\text{Var}(\hat{y}_0) = \sigma^2 - k^T \Sigma^{-1} k$$

```
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2[N / 2,  N / 2] <- NA
# k vector
k <- Sigma_Matern[435, -435]
# Sigma matrix
Sigma <- Sigma_Matern[-435, -435]
# y vector
y <- y2[-435]
## prediction
system.time(y0_hat <- t(k) %*% solve(Sigma) %*% y)
```

```
##    user  system elapsed
##   0.440   0.011   0.550
```

```
system.time(y0_hat_faster <-  t(k) %*% solve(Sigma, y))
```

```
##    user  system elapsed
##   0.104   0.009   0.155
```

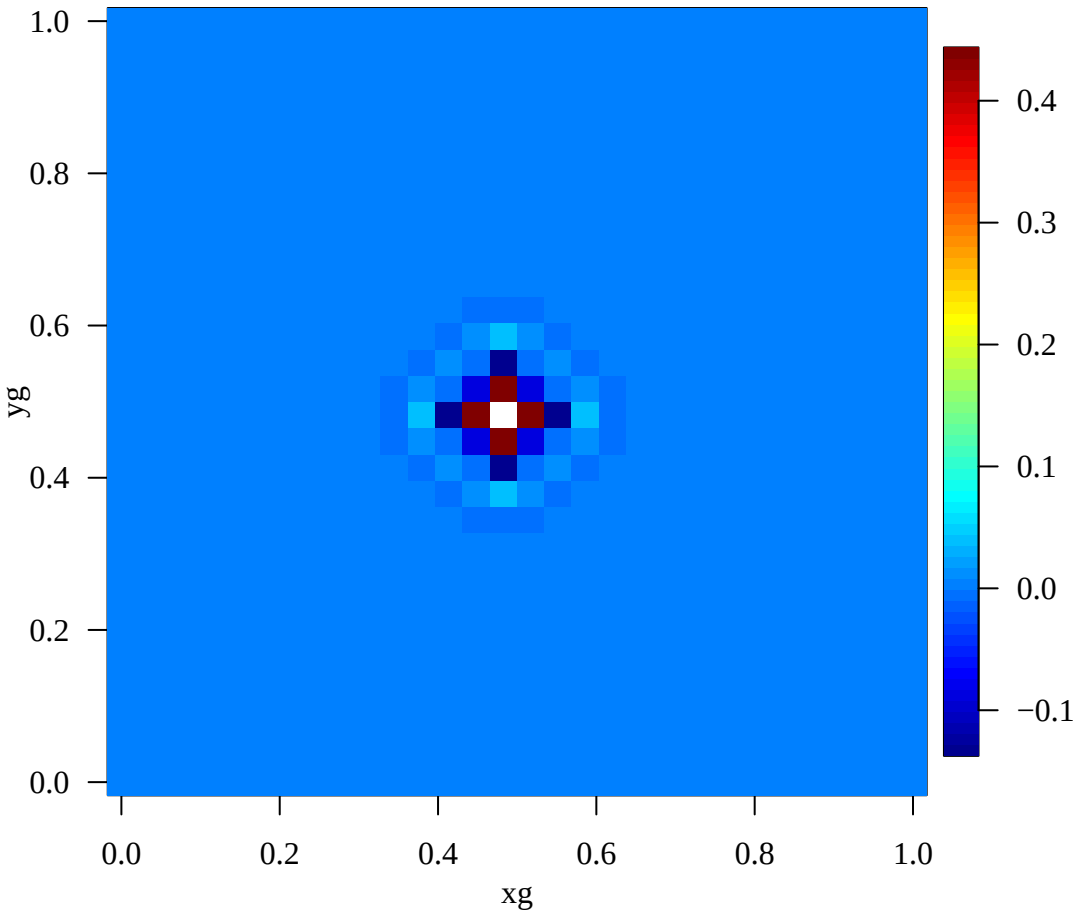
```
## prediction uncertainty
system.time(var_y0_hat <- Sigma_Matern[435, 435] - t(k) %*% solve(Sigma) %*% k)
```

```
##    user  system elapsed
##   0.435   0.016   0.579
```

```
system.time(Sigma_Matern[435, 435] - t(k) %*% solve(Sigma, k))
```

```
##    user  system elapsed
##   0.098   0.002   0.109
```

```
w <- t(k) %*% solve(Sigma)
weight_map <- array(c(w[1, 1:434], NA, w[1, 435:899]),
                    dim = c(30, 30))
xg <- yg <- seq(0, 1, length = N)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
image.plot(xg, yg, weight_map)
```



Predicting multiple locations

```
N = 30
xg <- yg <- seq(0, 1, length = N)
locs <- expand.grid(xg, yg); dist <- rdist(locs)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2_vec <- c(y2)
set.seed(123)
rm <- sample(1:(N^2), 0.75 * N^2)
y2_vec[rm] <- NA
y2_rm <- array(y2_vec, dim = c(N, N))

# k matrix
k <- Sigma_Matern[-rm, rm]
# Sigma matrix
Sigma <- Sigma_Matern[-rm, -rm]
# y vector
y <- y2_rm[-rm]
## prediction
system.time(y0_hat <- t(k) %*% solve(Sigma) %*% y)
```

```
##    user  system elapsed
##   0.020   0.000   0.022
```

```
system.time(y0_hat_faster <-  t(k) %*% solve(Sigma, y))
```

```
##    user  system elapsed
##   0.003   0.000   0.003
```

```
## prediction uncertainty
system.time(var_y0_hat <- Sigma_Matern[rm, rm] - t(k) %*% solve(Sigma) %*% k)
```

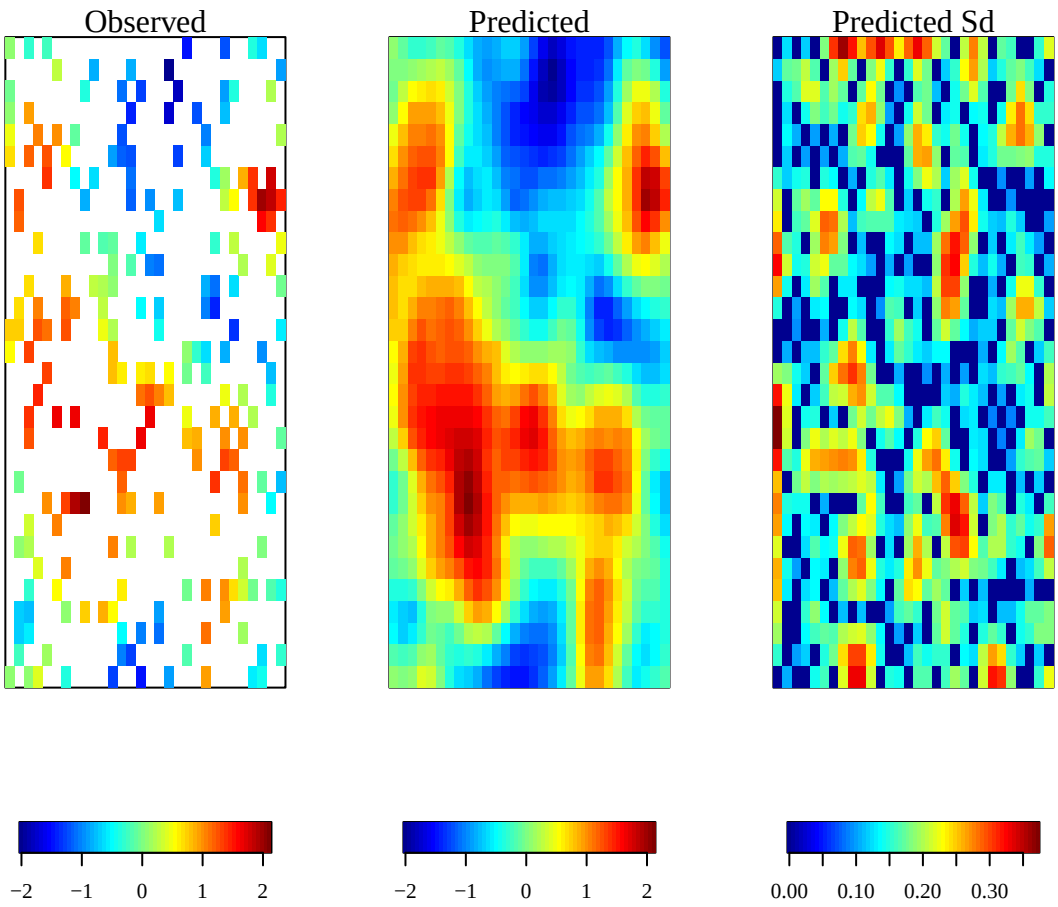
```
##    user  system elapsed
##   0.055   0.001   0.057
```

```
system.time(Sigma_Matern[rm, rm] - t(k) %*% solve(Sigma, k))
```

```
##    user  system elapsed
##   0.055   0.001   0.057
```

```
par(mfrow = c(1, 3), las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
image.plot(xg, yg, y2_rm, xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line= 1)
mtext("Observed")

y2_pred <- y2_rm
y2_pred[rm] <- y0_hat[, 1]
```

```r
image.plot(xg, yg, y2_pred, xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line= 1)
mtext("Predicted")

con_sd <- sqrt(diag(var_y0_hat))
condSD <- array(0, dim = c(N, N))
temp <- c(condSD); temp[rm] <- con_sd
condSD <- array(temp, dim = c(N, N))

image.plot(xg, yg, matrix(condSD, N, N), xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line = 1)
mtext("Predicted Sd")
```



Rainfall Data from Parana State, Brazil

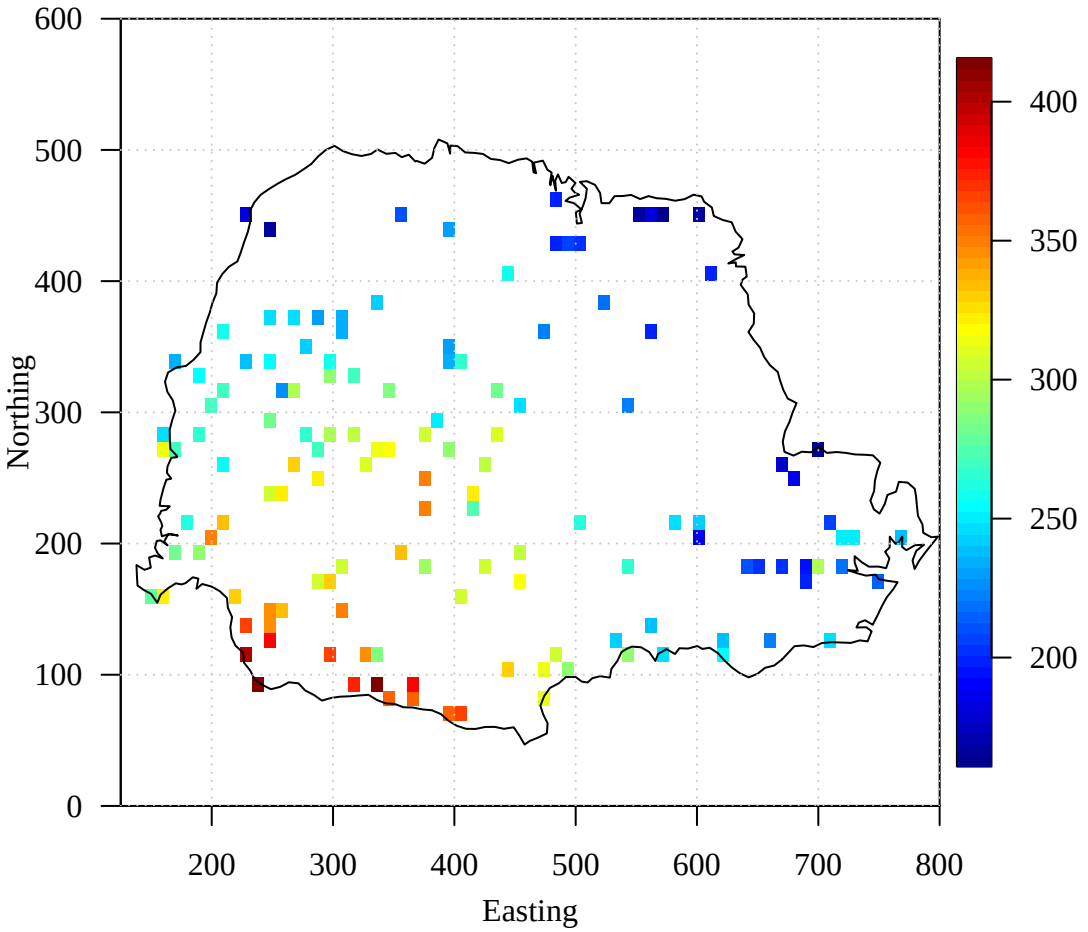
In this R lab, we analyze dry-season (May-June) rainfall data collected from 143 weather stations in Parana State, Brazil, to illustrate spatial interpolation.

Loading and summarizing the data

```r
library(geoR)
data(parana)
summary(parana)
```

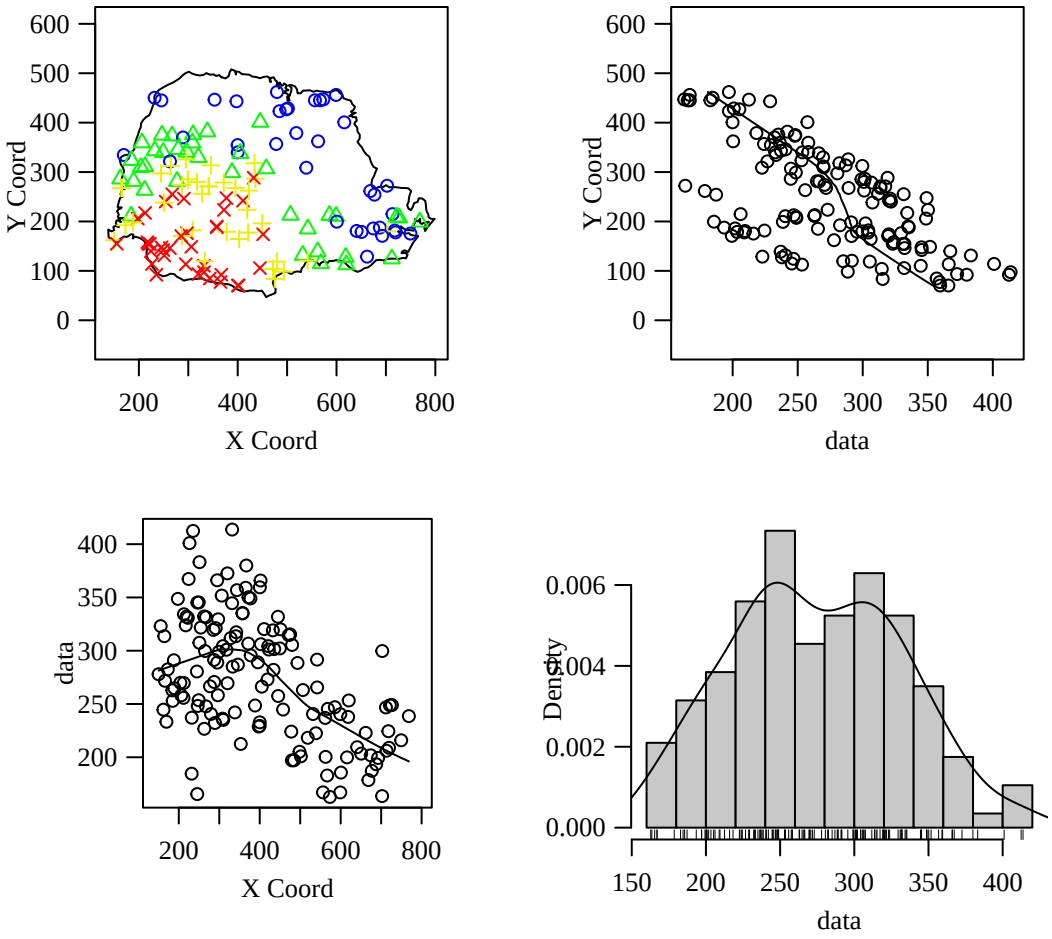
```
## Number of data points: 143
##
## Coordinates summary
##         east    north
## min 150.1220  70.3600
## max 768.5087 461.9681
##
## Distance summary
##      min      max
##   1.0000 619.4925
##
## Borders summary
##         east    north
## min 137.9873  46.7695
## max 798.6256 507.9295
##
## Data summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 162.7700 234.1900 269.9200 274.4106 318.2300 413.7000
##
## Other elements in the geodata object
## [1] "loci.paper"
```

```r
library(fields)
par(las = 1, mgp = c(2.2, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
quilt.plot(parana$coords, parana$data, ny = 36, ylim = c(0, 600),
           xlim = c(125, 800), xlab = "Easting", ylab = "Northing")
lines(parana$borders)
grid()
```

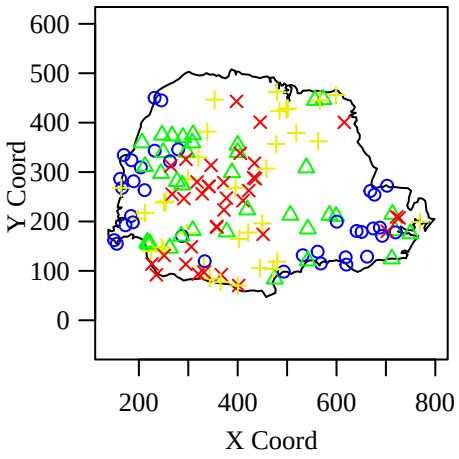


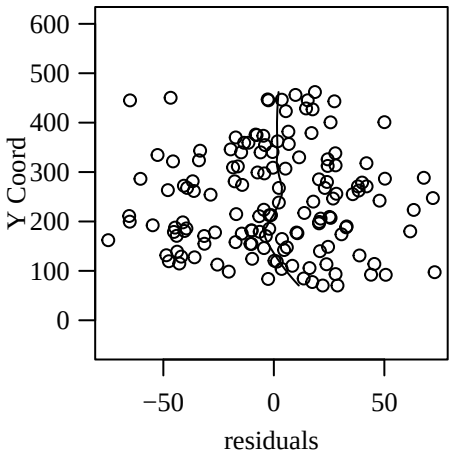
Below, we examine the potential spatial trend that characterizes large-scale spatial variation. The non-parametric LOWESS fits along the x-axis (longitude) and y-axis (latitude) suggest approximately linear trends in both directions, supporting the reasonableness of this assumption.

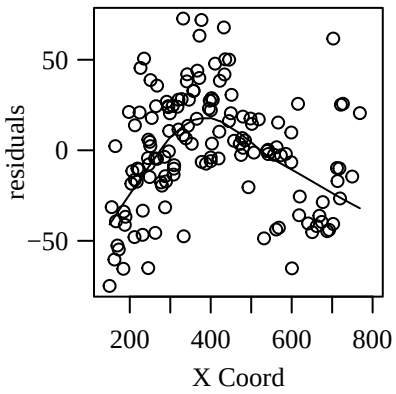
```
par(las = 1, mgp = c(2.5, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
plot(parana, lowess = TRUE)
```

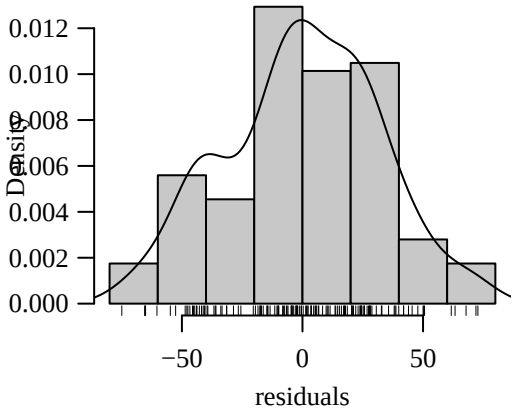



```
plot(parana, trend = "1st", lowess = TRUE)
```









Variogram Analysis

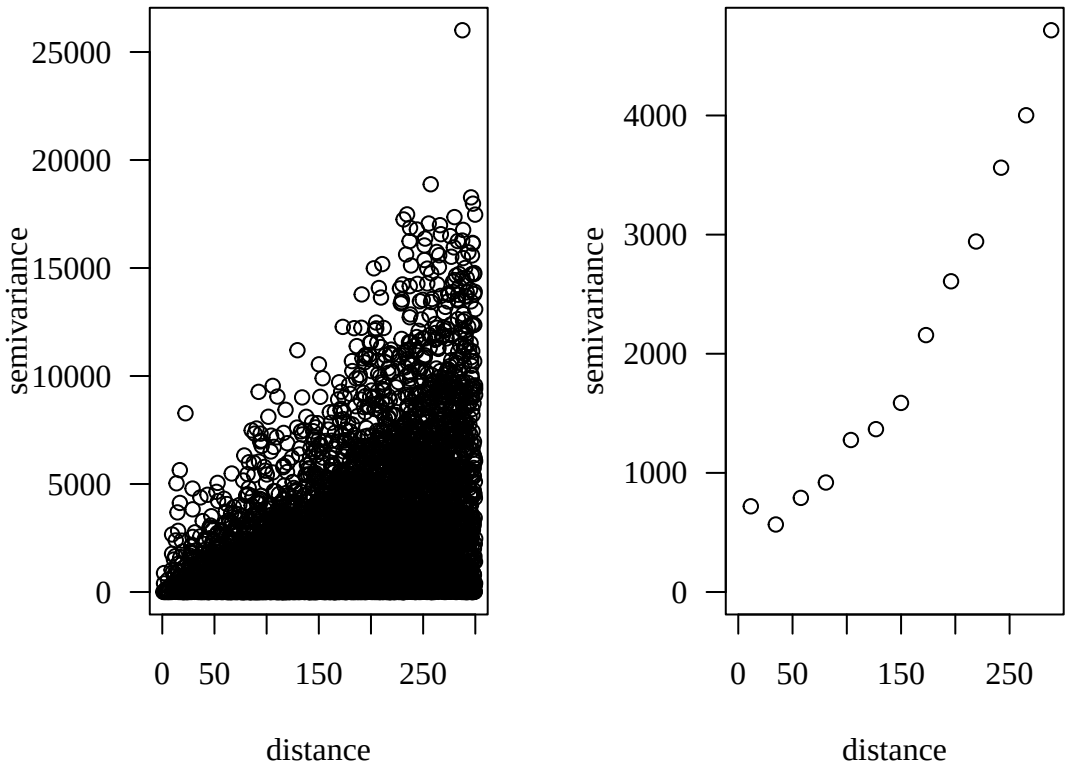
```
par(mfrow = c(1, 2), las = 1, mgp = c(3, 1, 0), family = "serif")
parana.vario <- variog(parana, max.dist = 300, cex = 0.5, option = "cloud")
```

```
## variog: computing omnidirectional variogram
```

```
plot(parana.vario)
parana.vario <- variog(parana, max.dist = 300, cex = 0.5)
```

```
## variog: computing omnidirectional variogram
```

```
plot(parana.vario)
```



Below, you will see the variogram with (right) and without (left) the spatial trend component removed.

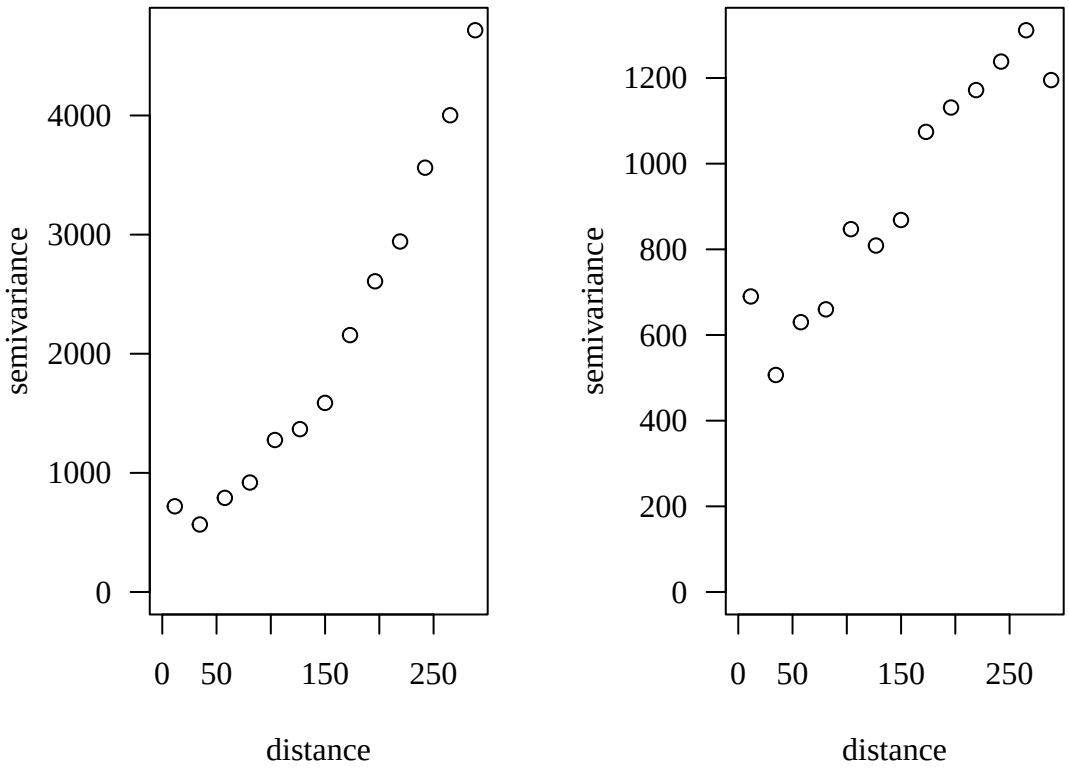
```
par(mfrow = c(1, 2), las = 1, mgp = c(3, 1, 0), family = "serif")
parana.vario <- variog(parana, max.dist = 300, cex = 0.5)
```

```
## variog: computing omnidirectional variogram
```

```
plot(parana.vario)
parana.variot <- variog(parana, trend = "1st",  max.dist = 300)
```

```
## variog: computing omnidirectional variogram
```

```
plot(parana.variot)
```



Generates a Monte Carlo envelope (confidence bands) for the variogram under the assumption of spatial independence. This is useful for assessing whether the observed spatial dependence is statistically significant. Plots the empirical variogram (parana.v) and overlays the Monte Carlo envelope (parana.v.env) to visually assess spatial structure and compare it against the null hypothesis of independence.

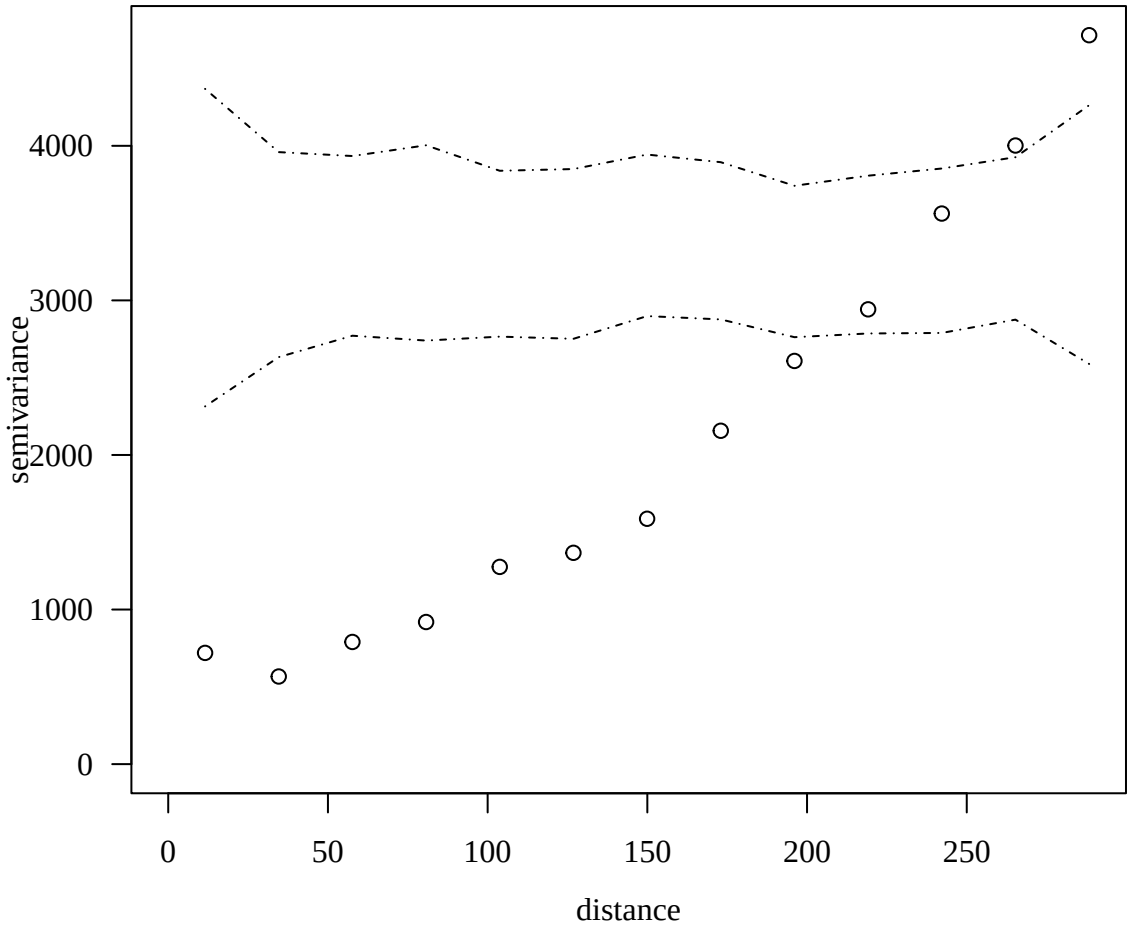
```
par(las = 1, mgp = c(2.5, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
parana.v <- variog(parana, max.dist = 300)
```

```
## variog: computing omnidirectional variogram
```

```
parana.v.env <- variog.mc.env(parana, obj.variog = parana.v)
```

```
## variog.env: generating 99 simulations by permutating data values
## variog.env: computing the empirical variogram for the 99 simulations
## variog.env: computing the envelops
```

```
plot(parana.v, env = parana.v.env)
```

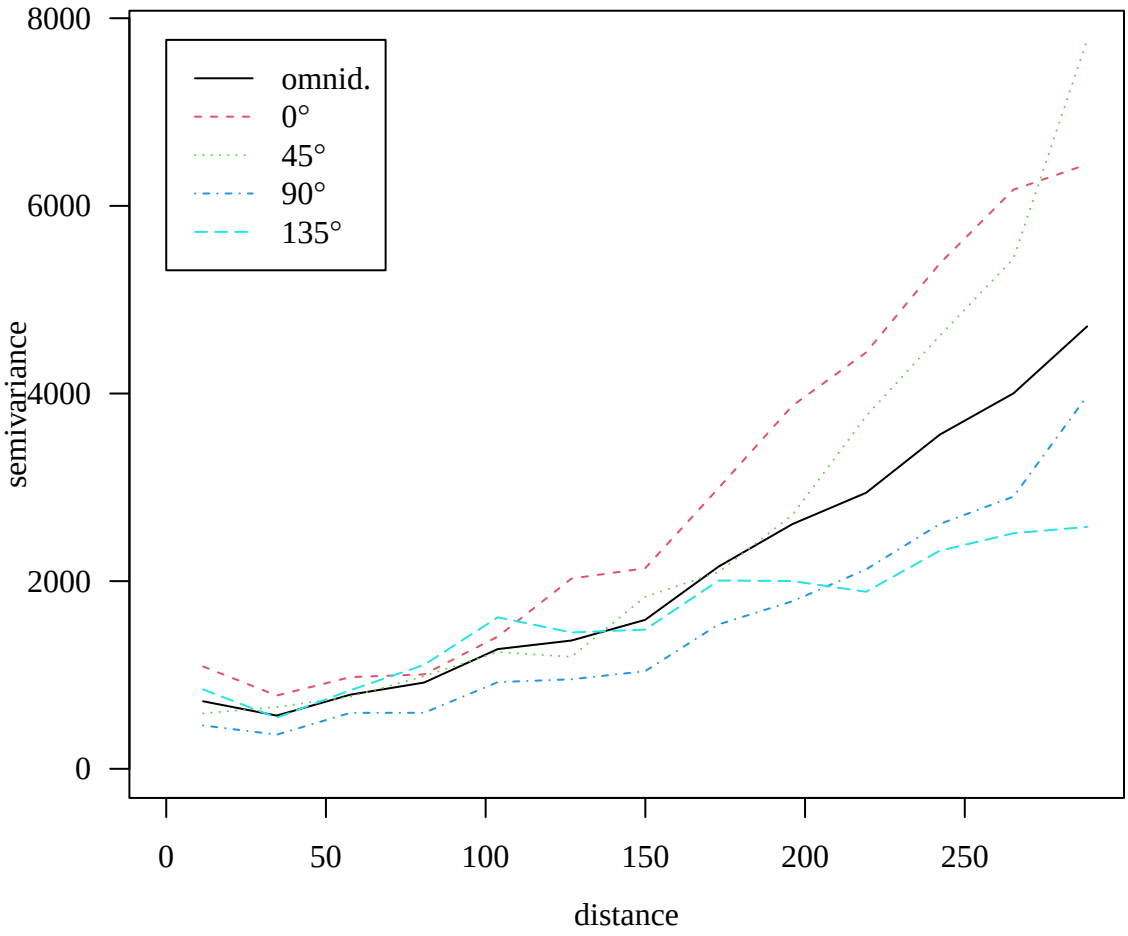


The code chunk below specifically assesses the isotropy assumption by plotting the variogram along different directions. The variograms should be similar if isotropy holds.

```
par(las = 1, mgp = c(2.5, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
parana.v4 <- variog4(parana, max.dist = 300)
```

```
## variog: computing variogram for direction = 0 degrees (0 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 45 degrees (0.785 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 90 degrees (1.571 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 135 degrees (2.356 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing omnidirectional variogram
```

```
plot(parana.v4, env = parana.v.env, omni = TRUE)
```



Parameter Estimation

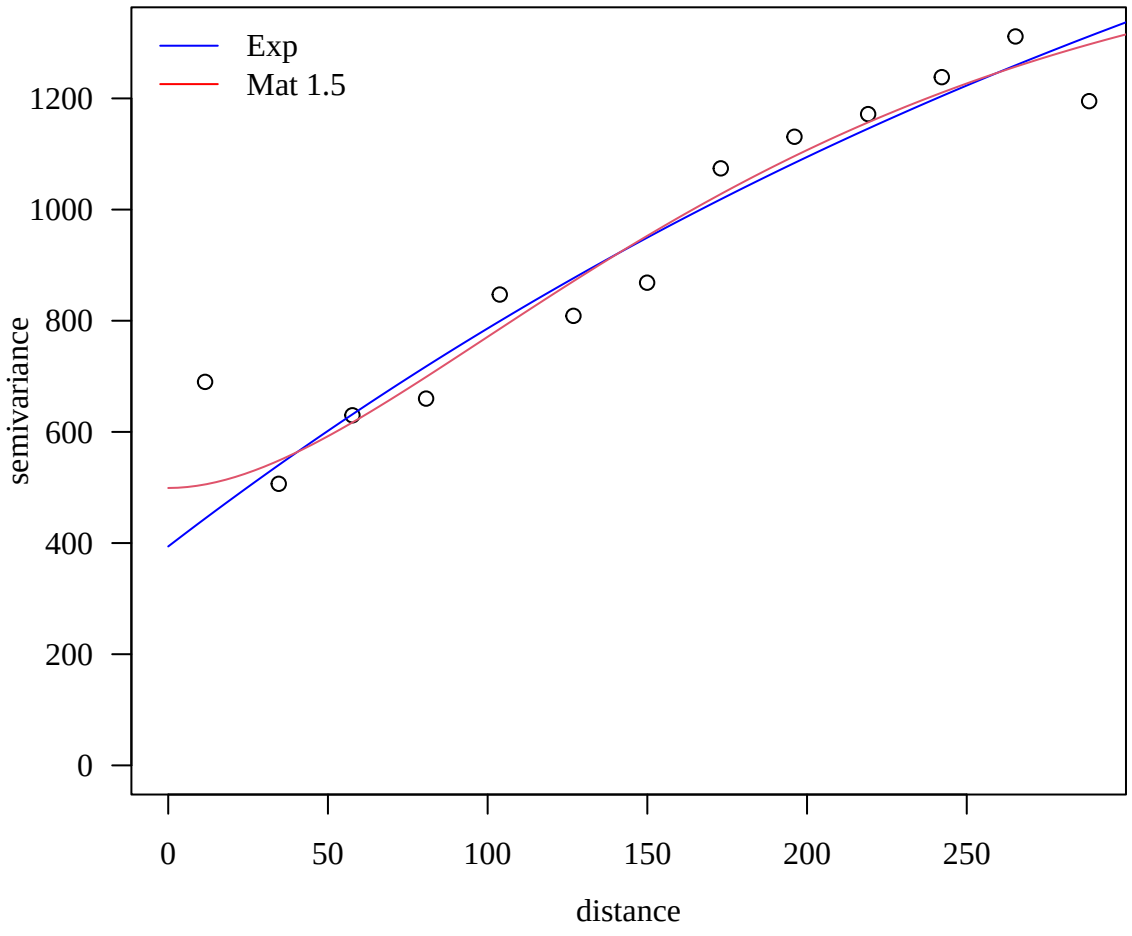
```
# with linear trend
parana.vtfit.exp <- variofit(parana.variot)
```

```
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(parana.variot): initial values not provided - running the
## default search

## variofit: searching for best initial value ... selected values:
##               sigmasq   phi      tausq    kappa
## initial.value "1311.47" "230.66" "327.87" "0.5"
## status        "est"     "est"    "est"    "fix"
## loss value: 33524269.3444706
```

```
parana.vtfit.mat1.5  <- variofit(parana.variot, kappa = 1.5)
```

```
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(parana.variot, kappa = 1.5): initial values not provided -
## running the default search

## variofit: searching for best initial value ... selected values:
##                sigmasq phi      tausq    kappa
## initial.value "983.6" "138.39" "655.73" "1.5"
## status        "est"   "est"    "est"    "fix"
## loss value: 43717205.8946468
```

```
par(las = 1, mgp = c(2.5, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
plot(parana.variot)
lines(parana.vtfit.exp, col = "blue"); lines(parana.vtfit.mat1.5, col = 2)
legend("topleft", legend = c("Exp", "Mat 1.5"), col = c("blue", "red"),
       lty = 1, bty = "n")
```



MLE

```
(parana.ml1 <- likfit(parana, trend = "1st", ini = c(1000, 50), nug = 100))
```

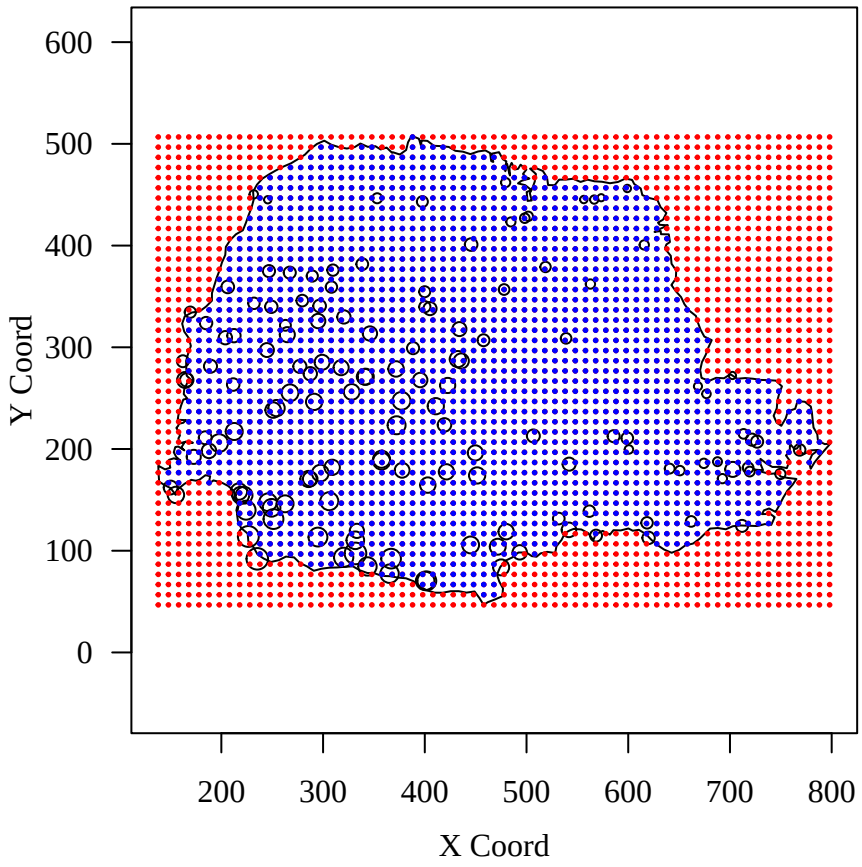
```
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.

## likfit: estimated model parameters:
##      beta0      beta1      beta2      tausq    sigmasq        phi
## "416.4984" " -0.1375" " -0.3997" "385.5180" "785.6904" "184.3863"
## Practical Range with cor=0.05 for asymptotic range: 552.3719
##
## likfit: maximised log-likelihood = -663.9
```

```
(parana.ml2 <- likfit(parana, trend = "2nd", ini = c(1000, 50), nug = 100))
```

```
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.

## likfit: estimated model parameters:
##      beta0      beta1      beta2      beta3      beta4      beta5      tausq
## "423.9282" "  0.0620" " -0.6360" " -0.0004" "  0.0000" "  0.0006" "381.2267"
##    sigmasq        phi
## "372.5993" " 77.5441"
## Practical Range with cor=0.05 for asymptotic range: 232.3013
##
## likfit: maximised log-likelihood = -660.2
```

Spatial Prediction

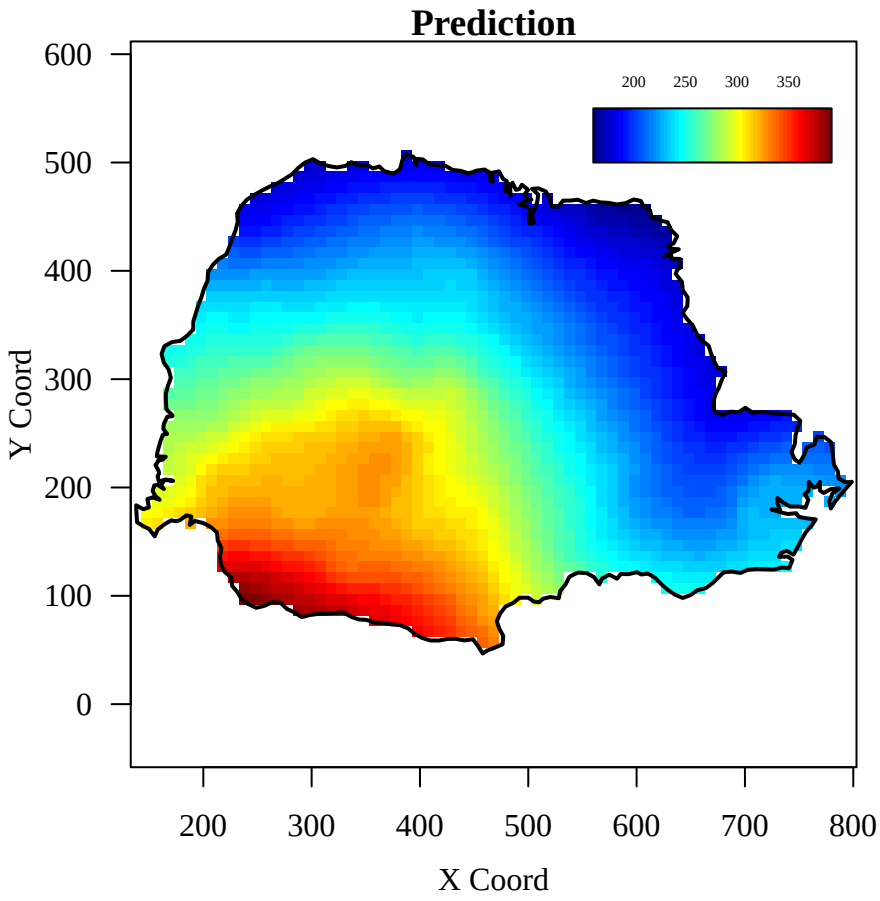
```
par(las = 1, mgp = c(2.4, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
parana.gr <- pred_grid(parana$borders, by = 10)
points(parana)
points(parana.gr, pch = 19, col = "red", cex = 0.25)
parana.gr0 <- locations.inside(parana.gr, parana$borders)
points(parana.gr0, pch = 19, col = "blue", cex = 0.25)
```



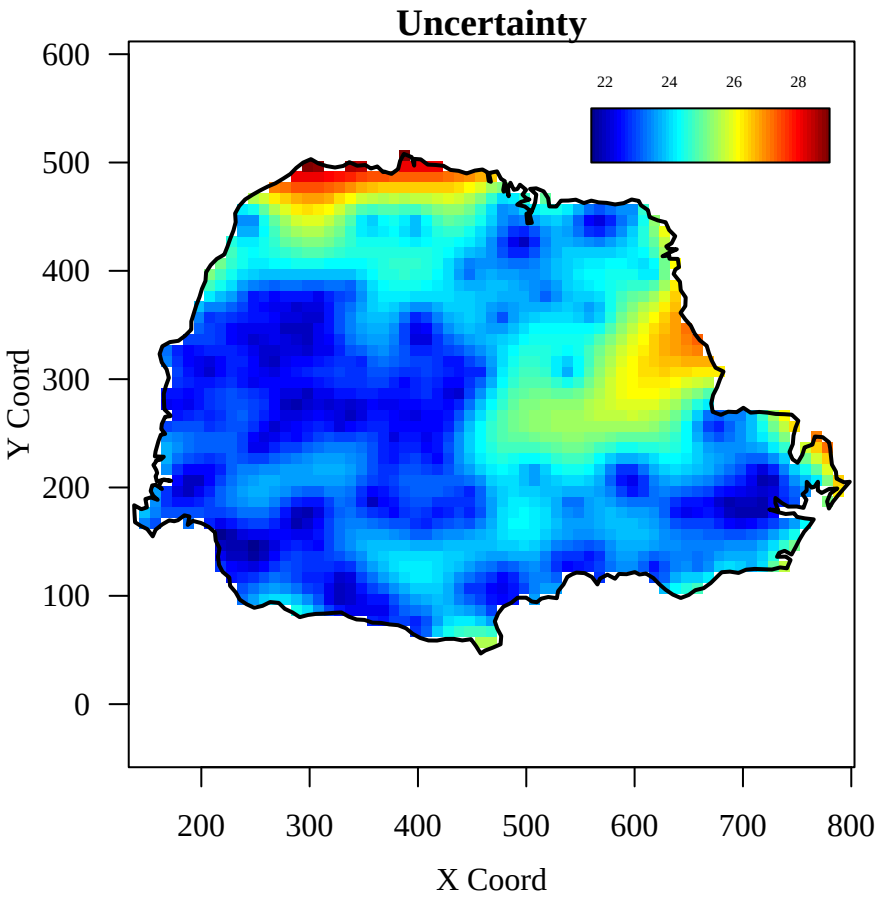

```
KC <- krige.control(obj.m = parana.ml1, trend.d = "1st", trend.l = "1st")
OC <- output.control(simulations = TRUE, n.pred = 1000)
parana.kc <- krige.conv(parana, loc = parana.gr, krige = KC, output = OC)
```

```
## krige.conv: results will be returned only for prediction locations inside the borders
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: sampling from the predictive distribution (conditional simulations)
## krige.conv: Kriging performed using global neighbourhood
```

```
## Spatial prediction and prediction uncertainty
image(parana.kc, col = tim.colors(), x.leg = c(560, 780),
      y.leg = c(500, 550), cex = 0.5, main = "Prediction")
```

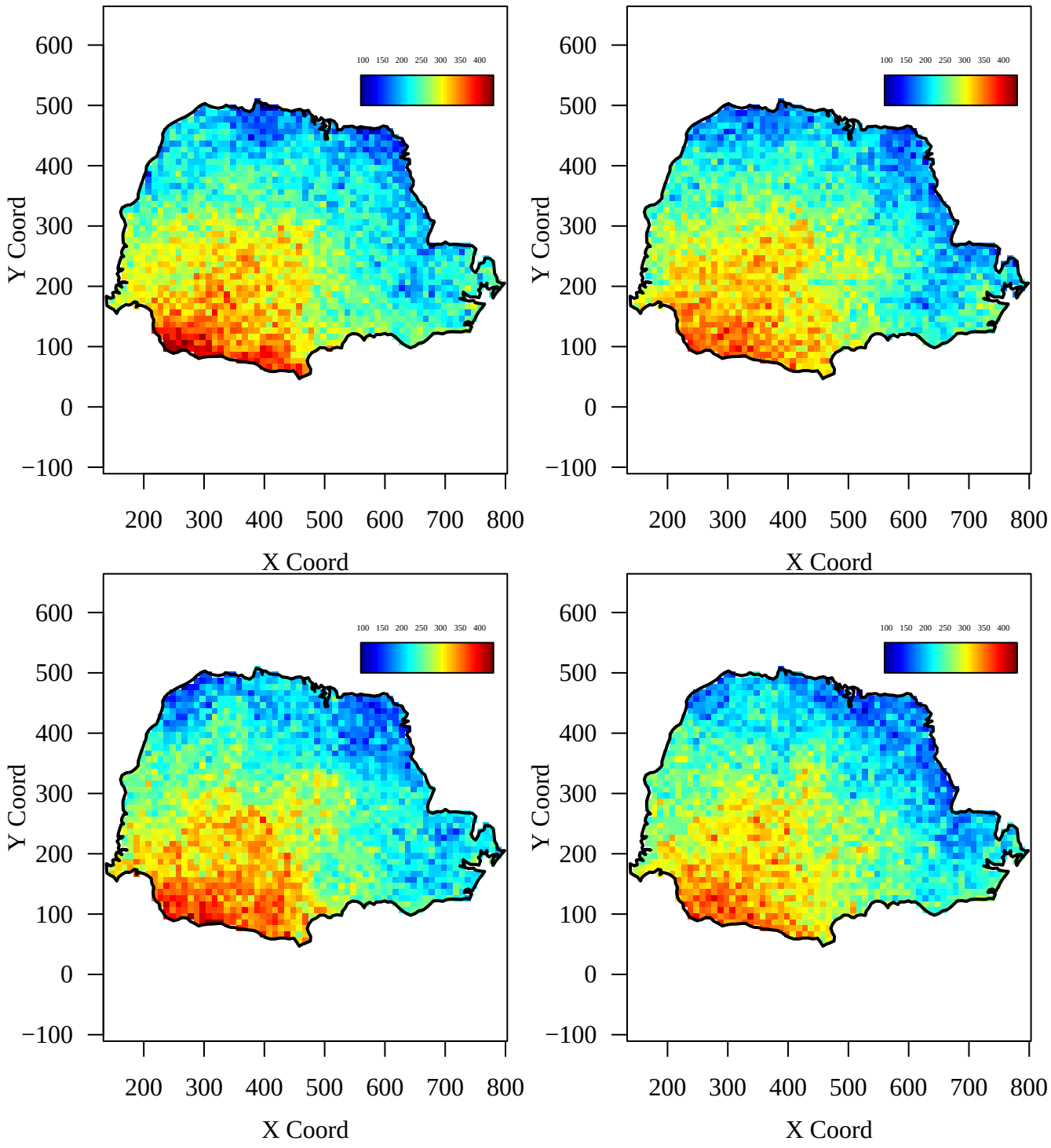



```
image(parana.kc, val = sqrt(parana.kc$krige.var), col = tim.colors(),
      x.leg = c(560, 780), y.leg = c(500, 550), cex = 0.5, main = "Uncertainty")
```



Conditional Simulations

```
par(las = 1, mgp = c(2.4, 1, 0), mar = c(3.5, 3.5, 1, 0.6),
    family = "serif")
for (i in 1:4){
  image(parana.kc, val = parana.kc$simulation[, i], col = tim.colors(),
      x.leg = c(560, 780), y.leg = c(500, 550), cex = 0.35,
      main = "", zlim = range(parana.kc$simulation[, 1:4]))
}
```


100 150 200 250 300 350 400
Y Coord
600
500
400
300
200
100
0
−100
200 300 400 500 600 700 800
X Coord